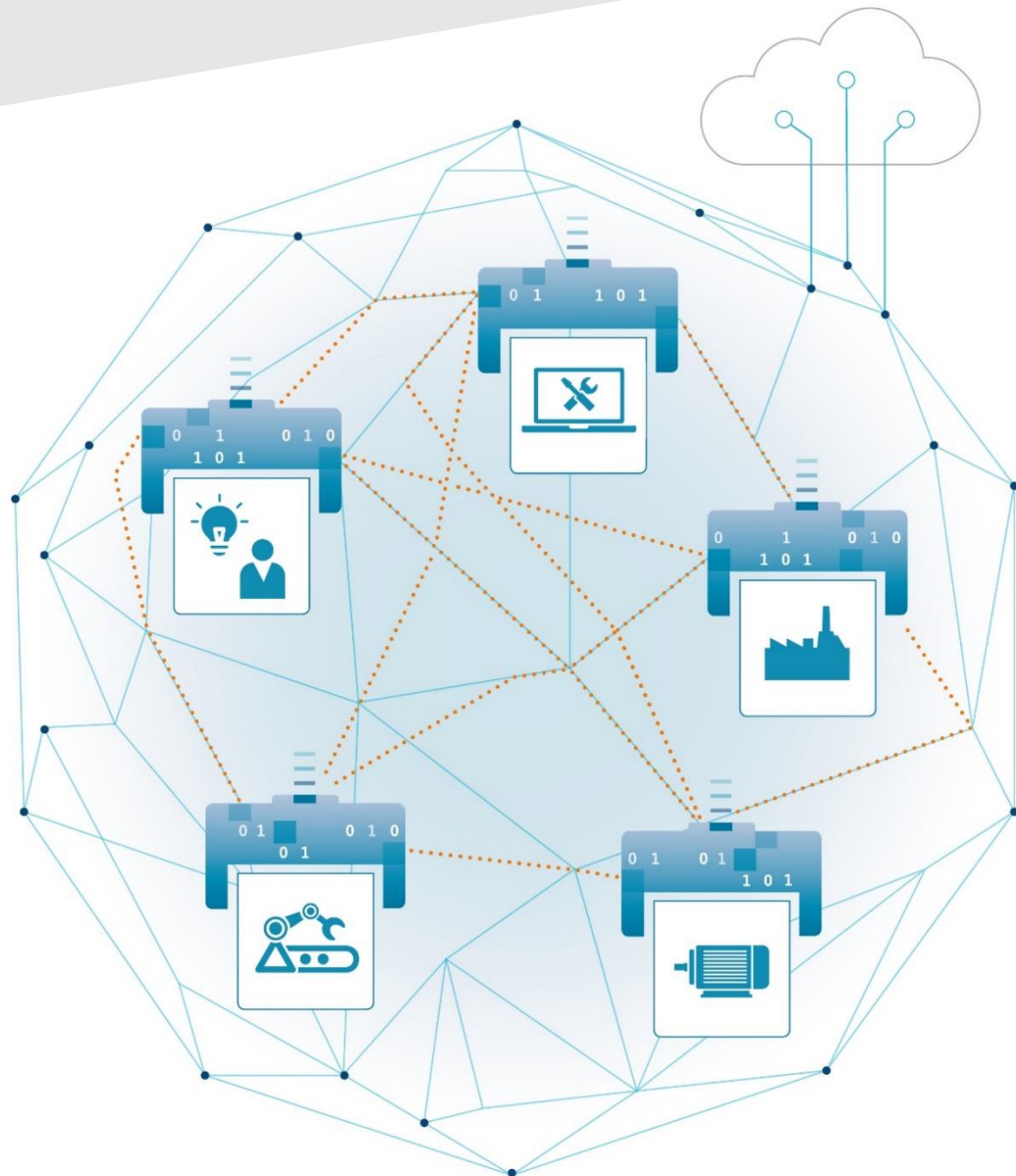


SPECIFICATION

Details of the Asset Administration Shell



in cooperation with:

IDTA

Industrial
Digital
Twin
Association

ZVEI:

Die Elektroindustrie

Part 1 - The exchange of information
between partners in the value chain of
Industrie 4.0 (Version 3.0RC02)

Imprint

Publisher

Federal Ministry for Economic Affairs
and Climate Action (BMWK)

Public Relations

10119 Berlin

www.bmwk.de

Text and editing

Plattform Industrie 4.0

Bülowstraße 78

10783 Berlin

Design and production

The Plattform Industrie 4.0 secretariat, Berlin

Status

Released

Illustrations

Plattform Industrie 4.0; Anna Salari, designed by freepik (Title)

Table of Content

1	Preamble.....	12
1.1	Editorial Notes	13
1.2	Scope of this Document.....	13
1.3	Structure of the Document.....	14
1.4	Principles of the Work.....	14
2	Terms, Definitions and Abbreviations	15
2.1	Terms & Definitions	16
2.2	Abbreviations used in Document.....	20
2.3	Abbreviations of Metamodel	21
3	Introduction	23
4	Basic Concepts and Leading Picture.....	25
4.1	Basic Concepts	26
4.2	Leading Picture	26
5	The Metamodel of the Administration Shell.....	28
5.1	Introduction	29
5.2	Types and Instances.....	29
5.2.1	Life Cycle with Asset Types and Asset Instances	29
5.2.2	Example for Asset Types and Asset Instances	30
5.2.3	Asset Administration Shell Types and Instances.....	31
5.3	Composite I4.0 Components	32
5.4	Identification of Elements.....	33
5.4.1	Overview	33
5.4.2	What Identifiers Exist?	34
5.4.3	Identifiers for Assets and Administration Shells	34
5.4.4	Which Identifiers to use for which Elements	35
5.4.5	How are New Identifiers Created?.....	37
5.4.6	Matching Strategies for Semantic Identifiers	37
5.4.7	Best Practice for Creating URI Identifiers.....	38
5.4.8	Creating a Submodel Instance based on an Existing Submodel Template	39
5.4.9	Can New or Proprietary Submodels be Formed?.....	40
5.4.10	Usage of Short ID for Identifiable Elements	40
5.5	Events	41
5.5.1	Overview	41
5.5.2	Brief Use Cases for Events Used in Asset Administration Shells	41

5.5.3	Input and Output Directions of Events	42
5.5.4	Types of Events	42
5.6	Overview Metamodel of the Administration Shell	44
5.7	Metamodel Specification Details: Designators (normative).....	46
5.7.1	Introduction	46
5.7.2	Common Attributes	47
5.7.3	Asset Administration Shell Attributes.....	57
5.7.4	Asset Information Attributes.....	58
5.7.5	Submodel Attributes	60
5.7.6	Submodel Element Attributes	61
5.7.7	Overview of Submodel Element Types	63
5.7.8	Concept Description Attributes	81
5.7.9	Environment.....	82
5.7.10	Referencing in Asset Administration Shells.....	82
5.7.11	Templates, Inheritance, Qualifiers and Categories	94
5.7.12	Primitive and Simple Data Types.....	95
5.7.13	Cross Constraints and Invariants.....	104
6	Predefined Data Specification Templates	106
6.1	General	107
6.2	Data Specification Template Specification Details: Designators.....	109
6.3	Predefined Template for IEC61360 Properties, Value Lists and Values.....	110
6.3.1	General	110
6.3.2	Overview of Data Specification Template IEC61360.....	112
6.3.3	Data Specification IEC61360 Template Specification Details: Designators.....	113
6.3.4	Category of Concept Descriptions	119
6.4	Predefined Templates for Unit Concept Descriptions	123
6.4.1	General	123
6.4.2	Data Specification Physical Unit Template Specification Details: Designators	125
6.5	Cross Constraints and Invariants for Predefined Data Specifications.....	126
6.5.1	General	126
6.5.2	Constraints for DataSpecificationIEC61360	126
6.5.3	Constraints for DataSpecificationPhysicalUnit	127
7	The Metamodel of the Asset Administration Shell w.r.t. Security.....	128
7.1	General	128
7.2	Passing Access Permissions.....	128
7.3	Overview Metamodel of Administration Shell w.r.t. Security	129
7.4	Metamodel Specification Details: Designators	132

7.4.1	Introduction	132
7.4.2	Security Attributes.....	132
7.4.3	Access Control Policy Point Attributes	133
7.4.4	Access Control Attributes	134
7.4.5	Access Permission Rule Attributes.....	136
7.4.6	Formula Attributes	138
7.4.7	Cross Constraints and Invariants.....	139
8	Package File Format for the Asset Administration Shell (AASX)	140
8.1	General	141
8.2	Basic Concepts of the Open Packaging Conventions	141
8.3	Conventions for the Asset Administration Shell Package File Format (AASX)	142
8.4	ECMA-376 Relationships.....	142
8.5	File Name Conventions	144
8.6	Digital Signatures.....	144
8.7	Encryption.....	145
9	Mappings to Data Formats to Share I4.0-Compliant Information	146
9.1	General	147
9.2	General Rules	148
9.2.1	Introduction	148
9.2.2	Encoding	148
9.2.3	Serialization of Values of Type “Reference”	148
9.2.4	Semantic Identifiers for Metamodel and Data Specifications	148
9.2.5	Embedded Data Specifications.....	150
9.3	XML.....	152
9.4	JSON	152
9.5	RDF.....	152
9.6	AutomationML.....	153
9.7	OPC UA	153
10	Filtering of Information in Export and Import	154
11	Tools for the Asset Administration Shell	156
11.1	Open Source Tools	157
12	Summary and Outlook	158
Annex A.	Concepts of the Administration Shell.....	161
i.	General	161
ii.	Relevant Sources and Documents	161
iii.	Basic Concepts for Industrie 4.0.....	161
iv.	The Concept of Properties.....	162

v.	The Concept of Submodels	163
vi.	Basic Structure of the Asset Administration Shell	164
vii.	Requirements	166
Annex B.	AASX Package File Format – Background Information.....	174
i.	Selection of the Reference Format for the Asset Administration Shell Package Format ...	174
Annex C.	Templates for UML Tables	175
i.	General	175
ii.	Template for Classes	175
iii.	Template for enumerations.....	176
iv.	Template for Primitives	176
v.	Handling of Constraints	176
Annex D.	Legend for UML Modelling.....	177
i.	OMG UML General.....	177
ii.	Notes to Graphical Representation	181
Annex E.	Metamodel UML with Inherited Attributes.....	184
Annex F.	Metamodel Changes.....	186
i.	General	186
ii.	Changes V3.0RC02 vs. V2.0.1	186
a.	Metamodel Changes V3.0RC02 vs. V2.0.1 w/o Security Part	186
b.	Metamodel Changes V3.0RC02 vs. V2.0.1 – Data Specification IEC61360	196
c.	Metamodel Changes V3.0RC02 vs. V2.0.1 – Security Part	198
iii.	Changes V3.0RC02 vs. V3.0RC01	200
a.	Metamodel Changes V3.0RC02 vs. V3.0RC01 w/o Security Part	200
b.	Metamodel Changes V3.0RC02 vs. V3.0RC01 – Data Specification IEC61360	212
c.	Metamodel Changes V3.0RC02 vs. V3.0RC01 – Security Part.....	214
iv.	Changes V3.0RC01 vs. V2.0.1.....	215
a.	Metamodel Changes V3.0RC01 w/o Security Part.....	215
b.	Metamodel Changes V3.0RC01 – Security Part	220
v.	Changes V2.0.1 vs. V2.0	221
a.	Metamodel Changes V2.0.1 w/o Security Part.....	221
b.	Metamodel Changes V2.0.1 – Security Part	222
vi.	Changes V2.0 vs. V1.0	222
a.	Metamodel Changes V2.0 w/o Security Part.....	222
b.	Metamodel Changes V2.0 – Security Part	226
Annex G.	Bibliography	228

Table of Tables

Table 1 Life Cycle Phases and Roles of Asset Type and Instance	29
Table 2 Elements with Allowed Identifying Values	35
Table 3 Proposed Structure for URIs	38
Table 4 Example URN and URL-based Identifiers of the Asset Administration Shell	39
Table 5 Categories for Elements with Value	50
Table 6 Primitive DataTypes Used in Metamodel	95
Table 7 Data Types with Examples	99
Table 8 IEC61360 Data Specification Template for Properties and Ranges	120
Table 9 IEC612360 Data Spec. Template for other Data Elements, Relationships Elements and Capabilities	121
Table 10 IEC612360 Data Specification Template for other Submodel Elements	122
Table 11 Other Elements with semanticId	123
Table 12 Distinction of Different Data Formats for the AAS	147
Table 13 Example Filtering of Information in XML	155
Table 14 Changes w/o Security	187
Table 15 New Elements in Metamodel w/o Security	189
Table 16 New, Changed or Removed Constraints w/o Security	192
Table 17 Changes w.r.t. Data Specification IEC61360	196
Table 18 New Elements in Metamodel DataSpecification IEC61360	196
Table 19 New, Changed or Removed Constraints Data Specification IEC61360	196
Table 20 Changes w.r.t. Security	198
Table 21 New Elements in Metamodel Security	198
Table 22 New, Changed or Removed Constraints Security	198
Table 23 Changes w/o Security	200
Table 24 New Elements in Metamodel w/o Security	203
Table 25 New, Changed or Removed Constraints w/o Security	206
Table 26 Changes w.r.t. Data Specification IEC61360	212
Table 27 New Elements in Metamodel DataSpecification IEC61360	212
Table 28 New, Changed or Removed Constraints Data Specification IEC61360	213
Table 29 Changes w.r.t. Security	214
Table 30 New Elements in Metamodel Security	215
Table 31 New, Changed or Removed Constraints Security	215
Table 32 Changes w.r.t. V2.0 w/o Security	216
Table 33 New Elements in Metamodel V3.0RC01 w/o Security	216
Table 34 New, Changed or Removed Constraints w/o Security	217
Table 35 Changes Metamodel w.r.t. Security	220

Table 36 New Elements in Metamodel Security.....	220
Table 37 New, Changed or Removed Constraints Security.....	221
Table 38 Changes w.r.t. V2.0.1 w/o Security	221
Table 39 New Elements in Metamodel V2.0.1 w/o Security.....	221
Table 40 New, Changed or Removed Constraints w/o Security	222
Table 41 Changes Metamodel w.r.t. V2.0 Security	222
Table 42 New Elements in Metamodel V2.1 w.r.t. V2.0 Security.....	222
Table 43 New, Changed or Removed Constraints w/o Security	222
Table 44 Changes w.r.t. V1.0 w/o Security	223
Table 45 New Elements in Metamodel V1.0 w/o Security.....	224
Table 46 New, Changed or Removed Constraints w/o Security	225
Table 47 Changes Metamodel w.r.t. V1.0 Security	226
Table 48 New Elements in Metamodel w.r.t. Security.....	226
Table 49 New, Changed or Removed Constraints w/o Security	227

Table of Figures

Figure 1 Types of Information Exchange via Asset Administration Shells.....	24
Figure 2 Use Case File Exchange between Value Chain Partners.....	26
Figure 3 File Exchange between two value chain partners.....	27
Figure 4 Exemplary types and instances of assets represented by multiple AAS	30
Figure 5 Exemplary relations between metamodel of AAS, AAS types and AAS instances	32
Figure 6 Extract from Metamodel for Composite I4.0 Components	33
Figure 7 Unique Identifier for Administration Shell and Asset being described (Modified figure from [4]).....	35
Figure 8 Motivation of exemplary identifiers and idShort	40
Figure 9 Forward and Reverse Events.....	41
Figure 10 Tracking of Changes via Events.....	42
Figure 11 Value Push Events across Clouds	42
Figure 12 Overview Metamodel of the Asset Administration Shell.....	44
Figure 13 Metamodel package overview.....	46
Figure 14 Metamodel of HasExtensions.....	47
Figure 15 Metamodel of Referables	48
Figure 16 Metamodel of Identifiables	50
Figure 17 Metamodel of HasKind	51
Figure 18 Metamodel of Administrative Information	52
Figure 19 Metamodel of Semantic References (HasSemantics)	53
Figure 20 Metamodel of Qualifiables.....	54

Figure 21 Metamodel of Qualifiers	54
Figure 22 Metamodel of HasDataSpecification	56
Figure 23 Metamodel AssetAdministrationShell.....	57
Figure 24 Metamodel of Asset Information.....	58
Figure 25 Metamodel of Submodel	60
Figure 26 Metamodel of Submodel Element	61
Figure 27 Metamodel Overview for Submodel Element Subtypes.....	63
Figure 28 Metamodel of Annotated Relationship Elements	64
Figure 29 Metamodel of Basic Event Element	64
Figure 30 Metamodel Event Payload	67
Figure 31 Metamodel of Blobs.....	68
Figure 32 Metamodel of Capabilities	69
Figure 33 Metamodel of Data Elements	69
Figure 34 Metamodel of Entities	70
Figure 35 Metamodel of Events.....	72
Figure 36 Metamodel of File Submodel Element	72
Figure 37 Metamodel of Multi Language Properties	73
Figure 38 Metamodel of Operations	74
Figure 39 Metamodel of Properties	75
Figure 40 Metamodel of Ranges	76
Figure 41 Metamodel of Reference Elements	76
Figure 42 Metamodel of Relationship Elements.....	77
Figure 43 Metamodel of Submodel Element Collections	78
Figure 44 Metamodel of Submodel Element Lists.....	79
Figure 45 Metamodel of Concept Descriptions	81
Figure 46 Metamodel for Environment	82
Figure 47 Metamodel of Reference	83
Figure 48 Logical Model for Keys of References	84
Figure 49 Metamodel of KeyTypes Enumeration	85
Figure 50 DataTypeDefXsd Enumeration.....	98
Figure 51 DefTypeDefRdf Enumeration	98
Figure 52 Built-In Types of XML Schema Definition 1.1 (XSD).....	104
Figure 53 Data Specification Template IEC61360	108
Figure 54 Data Specification Templates.....	109
Figure 55 Example Property from ECLASS	111
Figure 56 Example Property Description with Value List from ECLASS	111
Figure 57 Example Value Description from ECLASS.....	111

Figure 58 Example Value Description from ECLASS Advanced..... 112

Figure 59 Concept Descriptions for Properties Conformant to IEC61360 113

Figure 60 Metamodel of Data Specification IEC6136 114

Figure 61 ValueList..... 117

Figure 62 Categories of Concept Descriptions (non normative) 119

Figure 63 Example of a concept description for a unit: 1/min (from ECLASS) 124

Figure 64 Metamodel of Data Specification Physical Unit..... 125

Figure 65 Attribute Based Access Control [22] 130

Figure 66 Metamodel Overview for Access Control 131

Figure 67 Security Overview Packages..... 132

Figure 68 Metamodel of Security Attributes of AAS 132

Figure 69 Metamodel of Access Control Policy Points..... 133

Figure 70 Metamodel of Access Control 134

Figure 71 Metamodel of Access Permission Rule..... 136

Figure 72 Metamodel of Formulas..... 138

Figure 73 Example Formula “Machine Status not Running” (non normative) 139

Figure 74 Process for generating and consuming AASX packages 141

Figure 75 Relationship Types for AASX Packages 143

Figure 76 Example of an AASX package content - tree view (left) and ECMA-376 relationship types (right) 144

Figure 77 Graphic View on Exchange Data Formats for the Asset Administration Shell..... 147

Figure 78 Realization of Embedded Data Specifications 151

Figure 79 Example Filtering for Export and Import..... 154

Figure 80 Important concepts of Industrie 4.0 attached to the asset [2] [23]. I4.0 Component to be formed by Administration Shell and Asset. 162

Figure 81 Exemplary definition of a property in the IEC CDD 163

Figure 82 Examples of different domains providing properties for submodels of the Administration Shell .. 164

Figure 83 Basic structure of the Asset Administration Shell..... 165

Figure 84 Class..... 177

Figure 85 Inheritance/Generalization 177

Figure 86 Multiplicity 178

Figure 87 Ordered Multiplicity..... 178

Figure 88 Association 178

Figure 89 Composition (composite aggregation) 178

Figure 90 Aggregation 179

Figure 91 Navigable Attribute Notation for Associations 179

Figure 92 Default Value 179

Figure 93 Dependency 179

Figure 94 Abstract Class	179
Figure 95 Package.....	180
Figure 96 Imported Package	180
Figure 97 Enumeration	180
Figure 98 Data Type	180
Figure 99 Primitive Data Type	180
Figure 100 Note	181
Figure 101 Constraint	181
Figure 102 Graphical Representations of Composite Aggregation/Composition	181
Figure 103 Graphical Representation of Shared Aggregation	182
Figure 104 Graphical Representation of Generalization/Inheritance	183
Figure 105 Graphical Representation for Enumeration with Inheritance	183
Figure 106 Graphical Representation for deprecated classes	183
Figure 107 Core Model with Inherited Attributes	184
Figure 108 Model for Submodel Elements with Inherited Inheritance	185

1 Preamble

1.1 Editorial Notes

This document, version 3.0RC02, was produced from November 2020 to May 2022 by the sub working group “Asset Administration Shell” of the joint Working Group of Platform Industrie 4.0 Working Group “Reference Architectures, Standards and Norms” and the “Open Technology” Working Group of the Industrial Digital Twin Association.

Version 3.0RC01 of this document, published in November 2020, was produced from November 2019 to November 2020 by the sub working group “Asset Administration Shell” of the Platform Industrie 4.0 Working Group “Reference Architectures, Standards and Norms”.

The second version V2.0 of this document was produced August 2018 to November 2019 by the sub working group “Asset Administration Shell” of the Platform Industrie 4.0 Working Group “Reference Architectures, Standards and Norms”. Version 2.0.I was published in May 2020.

The first version of this document was produced September 2017 to July 2018 by a joint working group with members from ZVEI SG “Models and Standards” and Plattform Industrie 4.0 Working Group “Reference Architectures, Standards and Norms “. The document was subsequently validated by the platform’s Working Group “Reference Architectures, Standards and Norms”.

For better readability, in compound terms the abbreviation "I4.0" is consistently used for "Industrie 4.0". Used on its own "Industrie 4.0" continues to be used.

This specification is versioned using Semantic Versioning 2.0.0 (semver) and follows the semver specification [48].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 RFC2119 RFC8174.

1.2 Scope of this Document

The aim of this document is to make selected specifications of the structure of the Administration Shell in such a way that information about assets and I4.0 Components can be exchanged in a meaningful way between partners in a value creation network.

This part of this document therefore focuses on the question of how such information needs to be processed and structured. In order to make these specifications, the document formally stipulates a few structural principles of the Administration Shell. This part does not describe technical interfaces of the Administration Shell or other systems to exchange information, protocols or interaction patterns.

This document focuses on:

- ▶ Metamodel for specifying information of an Asset Administration Shell and its submodels
- ▶ Exchange format for the transport of information from one partner in the value chain to the next
- ▶ Identifiers
- ▶ Access Control
- ▶ Introduction to the need of mappings to suitable technologies to be used in different life cycle phases of a product: XML, JSON, RDF, AutomationML and OPC UA

This document assumes some familiarity with the concept of the Asset Administration Shell. For convenience some of the concepts are repeated in the Annex A. The concepts are being standardized as IEC standard IEC 63278 series [59]. The main stakeholders addressed in this document are architects and software developers aiming to implement a digital twin using the Asset Administration Shell in an interoperable way. Additionally, the content can also be used as input for discussions with international standardization organisations and further collaborations. For an overview on documents on the Asset Administration Shell please see the continuously updated reading guide [50]. The reading guide gives advice which documents to read depending on the role of the reader.

1.3 Structure of the Document

Clause 2 provides terms and definitions as well as abbreviations, both for abbreviations used in the document and for abbreviations that may be used for elements of the metamodel defined in this document.

Clause 3 gives a short introduction into the content of this document.

Clause 4 summarises relevant, existing content from the standardization of Industrie 4.0. In other words, this clause provides an overview and explains the motives, and is not absolutely necessary for an understanding of the subsequent definitions.

Clause 5 stipulates sufficient structural principles of the Administration Shell in a formal manner in order to ensure an exchange of information between the Administration Shells. An excerpt of a UML diagram is drafted for this purpose. A more comprehensive UML discussion which does not set standards can be found in the annex. Security topics are discussed in Clause 7.

Predefined data specifications, including those for defining concept descriptions, are specified in Clause 6.

Clause 7 describes the promotion of attribute based access control (ABAC) for information security.

Clause 8 describes, how the information of one or more Administration Shells can be packed into a compound file format (AASX). Background information with respect to this format can be found in Annex B.

Clause 9 provides information on the exchange of information compliant to this specification in existing data formats like XML, AutomationML, OPC UA information models, JSON or RDF.

Clause 10 deals with filtering information before exchange with external partners.

In Clause 11 hints on existing open source tools that can be used for editing, implementing or managing Asset Administration Shells are given.

Finally, Clause 12 summarizes the content and gives an outlook on future work.

The Annex contains additional background information on Asset Administration Shell (Annex A). In the Annex also information about UML (Annex D) and the tables used to specify UML classes as used in this specification (Annex C) are contained.

Metamodel changes compared to previous versions are described in Annex F. For developers there are also selected metamodel diagrams including all inherited attributes provided in Annex E.

The bibliography can be found in Annex G.

1.4 Principles of the Work

The work is based on the following principle: keep it simple but do not simplify if it affects interoperability.

For creating a detailed specification of the Administration Shell according to the scope of part 1 result papers published by Plattform Industrie 4.0, the Trilateral cooperation with France and Italy and international standardization results were analysed and taken as source of requirements for the specification process. As many ideas as possible from the discussion papers were considered. See also Annex A ii for more information.

The partners represented in the Plattform Industrie 4.0 and the Industrial Digital Twin Association (IDTA) and associations such as the ZVEI, the VDMA, VDI/ VDE and Bitkom, ensure that there is broad sectoral coverage, both in process, hybrid and factory automation and in terms of integrating information technology (IT) and operational technology (OT).

Design alternatives were intensively discussed within the working group. An extensive feedback process of this document series is additionally performed within the working groups of Plattform Industrie 4.0 and the IDTA.

Guiding principle for the specification was to provide detailed information, which can be easily implemented also by small and medium sized enterprises.

2 Terms, Definitions and Abbreviations

2.1 Terms & Definitions

Forward notice:

Definition of terms are only valid in a certain context. This glossary applies to the context of this document.

access control

protection of system resources against unauthorized access; a process by which use of system resources is regulated according to a security policy and is permitted by only authorized entities (users, programs, processes, or other systems) according to that policy

→ [SOURCE: IEC TS 62443-1-1]

application

software functional element specific to the solution of a problem in industrial-process measurement and control

Note: An application can be distributed among resources and may communicate with other applications.

→ [SOURCE: IEC TR 62390:2005-01, 3.1.2]

asset

physical or logical object owned by or under the custodial duties of an organization, having either a perceived or actual value to the organization

Note: In the case of industrial automation and control systems, the physical assets that have the largest directly measurable value can be the equipment under control.

→ [SOURCE: IEC TS 62443-1-1:2009, 3.2.6]

Asset Administration Shell (AAS)

standardized *digital representation* of the *asset*, corner stone of the interoperability between the applications managing the manufacturing systems. It identifies the Administration Shell and the assets represented by it, holds digital models of various aspects (*submodels*) and describes *technical functionality* exposed by the Administration Shell or respective assets.

Note: Asset administration shell and Administration shell are used synonymously.

→ [SOURCE: Glossary Industrie 4.0]

attribute

data element of a *property*, a relation, or a class in information technology

→ [SOURCE: ISO/IEC Guide 77-2, ISO/IEC 27460, IEC 61360]

class

description of a set of objects that share the same *attributes*, *operations*, methods, relationships, and semantics

→ [SOURCE: IEC TR 62390:2005-01, 3.1.4]

capability

implementation-independent potential of an Industrie 4.0 component to achieve an effect within a domain

Note 1: Capabilities can be orchestrated and hierarchically structured.

Note 2: Capabilities can be made executable via services.

Note 3: The impact manifests in a measurable effect within the physical world

→ [SOURCE: Glossary Industrie 4.0]

component

product used as a constituent in an assembled product, *system* or plant

→ [SOURCE: IEC 61666:2010, 3.6]

concept

unit of knowledge created by a unique combination of characteristics

→ [SOURCE: IEC 61360-1, ISO 22274:2013, 3.7]

digital representation

information that represents characteristics and behaviors of an entity

Note 1: Information is data that within a certain context has a particular meaning. Data is content represented in a digital and formalized manner suitable for communication, storage, interpretation or processing
 Note 2: Behavior includes functionality (description and execution)

→ SOURCE: [IIC IIC:IIVOC:V2.3:20201025, adapted (attributes changed to characteristics + notes added)]

digital twin

digital representation, sufficient to meet the requirements of a set of use cases

Note: in this context, the entity in the definition of digital representation is typically an asset

→ [SOURCE: IIC Vocabulary IIC:IIVOC:V2.3:20201025, adapted (an asset, process or system was changed to an asset)]

identifier (ID)

identity information that unambiguously distinguishes one entity from another one in a given domain

Note: There are specific identifiers, e.g. UUID Universal unique identifier, IEC 15418 (GS1).

→ [SOURCE: Glossary Industrie 4.0]

instance

concrete, clearly identifiable component of a certain *type*

Note 1: It becomes an individual entity of a type, for example a device, by defining specific property values.
 Note 2: In an object-oriented view, an instance denotes an object of a class (of a type).

→ [SOURCE: IEC 62890:2016, 3.1.16 65/617/CDV]

operation

executable realization of a function

Note 1: The term method is synonym to operation
 Note 2: an operation has a name and a list of parameters [ISO 19119:2005, 4.1.3]

→ [SOURCE: Glossary Industrie 4.0]

ontology

an explicit specification of a (shared) conceptualization

→ [SOURCE: Gruber "A Translation Approach to portable ontology specifications", Knowledge acquisition 5.2 (1993): 199-220]

property

defined characteristic suitable for the description and differentiation of products or components

- Note 1: The concept of type and instance applies to properties.
- Note 2: This definition applies to properties such as described in IEC 61360/ ISO 13584-42
- Note 3: The property types are defined in dictionaries (like IEC component Data dictionary or ECLASS), they do not have a value. The property type is also called data element type in some standards.
- Note 4: The property instances have a value, and they are provided by the manufacturers. A property instance is also called property-value pair in certain standards.
- Note 5: Properties include nominal value, actual value, runtime variables, measurement values, etc.
- Note 6: A property describes one characteristic of a given object.
- Note 7: A property can have attributes such as code, version, and revision.
- Note 8: The specification of a property can include predefined choices of values.

→ [SOURCE:according ISO/IEC Guide 77-2] as well as [SOURCE:according Glossary Industrie 4.0]

qualifier

well-defined element associated with a *property* instance or *submodel element*, restricting the value statement to a certain period of time or use case

Note: qualifier can have value associated

→ [SOURCE: according to IEC 62569-1]

variable

software *entity* that may take different values, one at a time

→ [SOURCE: IEC 61499-1]

smart manufacturing

manufacturing approach, that improves its performance aspects with integrated and intelligent use of processes and resources in cyber, physical and human spheres to create and deliver products and services, which also collaborates with other domains within an enterprise's' value chains.

- Note 1: Performance aspects include agility, efficiency, safety, security, sustainability or any other performance indicators identified by the enterprise.
- Note 2: In addition to manufacturing, other enterprise domains can include engineering, logistics, marketing, procurement, sales or any other domains identified by the enterprise.
- Note 3: this definition is, as of November 2020, under discussion within the ISO/ IEC joint working group (JWG) 21. However, it gives a good indication and a citable source.

→ [SOURCE: ISO/TMB/SMCC]

submodel

models which are technically separated from each other and which are included in the *Asset Administration Shell*

- Note 1: Each submodel refers to a well-defined domain or subject matter. Submodels can become standardized and thus become submodels templates.
- Note 2: Submodels can have different life cycles.
- Note 3: The concept of template and instance applies to submodels.

→ [SOURCE: Glossary Industrie 4.0]

submodel element

element suitable for the description and differentiation of *assets*

Note 1: extends the definition of *properties*
Note 2: could comprise *operations*, binary objects

→ [SOURCE: Glossary Industrie 4.0]

system

interacting, interrelated, or interdependent elements forming a complex whole

→ [SOURCE: IEC TS 62443-1-1:2009, 3.2.123]

technical functionality

functionality of the *Administration Shell* that is exposed by an application programming interface (API) and that is creating added value to the respective *assets(s)*.

Note: can consist of single elements, which are also known as functions, operations, methods, skills.

→ [SOURCE: according [18]]

template

specification of the common features of an object in sufficient detail that such object can be instantiated using it

Note: object can be anything that has a type

→ [SOURCE: according ISO/IEC 10746-2]

type

hardware or software element which specifies the common *attributes* shared by all instances of the type

→ [SOURCE: IEC TR 62390:2005-01, 3.1.25]

view

projection of a model or models, which is seen from a given perspective or vantage point and omits *entities* that are not relevant to this perspective

→ [SOURCE: Unified Modelling Language - UML]

2.2 Abbreviations used in Document

Abbreviation	Description
AAS	Asset Administration Shell
AASX	Package file format for the AAS
AML	AutomationML
API	Application Programming Interface
BITKOM	Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V.
BLOB	Binary Large Object
CDD	Common Data Dictionary
GUID	Globally unique identifier
I4.0	Industrie 4.0
ID	identifier
IDTA	Industrial Digital Twin Association
IEC	International Electrotechnical Commission
IRDI	International Registration Data Identifier
IRI	Internationalized Resource Identifier
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
MIME	Multipurpose Internet Mail Extensions
OPC	Open Packaging Conventions (ECMA-376, ISO/IEC 29500-2)
OPC	Open Platform Communications
OPCF	OPC Foundation
OPC UA	OPC Unified Architecture
PDF	Portable Document Format
RAMI4.0	Reference Architecture Model Industrie 4.0
RDF	Resource Description Framework
REST	Representational State Transfer
RFC	Request for Comment
SOA	Service Oriented Architecture
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTC	Universal Time Coordinated
VDI	Verein Deutscher Ingenieure e.V.
VDE	Verband der Elektrotechnik Elektronik Informationstechnik e.V.
VDMA	Verband Deutscher Maschinen- und Anlagenbau e.V.
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

Abbreviation	Description
ZIP	archive file format that supports lossless data compression
ZVEI	Zentralverband Elektrotechnik- und Elektronikindustrie e. V.

2.3 Abbreviations of Metamodel

The following abbreviations are not used in the document but may be used as abbreviations for the elements in the metamodel defined in this document.

Abbreviation	Description
AAS	Asset Administration Shell
Cap	Capability
CD	Concept Description
DE	DataElement
DST	DataSpecification Template
InOut	inputoutputVariable
In	inputVariable
Prop	Property
MLP	MultiLanguageProperty
Range	Range
Ent	Entity
Evt	Event
File	File
Blob	Blob
Opr	Operation
Out	outputVariable
Qfr	Qualifier
Ref	ReferenceElement
Rel	RelationshipElement
RelA	AnnotatedRelationshipElement
SM	Submodel
SMC	SubmodelElementCollection
SME	SubmodelElement
SML	SubmodelElementList

Abbreviation	Description
Sec	Security
ACPP	Access Control Policy Points
PAP	Policy Administration Point

Abbreviation	Description
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Points
AC	Access Control
APR	Access Permission Rule
PpO	Permissions per Object

3 Introduction

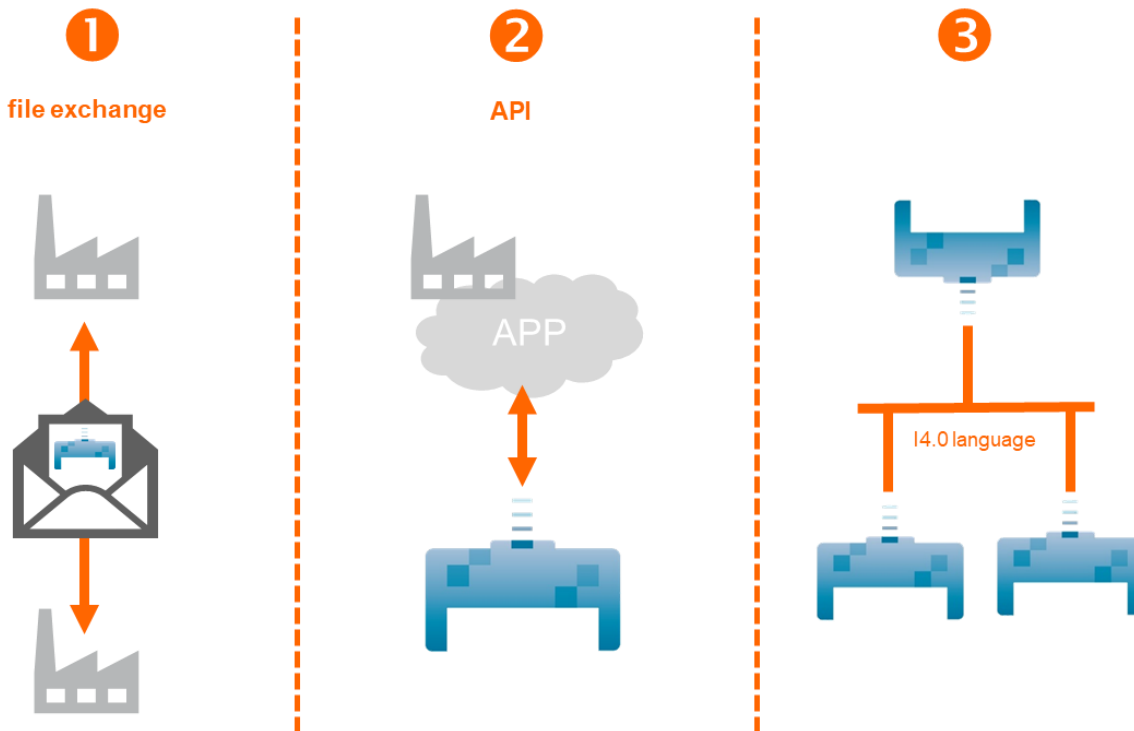
In this document the information model (meta model) of the Asset Administration Shell together with a file exchange format is specified.

Besides a technology neutral specification of the information model in UML, several different formats for exchanging Asset Administration Shells are provided: XML, JSON, RDF, AutomationML as well as an OPC UA information model.

Figure 1 shows the different ways for information exchange via Asset Administration Shells. This part of the “Asset Administration Shell in Detail” series deals mainly with type 1: file exchange. For enabling exchange between partners the following steps need to be executed:

- 1) Definition of the Asset Administration Shell in a selected format, for example XML as explained in this document.
- 2) Selecting of the additional files that are referenced in submodels of the Asset Administration Shell and should be exchanged as well.
- 3) Provision of the Asset Administration Shell together with the selected files in a standardized exchange format, the AASX package format as specified in this document.
- 4) Defining a secure way to exchange the file, for example via secure file download on a web-server [53].

Figure 1 Types of Information Exchange via Asset Administration Shells



© Plattform Industrie 4.0

However, the information model specified in this document is not only used to exchange complete Asset Administration Shells via file exchange. It is also the basis for exchanging of information via a standardized API (type 2 in Figure 1). The API is specified in part 2 of the document series [49].

4 Basic Concepts and Leading Picture

4.1 Basic Concepts

Many concepts for Industrie 4.0 and smart manufacturing are already existing. The most important ones are summarised in the informative Annex A.

4.2 Leading Picture

The leading use case in this document is the exchange of an Asset Administration Shell including all its auxiliary documents and artifacts from one value chain partner to another. This is, in this document we do not deal with the use case of already deployed Asset Administration Shells running in a specific infrastructure but only with file exchange between partners.

Figure 2 Use Case File Exchange between Value Chain Partners

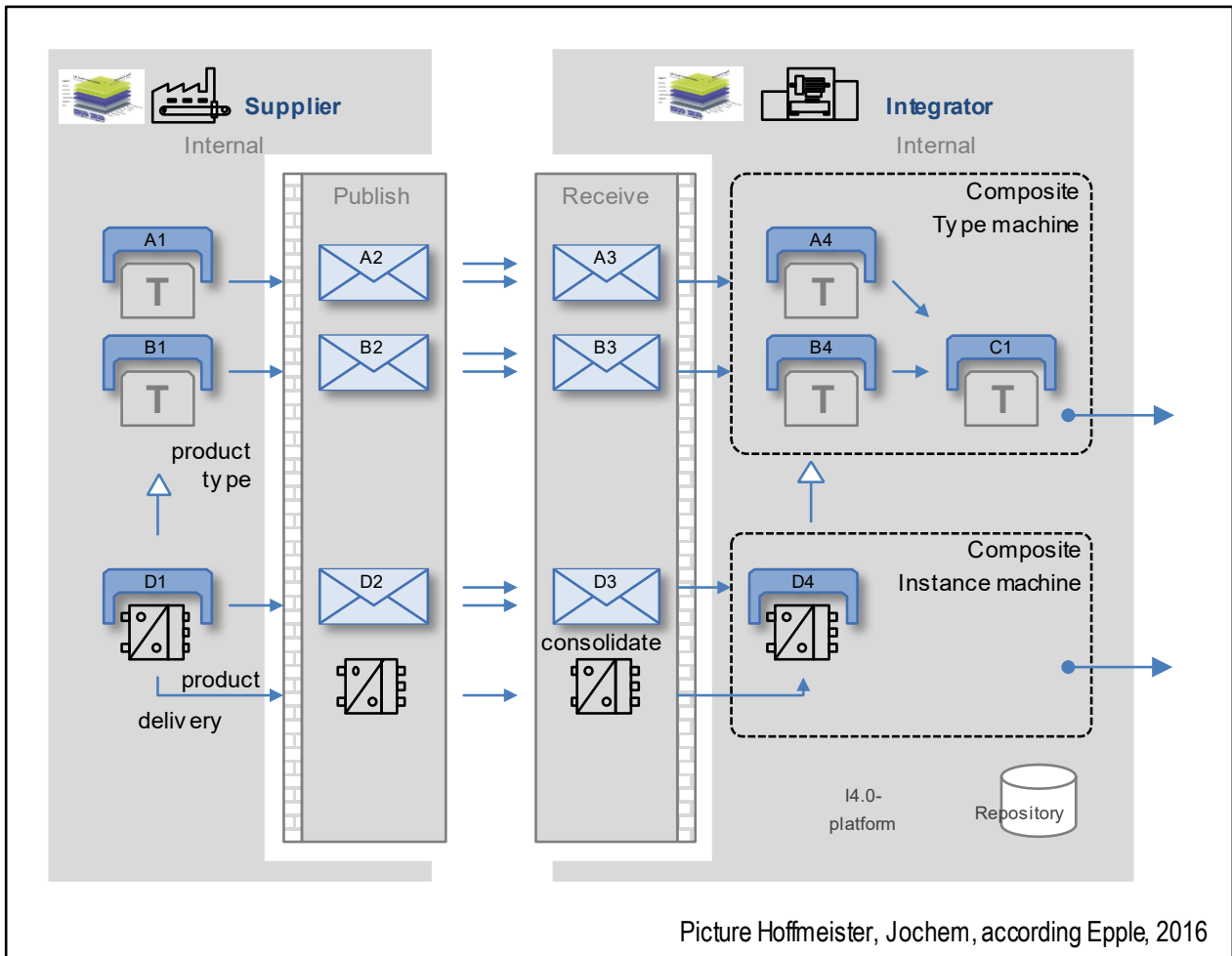
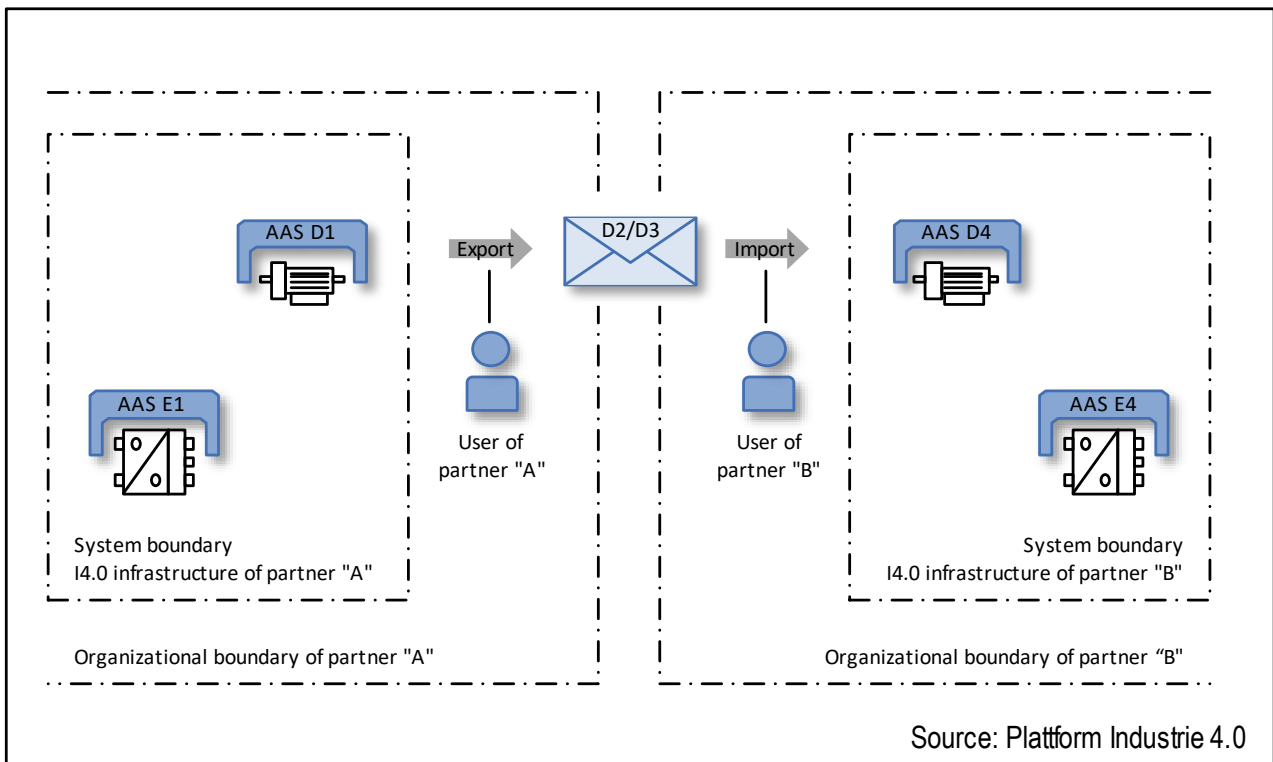


Figure 2 shows the overall picture. It depicts two value chain partners: "Supplier" is going to provide some products, "Integrator" is going to utilize this products in order to build a machine. Two kinds of Administration Shells are being provided; one for the asset being the type of a product, one for the assets being the actual product instances. "Supplier" and "Integrator" are forming two independent legal bodies (Figure 3).

Figure 3 File Exchange between two value chain partners



The exchange of files needs to fulfil some requirements with respect to usability and security. There needs to be a bilateral agreement on security constraints to be fulfilled for the transfer and usage of the files. This is explained in more detail in Clause 7.

For usability a container format for exchanging files is used and a corresponding structure is defined (see Clause 7.4.6). This predefined structure helps the consumer to understand the content of the single files. This is important because an Asset Administration Shell specification can be spread across several files. Additionally, the container may contain auxiliary files references by the AAS or even executable code.

5 The Metamodel of the Administration Shell

5.1 Introduction

This clause specifies the information metamodel of the Asset Administration Shell (AAS). Before doing so, some general aspects of the handling of asset types and instances are described (see Subclause 5.2 Types and Instances). In Subclause 5.3 handling of composite i4.0 components is explained. Another very important aspect of the AAS is the identification aspect, see Subclause 5.4. In Subclause 5.5 aspects of event handling are discussed.

An overview of the metamodel of the Asset Administration Shell is given in Subclause 5.6. In Subclause 5.7 the classes are described in detail together with all their attributes.

The metamodel for security aspects of the Administration Shell is described in Clause 7.

The legend for understanding the UML diagrams and the table specification of the classes is explained in Annex C and Annex D.

An xmi representation of the UML model can be found in the repository "aas-specs" in the github project admin-shell-io [41]: <https://github.com/admin-shell-io/aas-specs/tree/master/schemas/xmi>

5.2 Types and Instances

5.2.1 Life Cycle with Asset Types and Asset Instances

Industrie 4.0 utilizes an extended understanding of asset, comprising elements such as factories, production systems, equipment, machines, components, produced products and raw materials, business processes and orders, immaterial assets (such as processes, software, documents, plans, intellectual property, standards), services and human personnel, and more.

The RAMI4.0 model [3] features one, generalized life-cycle axis, which was derived from IEC 62890. The basic idea is to distinguish for all assets within Industrie 4.0 between possible types and instance. This makes it possible to apply the type/instance distinction for all elements such as material type/material instance, product type/product instance, machine type/ machine instance and more. Business related information will be handled on the 'Business' layer of the RAMI4.0 model. The business layer also covers order details and workflows, again with asset types/instances.

Table 1 Life Cycle Phases and Roles of Asset Type and Instance

Phase		Description
Asset Type	Development	Valid from the ideation/conceptualization to first prototypes/test. The 'type' of an asset is defined, and distinguishing properties and functionalities are defined and implemented. All (internal) design artefacts are created, such as CAD data, schematics, embedded software, and associated with the asset type.
	Usage / Maintenance	Ramping up production capacity. The 'external' information associated to the asset is created, such as technical data sheets, marketing information. The selling process starts.
Asset Instance	Production	Asset instances are created/ produced, based on the asset type information. Specific information about production, logistics, qualification and test are associated with the asset instances.
	Usage / Maintenance	Usage phase by the purchaser of the asset instances. Usage data is associated with the asset instance and might be

		<p>shared with other value chain partners, such as the manufacturer of the asset instance.</p> <p>Also included: maintenance, re-design, optimization and decommissioning of the asset instance. The full life-cycle history is associated with the asset and might be archived/shared for documentation.</p>
--	--	---

Table 1 gives an overview of the different life cycle phases and the role of asset type and asset instance in these phases: The most important relationship is between asset types and asset instance. This relationship should be maintained throughout the life of the asset instances. By this relationship, updates to the asset types can be forwarded to the asset instances, either automatically or on demand.

Note: For the distinction of asset 'type' and asset 'instance', the term 'asset kind' is used in this document.

The second class of relationships are feedback loops/information within the life cycle of the asset type and instance. For product assets, for example, information on usage and maintenance of product instances may be used to improve the manufacturing of products as also the (next) product type.

The third class of relationships are feedforward/ information exchange with assets of other asset classes. For example, sourcing information from business assets can influence design aspects of products; or the design of the products affects the design of the manufacturing line.

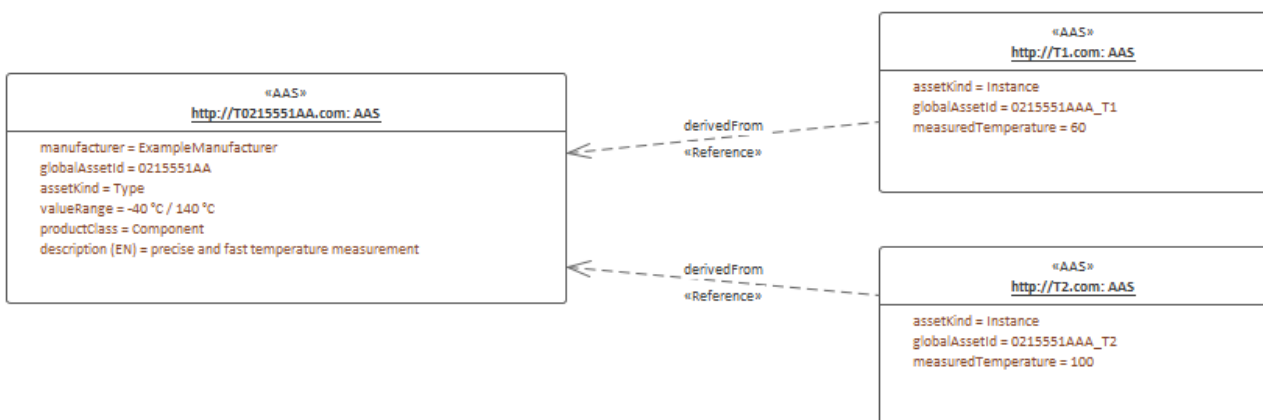
Note: For an illustration of the second/ third class of relationships confer the NIST model, as well.

A fourth class of relationships are between asset of different hierarchy levels. For example, these could be the (dynamic) relationships between manufacturing stations and products being currently produced. These could be also the decompositions of production systems in physical, functional or safety hierarchies. By this class of relationships, automation equipment is explained as a complex, interrelated graph of automation devices and products, performing intelligent production and self-learning/ optimization tasks.

5.2.2 Example for Asset Types and Asset Instances

The following figure gives an example for the handling of asset types and asset instances, and for the handling of some exemplary information as well. Further explanation will follow in the next clauses.

Figure 4 Exemplary types and instances of assets represented by multiple AAS



Note: The example is simplified for ease of understanding and does only roughly comply to the metamodel as it is specified in Clause 5. The ID handling is simplified as well: the names of the classes correspond to the unique global identifier of the AASs.

- Note: In the context of Platform Industrie 4.0 types and instances typically refer to "asset types" and "asset instances". When referring to types or instances of an AAS this is explicitly denoted as "AAS types" and "AAS instances" to not mix up both. AAS types are synonymously used with the term "AAS template".
- Note: Please refer to Clause 2 for the IEC definition of types and instances. For the scope of this document, there is no full equivalency between these definitions and the type/instance concepts of object-oriented programming (OO).

There shall be a concrete asset type of a temperature sensor and two uniquely identifiable physical temperature sensors of this type. The intention is to provide a separate AAS for the asset type as well as for every single asset instance.

In the example, the first sensor has the unique ID "0215551AA_T1" and the second sensor has the unique ID "0215551AA_T2". The AAS for the first sensor has the unique URI "http://T1.com/T1" and the AAS for the second sensor has the unique URI "http://T2.com/T2". The asset kind of both is "Instance". The example shows that the measured temperature at operation time of the two sensors is different: for T1 it is 60 °C, for T2 it is 100 °C. For the time-being we ignore the relationship "derivedFrom" of the two AAS "T1" and "T2" with AAS "http://T0215551AA.com".

- Note: Even though the HTTP scheme is used in the example, the URIs do not need to be valid URLs and therefore do not need to point to accessible content.
- Note: The physical unit can be obtained by the semantic reference of the element "measuredTemperature". For simplicity this is not shown in the example.

These two asset instances do have a lot of information they share: the information of the asset type (in this example a sensor type). For this asset type an own AAS is created. The unique ID for this AAS is "http://T0215551AA.com", the unique ID of the sensor type is "0215551AA". The asset kind in this case is "Type" and not "Instance". The information that is the same for all instances of this temperature sensor type is the ProductClass (= "Component"), the manufacturer (= "ExampleManufacturer") and the English Description "=precise and fast temperature measurement" as well as the value range "-40 °C / 140 °C".

Now the two AAS of the two asset instances may refer to the AAS of the asset type "0215551AA" using the relationship attribute "derivedFrom".

- Note: "attribute" refers in the UML sense to the property or characteristic of a class (instance).
- Note: Typically, if a specific asset type does exist, it exists in time before the respective asset instances.
- Note: The term AAS is used synonymously to the term AAS instance. An AAS may be realized based on an AAS type. AAS types are out of scope of this document.
- Note: In public standardization, the AAS Types might be standardized. However, it is much more important to standardize the property types (called property definitions or concept descriptions) or other submodel element typed as well as complete submodel types because those can be reused in different AAS.
- Note: In the domain of the internet of things (IoT), asset instances are typically denoted as "Things" whereas asset types are denoted as "Product".

5.2.3 Asset Administration Shell Types and Instances

In the previous clause, type and instances of assets were explained. Obviously, the question then comes up how to harmonize AAS as well as AAS types. In our example, it can be seen that the attributes "globalAssetId" and "assetKind" as well as the global AAS identifier (*id*, represented as name of the class) are present for all AAS. However, if there is no standard, it is not clear that the semantics of "id", "globalised" and "kind" are the same for all AAS and it is not clear, which of the attributes are mandatory and which are specific for the asset (type or instance). This is illustrated in Figure 5.

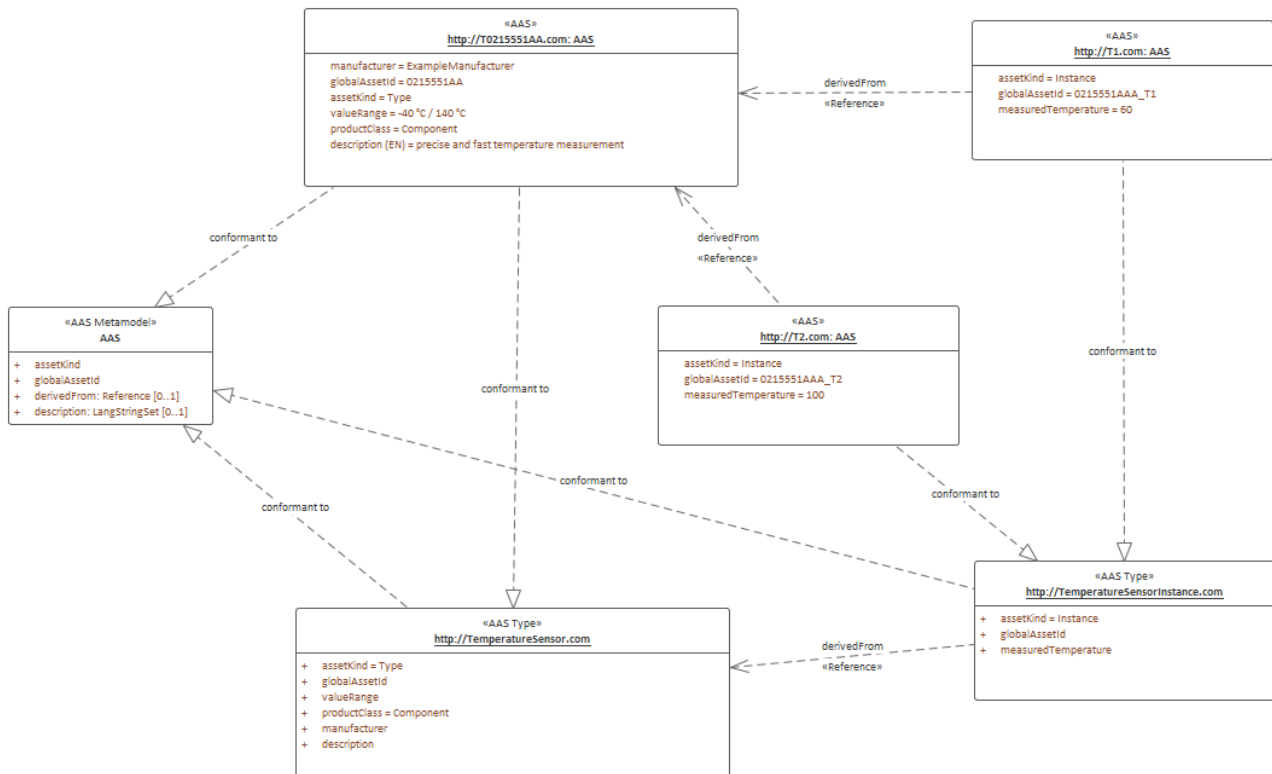
This is the purpose of this document: The definition of a metamodel that defines which attributes are mandatory and which are optional for all AAS. The Platform Industrie 4.0 metamodel for Asset Administration Shells is defined in Clause 5.

- Note: This approach ensures that requirement tAAS-#19 is fulfilled. Another approach could have been to define two metamodels: one for asset types and one for asset instances. However, the large set of similarities motivated to go with one metamodel.
- Note: The metamodel itself does not prescribe mandatory submodels. This is another step of standardization similar to the prescription of submodels of AAS Type level.

Note: An AAS type shall be realized based on the metamodel of an AAS as defined in this document. This Metamodel is referred to as the “AAS Metamodel”.

Note: It is not mandatory to define an AAS type before defining an AAS (instance). An AAS instance that does not realize an AAS type shall be realized based on the Metamodel of an AAS as defined in this document.

Figure 5 Exemplary relations between metamodel of AAS, AAS types and AAS instances



5.3 Composite I4.0 Components

As described in Clause 5.2.1 there is a class of relationships between assets of different hierarchy levels. By this class of relationships, automation equipment is explained as a complex, interrelated graph of automation devices and products, performing intelligent production and self-learning/ optimization tasks.

Details and examples for composite I4.0 Components can be found in [12].

The following modelling elements in the AAS metamodel can be used to realize such composite I4.0 Components:

- *RelationshipElement* – used to describe relationships between assets and other elements
- *Submodel* A complex asset is composed out of other entities and assets. These entities and assets together with their relationship to each other are specified in a bill of material.

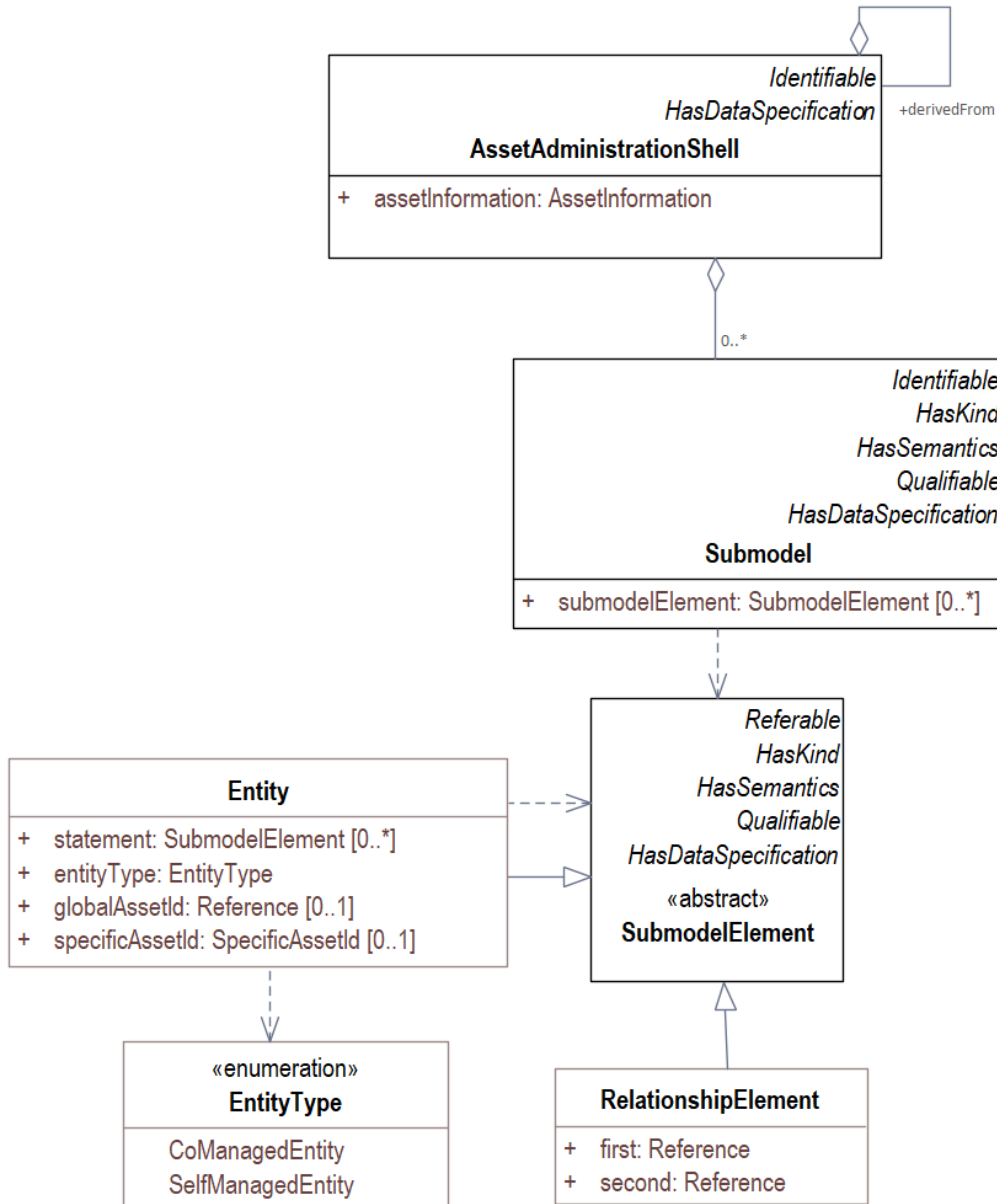
Note: The submodel template defining the structure of such a bill of material is not predefined by the AAS metamodel but is assumed to contain Entity elements.

- Not every entity (*Entity*) that is part of the bill of material of an asset has necessarily its own Asset Administration Shell. As described in [12] self-managed entities are distinguished from co-managed entities (*Entity/entityType*).
 - Self-Managed Entities have their own AAS. This is why a reference to this asset is specified as well (via *Entity/globalAssetId* or an *Entity/specificAssetId*). Additionally, further property statements (*Entity/statement*) (compare to [15]) can be added to the asset that are not specified in the AAS of the asset itself because they are specified in relation to the composite I4.0 Component only.

- For co-managed entities there is no separate AAS. The relationships and property statements of such entities are managed within the AAS of the composite I4.0 Component.

Figure 6 shows an extract of the metamodel that is introduced later containing the elements being the most important to describe composite I4.0 Components.

Figure 6 Extract from Metamodel for Composite I4.0 Components



5.4 Identification of Elements

5.4.1 Overview

Identifiers are needed according to [4] for the unique identification of many different elements within the domain of smart manufacturing. For this reason, they are a fundamental element of a formal description of the Administration Shell. Especially, identification is at least required for:

- Asset Administration Shells,
- Assets (as value of AssetAdministrationShell/assetInformation/globalAssetId),

- submodel instances and submodel templates,
- property definitions/concept descriptions in external repositories, such as ECLASS or IEC CDD

Identification will take place for two purposes:

- (1) to uniquely distinguish all elements of an Administration Shell and the asset it is representing, and
- (2) to relate elements to external definitions, such as submodel templates and property definitions, in order to bind a semantics to these data and functional elements of an Administration Shell.

5.4.2 What Identifiers Exist?

In [4], [20] two standard-conforming global identification types are defined:

- IRDI** - ISO29002-5, ISO IEC 6523 and ISO IEC 11179-6 [20] as an identifier scheme for properties and classifications. They are created in a process of consortium-wise specification or international standardization. To this end, users sit down together and feed their ideas into the consortia or standardization bodies. Properties in ISO, IEC help to safeguard key commercial interests. Repositories like ECLASS and others make it possible to standardise a relatively large number of identifiers in an appropriately short time.
- IRI** – IRI (Rfc 3987¹) or URI and URL according to RFC 3986² as identification of assets, Administration Shells and other (probably not standardized, but globally unique) properties and classifications.

The following is also permitted:

- Custom** - Internal custom identifiers such as UUIDs/GUIDs (globally unique identifiers/universally unique identifier³), which a manufacturer can use for all sorts of in-house purposes within the Administration Shell.

This means that the IRIs/URIs/URLs and internal custom identifiers can represent and communicate manufacturer-specific information and functions in the Administration Shell and the 4.0 infrastructure just as well as standardized information and functions. One infrastructure can serve both purposes.

CLSID are URIs for GUIDs. They start with a customer specific schema. So Custom should really only be used if the customer specific identifier is no IRDI nor an IRI.

Besides the global identifiers, there are also identifiers that are unique only within a defined namespace, typically its parent element. These identifiers are also called local identifiers. Example: Properties within a submodel have local identifiers.

Besides absolute URIs there are also relative URIs.

See also DIN SPEC 91406 [43] for further information on identification.

5.4.3 Identifiers for Assets and Administration Shells

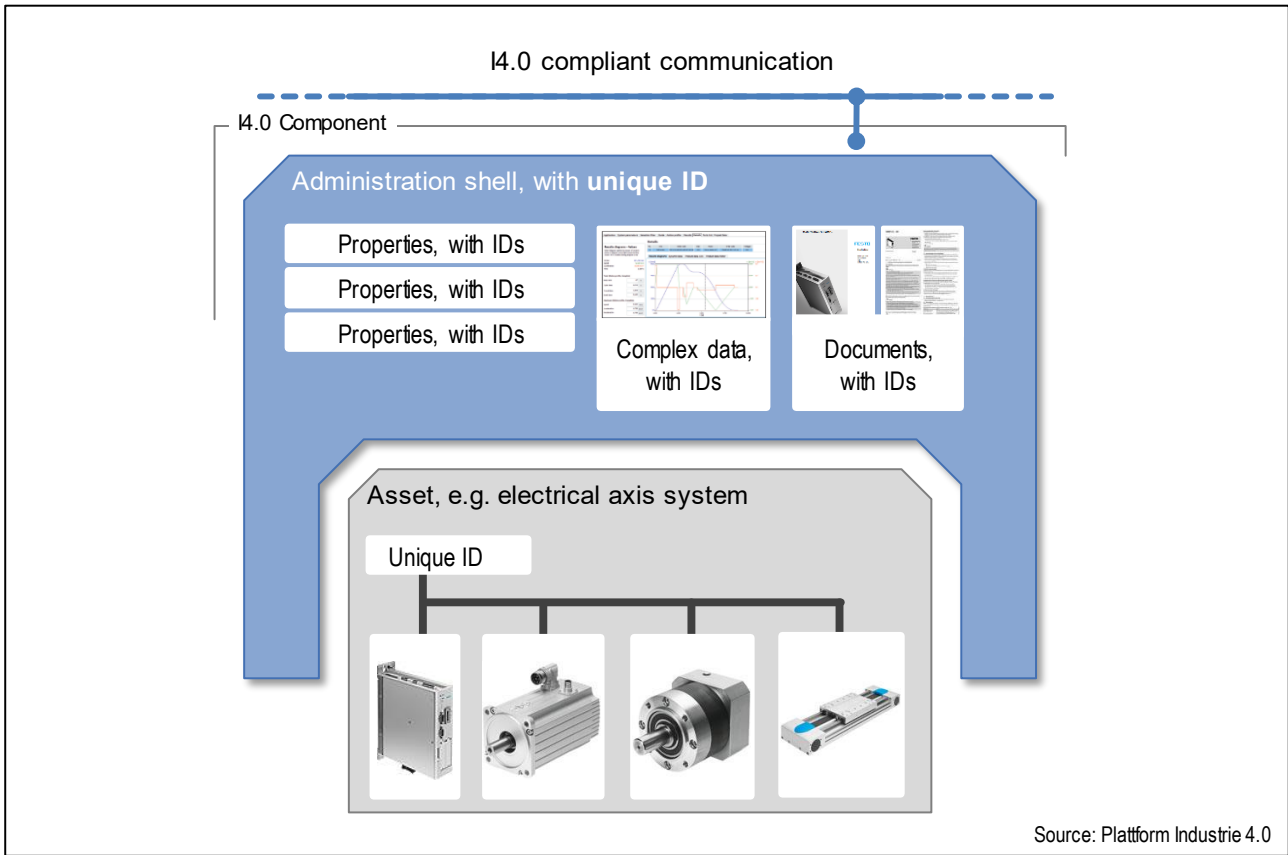
For the domain of smart manufacturing, the assets need to be uniquely identified worldwide [4] [20] by the means of identifiers (IDs). The Administration Shell has a unique ID, as well.

¹ <https://tools.ietf.org/html/rfc3987>

² <https://tools.ietf.org/html/rfc3986>

³ https://en.wikipedia.org/wiki/Universally_unique_identifier.

Figure 7 Unique Identifier for Administration Shell and Asset being described (Modified figure from [4])



An Administration Shell represents exactly one asset, with a unique asset ID. In a batch-based production, the batches will become the assets and will be described by a respective Administration Shell. If a set of assets shall be described by an Administration Shell, a unique ID for the composite asset needs to be created [12].

The ID of the asset needs to comply with the restrictions for global identifiers according to [4][20]. If the asset is featuring further identifications, serial numbers and alike, they are not to be confused with the unique global identifiers of the asset itself⁴.

5.4.4 Which Identifiers to use for which Elements

Not every identifier is applicable for every element of the UML model representing the Asset Administration Shell. Table 2 therefore gives an overview on the different constraints and recommendations on the various entities, which implement "Identifiable" or "HasSemantics". Attributes relate to the metamodel in Clause 5.6 and Clause 5.7.

Table 2 Elements with Allowed Identifying Values

Elements with identifying values	Attribute	Allowed identifiers (recommended or typical)	Remarks
AssetAdministration Shell	id	IRI (URL)	mandatory Typically, URLs will be used

⁴ Such additional asset identifiers are contained in *AssetInformation/specificAssetIds*.

Elements with identifying values	Attribute	Allowed identifiers (recommended or typical)	Remarks
	idShort	string	optional ⁵
Submodel with kind = Template	id	IRDI, IRI (URI)	mandatory IRDI, if the defined submodel is standardized and an IRDI was applied for it
	idShort	string	recommended Typically used as idShort for the submodel of kind Instance as well
	semanticId	IRDI, IRI (URI)	optional The semantic ID might refer to an external information source, which explains the formulation of the submodel (for example an PDF if a standard)
Submodel with kind = Instance	id	IRI (URI), Custom	mandatory
	idShort	string	recommended Typically, the idShort or English short name of the submodel template referenced via semanticId
	semanticId	IRDI, IRI (URI)	recommended Typically, the semantic is an external reference to an external standard defining the semantics of the submodel.
SubmodelElement	idShort	string	mandatory Typically the English short name of the concept definition referenced via semanticId
	semanticId	IRDI, IRI (URI), Custom	recommended link to a <i>ConceptDescription</i> or the concept definition in an external repository via a global ID
ConceptDescription	id	IRDI, IRI, Custom	mandatory <i>ConceptDescription</i> needs to have a global ID. If the concept description is a copy from an external dictionary like ECLASS or IEC CDD it may use the same global ID as it is used in the external dictionary.
	idShort	string	recommended e.g. same as English short name
	isCaseOf	IRDI, IRI (URI)	optional links to the concept definition in an external repository the concept description is a copy from or that it corresponds to

⁵ Note: In version V1.0 of this specification idShort was optional for Identifiables. This changed in V2.0: idShort was set to mandatory for all Referables. With V3.0RC02 idShort again was made optional.

Elements with identifying values	Attribute	Allowed identifiers (recommended or typical)	Remarks
Qualifier	semanticId	IRDI, IRI (URI), Custom	recommended Links to the qualifier type definition in an external repository IRDI, if the defined qualifier type is standardized and an IRDI was applied for it

5.4.5 How are New Identifiers Created?

Following the different identification types from Clause 5.4.3, it can be stated:

- (a) IRDIs are assumed to be already existing by an external specification and standardization process, when it comes to the creation of a certain Administration Shell. For bringing such IRDI identifiers into life, refer to Clause 5 of the document [4].
- (b) URIs and URLs can easily be formed by developers themselves, also on the fly when creating a certain Administration Shell. All that is needed is a valid authority, for example of the company, and to make sure that the way in which the domain (e.g. admin-shell.io) is organised ensures that the path appended to the host's name is reserved in a semantically unique way for these identifiers. In this way, each developer can create an arbitrary URI or URL by combining the host name and some chosen path, which only needs to be unique in the developer's organisation.
- (c) Custom identifiers can also be easily formed by developers themselves. All that is necessary is a corresponding programmatic functionality to be retrieved. It is necessary to ensure that internal custom identifiers can be clearly distinguished from (a) or (b).
- (d) Local identifiers can also be created on the fly. They have to be unique within their namespace.

5.4.6 Matching Strategies for Semantic Identifiers

When comparing two elements there are different use cases to be considered to be able to state how these two elements are semantically related. This chapter just gives first hints which aspects to consider when dealing with matching semantic identifiers. For example semantic references including context information like possible with IRDI-Path in ECLASS are not yet considered.

- **Exact Matching (identical semanticIds) – DEFAULT**
 - With exact matching two semantic ids need to be string-identical
 - Example: Property with idShort "ManufacturerName" + semanticId 0173-1#02-AAO677#002 and Property with idShort "Herstellername" + semanticId 0173-1#02-AAO677#002 have exact equal semantics
- **Intelligent Matching (compatible semanticIds)**
 - *Ignore Versioning*
 - With intelligent matching different versions of a concept definition may be matched. In case of semantic versioning only compatible, i.e. upward or backward compatible, versions should be matched.
Example 1: Property with idShort "ManufacturerName" + semanticId 0173-1#02-AAO677#002 and Property with idShort "Herstellername" + semanticId 0173-1#02-AAO677#003 have equal semantics. Note: For comparing two semantic ids knowledge about versioning needs to be available. In the example two IRDIs from ECLASS are compared: ECLASS rules ensure that the semantics is always backward compatible for new versions; for breaking changes a new IRDI would be created.
 - *Consider Semantic Mappings*
 - With intelligent matching existing semantic mapping information can be considered. Semantic mappings may exist within one and the same dictionary but also between different dictionaries and ontologies.

Example: 0112/2///61360_4#AAE530 for nominal capacity of a battery in dictionary IEC CDD and 0173-1#02-AAI048#004 in ECLASS have equal semantics^{6 7 8}.

- o *Consider Domain Knowledge*
 - With intelligent matching domain knowledge available in machine readable form may be taken into account, for example a “is-a”-relationship between two concept definitions.
Example: A Hammer drill (0173-1#01-ADS698#010) and a percussion drill (0173-1#01-ADS700#010) are drills for mineral material (0173-1#01-ADN177#005) and thus are compatible to a request or constraints asking for drills for mineral material.

5.4.7 Best Practice for Creating URI Identifiers

The approach for semantics and interaction for I4.0 components [17] suggests the use of the following structure (see Table 3) for URIs⁹, which is slightly modified here. The idea is to always structure URIs following a scheme of different elements. However, this is just a recommendation and not mandatory to be used.

Table 3 Proposed Structure for URIs

Element	Description	Syntax component
Organisation	Legal body, administrative unit or company issuing the ID	A
Organisational subunit/ Document ID/ Document subunit	Sub entity in organisation above, or released specification or publication of organisation above.	P
Submodel / Domain-ID	Submodel of functional or knowledge-wise domain of asset or Administration Shell, the identifier belongs to.	P
Version	Version number in line with release of specification or publication of identifier	P
Revision	Revision number in line with release of specification or publication of identifier	P
Property / Element-ID	Property or further structural element ID of the Administration Shell	P
Instance number	Individual numbering of the instances within release of specification or publication	P

In the table, syntax component "A" refers to authority of RFC 3986 (URI) and namespace identifier of RFC 2141 (URN); "P" refers to path of RFC 3986 (URI) and namespace specific string of RFC 2141 (URN).

```

<AAS URI> ::= <scheme> ":" <authority> [ <path> ]
<scheme> ::= a valid URI scheme
<authority> ::= <Organisation>
<path> ::= <subunit> <domain> <release> <element>
    
```

⁶ Note: This example is not representing an existing semantic mapping but would just be a candidate.

⁷ Semantic mapping files are also used in ECLASS between ECLASS Classic and ECLASS Advanced: https://wiki.eclass.eu/wiki/Transaction_Update_File

⁸ This is the format used for semantic mapping in ECLASS: <https://www.eclass.eu/static/eClassXML/3.0/eCI@ssXML/mapping.xsd>

⁹ URLs are also URIs

<subunit> ::= [("/" | ":") <Organisational Subunit/Document ID/Document subunit>]*

<domain> ::= [("/" | ":") <Submodel / Domain-ID>

<release> ::= [("/" | ":") <Version> [("/" | ":") <Revision>]*]

<element> ::= [("/" | ":" | "#") (<Property/Element-ID> | <Instance number>)*]

Using this scheme, valid URNs and URLs can be created, both being URIs. For the use of Administration Shells, URLs are preferred as well, as functionality (such as REST services) can be bound to the identifiers. Examples of such identifiers are given in Table 4.

Table 4 Example URN and URL-based Identifiers of the Asset Administration Shell

Identifier	Description	Property class	Examples
Administration Shell ID	ID of the Administration Shell	Basis	urn:zvei:SG2:aas:1:1:demo11232322 http://www.zvei.de/SG2/aas/1/1/demo11232322
Submodel ID (Type)	Identification of type of submodel	Selected submodels are basis, others free	urn:GMA:7.20:contractnegotiation:1:1 http://www.vdi.de/gma720/contractnegotiation/1/1
Submodel ID (Instance)	Identification of the instance of the submodel	Free	urn:GMA:7.20:contractnegotiation:1:1#001 http://www.vdi.de/gma720/contractnegotiation/1/1#001
Property/parameter /status type IDs	Identification of the property, parameter and status types	Domain-specific	urn:PROFIBUS:PROFIBUS-PA:V3-02:Parameter:1:1:MaxTemp http://www.zvei.de/SG2/aas/1/1/demo11232322/maxtemp
Property/parameter /status instance IDs (not used by metamodel)	Identification of the property, parameter and status instance	Domain-specific	urn:PROFIBUS:PROFIBUS-PA:V3-02:Parameter:1:1:MaxTemp#0002 http://www.zvei.de/SG2/aas/1/1/demo11232322/maxtemp#0002

Note: the last row of Table 4 is only used for completion; the metamodel does not foresee identifiers for property/parameter/status instances.

5.4.8 Creating a Submodel Instance based on an Existing Submodel Template

In order to instantiate an existing submodel template, there should be a public specification of the submodel template, e.g. via publication by Plattform Industrie 4.0. As a special case, instantiating a submodel from a non-public submodel template, such as a manufacturer specification, is also possible.

In November 2020 the first two submodel templates for the Asset Administration Shell were published, one for a nameplate ([52]) and one for generic technical data ([51]). Others followed and will follow. For an overview of registered submodel templates see [60].

In each submodel template, the identifiers of concept definitions to be used as semantic references are already predefined. An instantiation of such a submodel merely requires the creation of properties each with a semantic reference to the property definition and attach a value. The same holds for other subtypes of submodel elements.

The only thing that cannot be defined in the template itself is the unique ID of the submodel instance itself (it is not identical with the ID of the submodel template) as well as the property values etc. Templates also define cardinalities, for example whether an element is optional or not. For submodel element lists typically more than one element is contained: in the template an exemplary element template is contained. The other elements can be created by copy/paste from this template.

5.4.9 Can New or Proprietary Submodels be Formed?

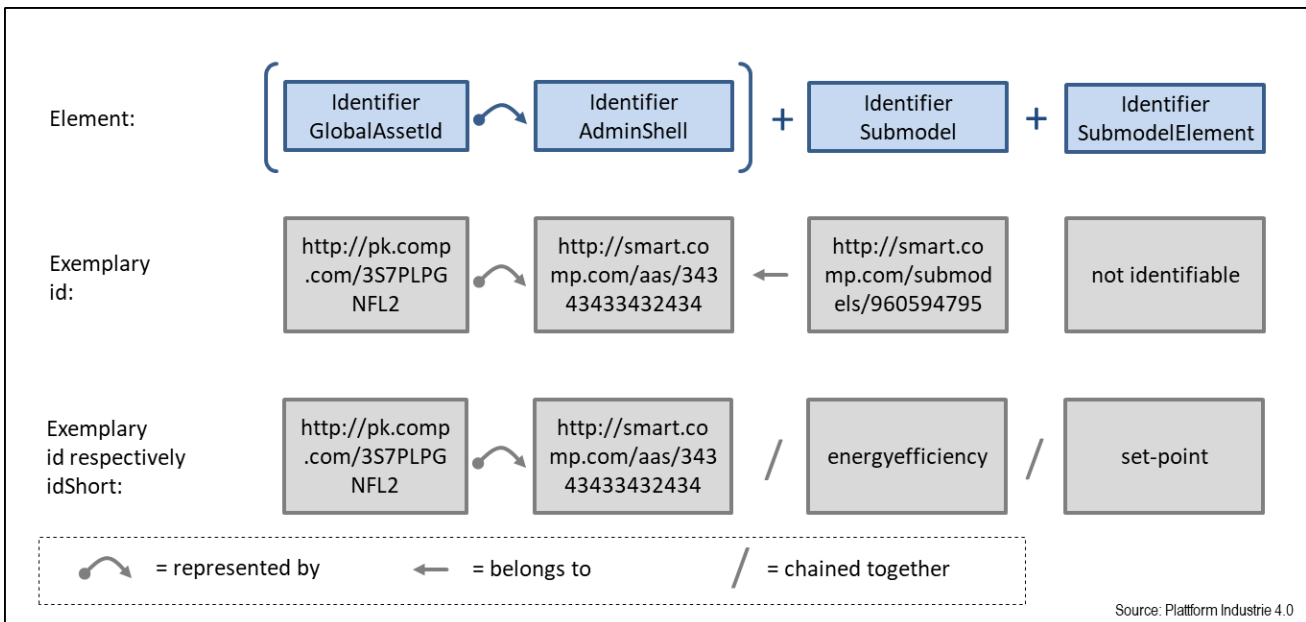
It is in the interest of Industrie 4.0 for as many submodels as possible, including free and proprietary submodels, to be formed (→ [4], “Free property sets”). A submodel can be formed at any time for a specific Administration Shell of an asset. For this purpose, the provider of the Administration Shell can form in-house identifiers for the type and instance of the submodel in line with Section 5.4.5. All I4.0 systems are called on to ignore submodels and properties that are not individually known, and simply to “overlook” them. For this reason, it is always possible to deposit proprietary – e.g. manufacturer-specific or user-specific – information, submodels or properties in an Administration Shell.

Note: It is in the intention of the Administration Shell, that proprietary information is included as well. For example, to link to company-wide identification schemes or information required for company-wide data processing. By this, a single infrastructure can be used to transport standardized and proprietary information at the same time; this conveys the introduction (and later standardization) of new information elements as well.

5.4.10 Usage of Short ID for Identifiable Elements

The Administration Shell fosters the use of worldwide unique identifiers to a large degree. However, in some cases, this may lead to inefficiencies. An example might be referring to a property, which is part of a submodel which is part of an Administration Shell and each of these identified by global identifiers [4]. For example, in an application featuring a resource-oriented architecture (ROA), a worldwide unique resource locator (URL) might be composed of a series of segments, which in turn do not need to be worldwide unique:

Figure 8 Motivation of exemplary identifiers and idShort



In order to allow such efficient addressing of elements by an API of an Administration Shell, idShort is provided for a set of classes of the metamodel, which inherit from abstract class *Referable*, in order to refer to such dependent elements (→ 5.6). However, an external system addressing resources of an Administration Shell is required to check the respective semantics, i.e. the value of *semanticId*, first, before accessing elements by *id* or *idShort* (→ 5.7.2).

5.5 Events

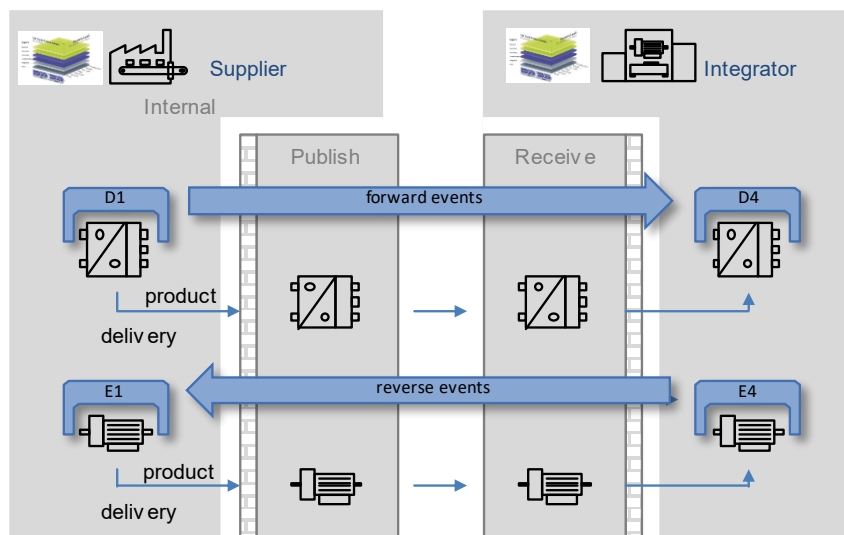
5.5.1 Overview

Events are a very versatile mechanism of the AAS. In the following sections, first some use-cases for events are described. Different types of events are summarized in order to depict requirements. A *SubmodelElement* "Event" is introduced, which is able to declare events of an AAS. The general format of event messages is specified.

5.5.2 Brief Use Cases for Events Used in Asset Administration Shells

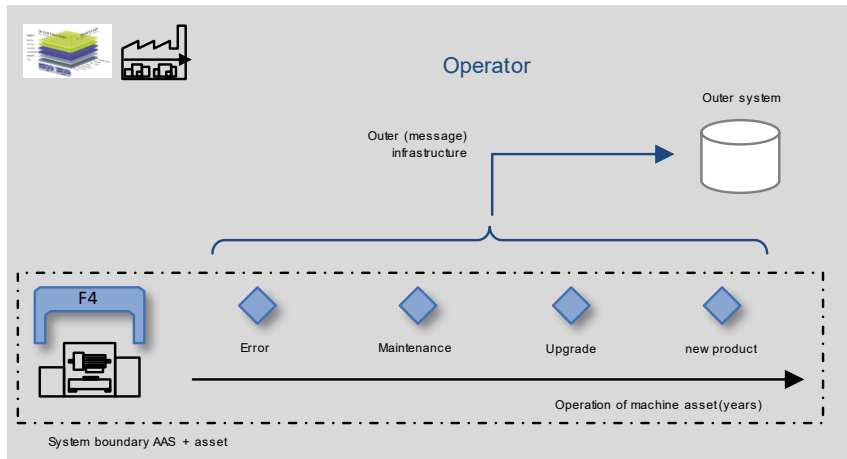
- An integrator has purchased a device. Later in time, the supplier of the device provides a new firmware. The integrator wants to detect the offer of a new firmware and wants to update the firmware after evaluating its suitability ("forward events"). The mechanism is, that a dependent AAS ("D4") detects events from a parent or type AAS ("D1"), which is described by the *derivedFrom* relation.
- An integrator/ operator operates a motor purchased from a supplier. During operation, condition monitoring incidents occur. Both parties agree on a business model providing availability. So, the supplier wants to monitor device statuses which are further in the value chain ("reverse events").

Figure 9 Forward and Reverse Events



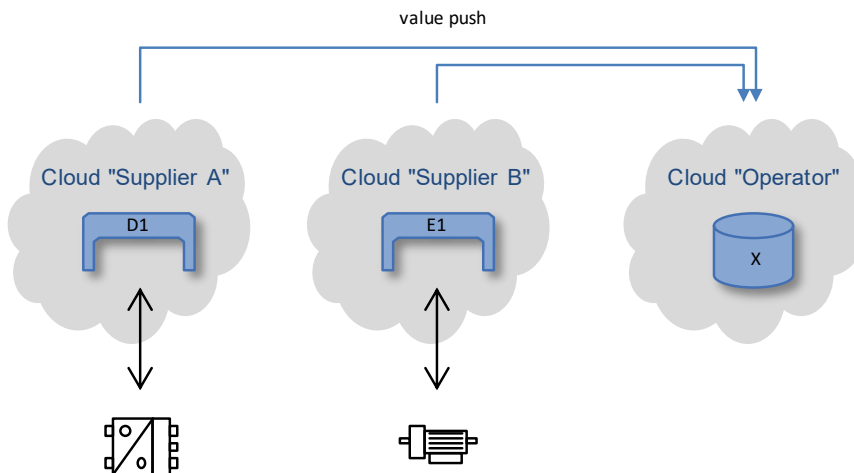
- An operator is operating a certain I4.0 component over time. Changes occasionally occur to these I4.0 components from different systems. For documentation and auditing, changes to this I4.0 component shall be tracked. This can be achieved by recording events over time.

Figure 10 Tracking of Changes via Events



- An operator is operating different I4.0 components, which are deployed to manufacturer clouds. The operator wants to integrate data from these components, according to DIN SPEC 92222. Therefore, information needs to be forwarded to the operator cloud ("value push").

Figure 11 Value Push Events across Clouds



5.5.3 Input and Output Directions of Events

It may be relevant to distinguish between input and output directions of an event with respect to the observed model, the respective Referable.

Direction	Descriptions
Output	The event is monitoring the <i>Referable</i> it is attached to. An outer message infrastructure, e.g. by OPC UA or MQTT or AMQP, will transport these events to other AASs and further outer systems and users.
Input	The software entity, which implements the respective <i>Referable</i> , can handle incoming events. These incoming events will be delivered by an outer message infrastructure, e.g. by OPC UA or MQTT or AMQP, to the software entity of the <i>Referable</i> .

5.5.4 Types of Events

According to the above use-cases, different types of events are possible. The following table gives an impression on possible event types. Each event type will be identified by a *semanticId* and will feature a specialized payload.

Group	Direction ¹⁰	Motivation / conditions
Structural changes of the AAS	Out	<ul style="list-style-type: none"> • CRUD¹¹ of Submodels, Assets, SubmodelElements and such
	In	<ul style="list-style-type: none"> • Detect updates on parent/ type/ <i>derivedFrom</i> AAS
Updates of Properties and dependent attribute	Out	<ul style="list-style-type: none"> • update of values of SubmodelElements • timestamped updates and time series update • explicit triggering of an update event
Operation of AAS	Out	<ul style="list-style-type: none"> • monitoring of (long-lasting) execution of <i>OperationElement</i> and updating events while execution
Monitoring, conditional, calculated events	Out	<ul style="list-style-type: none"> • e.g. when voiding some limits (e.g. stated by Qualifiers with expression semantics)
Infrastructure events	Out	<ul style="list-style-type: none"> • Booting, Shutdown, out of memory ... of software entity of respective Referable (AAS, Submodel)
Repository events	In/ Out	<ul style="list-style-type: none"> • Change of semantics of IRDIs (associated concept definition)
Security events	Out	<ul style="list-style-type: none"> • logging events • access violations, non-fitting roles & rights, denial of service, ...
Alarms & events	Out	<ul style="list-style-type: none"> • alarms and events management analog to distributed control systems (DCS)

Custom Event types

In any case, it is possible to define custom event types by using a proprietary, but worldwide unique, semanticId for this event type. Such customized events can be sent or received by the software entity of the respective Referable, based on arbitrary conditions, triggers or behavior. However, the general format of the event messages needs to comply this specification, but the payload might be completely customized.

Event scopes

Events can be stated with an *observableReference* to the *Referables* of AAS, *Submodels*, and *SubmodelElements*. These *Referables* are defining the scope of the events, which are to be received or sent.

Event attached to ...	Scope
AssetAdministrationShell	This event is monitoring/ representing all logical elements of an Administration Shell, such as <i>AssetAdministrationShell</i> , <i>AssetInformation</i> , <i>Submodels</i> .
Submodel	This event is monitoring/ representing all logical elements of the respective <i>Submodel</i> and all logical dependents.

¹⁰ see below

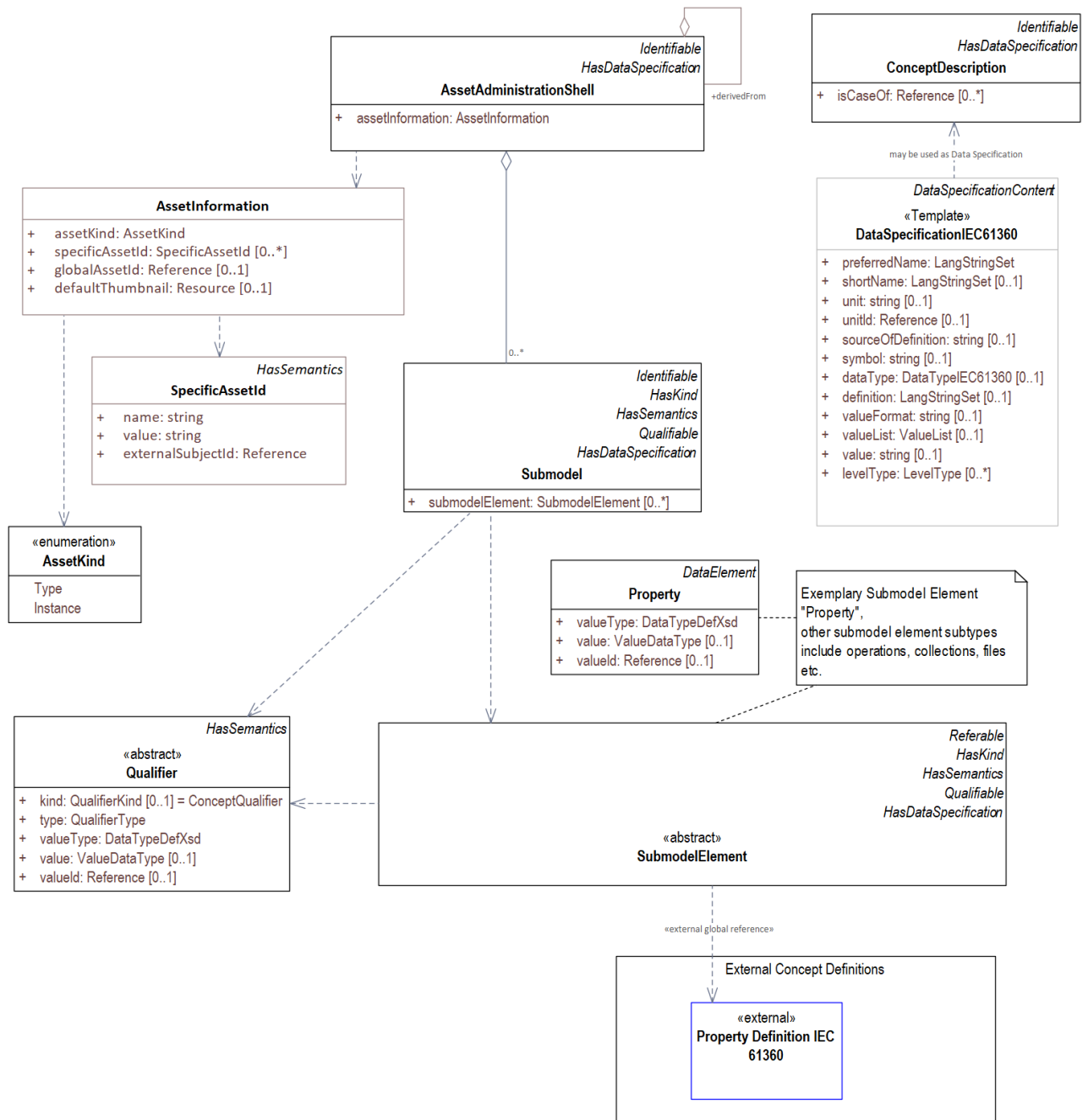
¹¹ Create, Retrieve, Update, Delete

SubmodelElementList and SubmodelElementCollection and Entity	This event is monitoring/ representing all logical elements of the respective <i>SubmodelElementCollection</i> , <i>SubmodelElementList</i> or <i>Entity</i> and all logical dependents (value or statement resp.).
SubmodelElement (others)	This event is monitoring/ representing a single atomic <i>SubmodelElement</i> , e.g. a data element which might include the contents of a <i>Blob</i> or <i>File</i> .

5.6 Overview Metamodel of the Administration Shell

In this clause an overview of the main concepts of the Asset Administration Shell (AAS) metamodel is presented, see Figure 12.

Figure 12 Overview Metamodel of the Asset Administration Shell



An AAS represents exactly one asset (*AssetAdministrationShell/assetInformation*). Asset types and asset instances are distinguished by setting the attribute “*AssetInformation/assetkind*”. For details see Clause 5.7.4.

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

In case of an AAS of an asset instance, a reference to the AAS representing the corresponding asset type or another asset instance it was derived from may be added (*AssetAdministrationShell/derivedFrom*). The same holds true 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 external identifiers are defined as specific asset IDs, each characterized by a user defined name, a value and the user domain (tenant, subject in Attribute Based Access Control) (*AssetInformation/specificAssetId*). For details see Clause 5.7.4. Additionally, a global asset identifier should be assigned to the asset (*AssetInformation/globalAssetId*) in production and operation phase.

AASs, submodels and concept descriptions need to be globally uniquely identifiable (*Identifiable*). Other elements like for example properties just need to be referable within the model and thus only need a local identifier (*idShort* from *Referable*). For details on identification see Clause 5.4. For details on *Identifiable* and *Referable* see Clause 5.7.2.2 and Clause 5.7.2.3.

Submodels consist of a set of submodel elements. Submodel elements may be qualified by a so-called *Qualifier*. For details see Clause 5.7.2.7 and Clause 5.7.2.8.

There are different subtypes of submodel elements like properties, operations, lists etc. For details see Clause 5.7.7. 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. For details on properties see Clause 5.7.7.11.

Every submodel element needs a semantic definition (*semanticId* in *HasSemantics*) to have a well-defined meaning. The submodel element might either refer directly to a corresponding semantic definition provided by an external reference (e.g. to an ECLASS or IEC CDD property definition) or it may indirectly reference a concept description (*ConceptDescription*). For matching strategies see Clause 5.4.6, for details see Clause 5.7.2.6.

A concept description may be derived from another property definition of an external standard or another concept description (*ConceptDescription/isCaseOf*). *isCaseOf* is a more formal definition of *sourceOfDefinition* that is just text.

Note: In this case most of the attributes are redundant because these are defined in the external standard. It is about usability to add attributes for information like *preferredName*, *unit* etc. Consistency w.r.t. the referenced submodel element definitions should be ensured by corresponding tooling.

In case a concept description is not just a copy or refinement of an external standard, the provider of the AAS using this concept description shall be aware that no interoperability with other AAS can be ensured.

Data Specification Templates can be used (*DataSpecification*) to define a named set of additional attributes (besides those predefined by the metamodel) for an element. For the concept description of properties typically the Data Specification Template following IEC 61360 is used providing for example an attribute “preferredName”. For denoting recommended Data Specification Templates to be used the <<template>>-dependency is used. For details see Clause 5.7.2.9.

Data Specification Templates like the template for IEC 61360 property definitions (*DataSpecificationIEC61360* and *DataSpecificationPhysicalUnit*) are explicitly predefined and recommended to be used by the Plattform Industrie 4.0. For details see Clause 6. If proprietary templates are used, interoperability with other AAS cannot be ensured.

Besides submodel elements including properties and concept descriptions also other identifiable elements may use additional templates (*HasDataSpecification*). Data Specification Templates are selected at design time. For details see Clause 5.7.2.9.

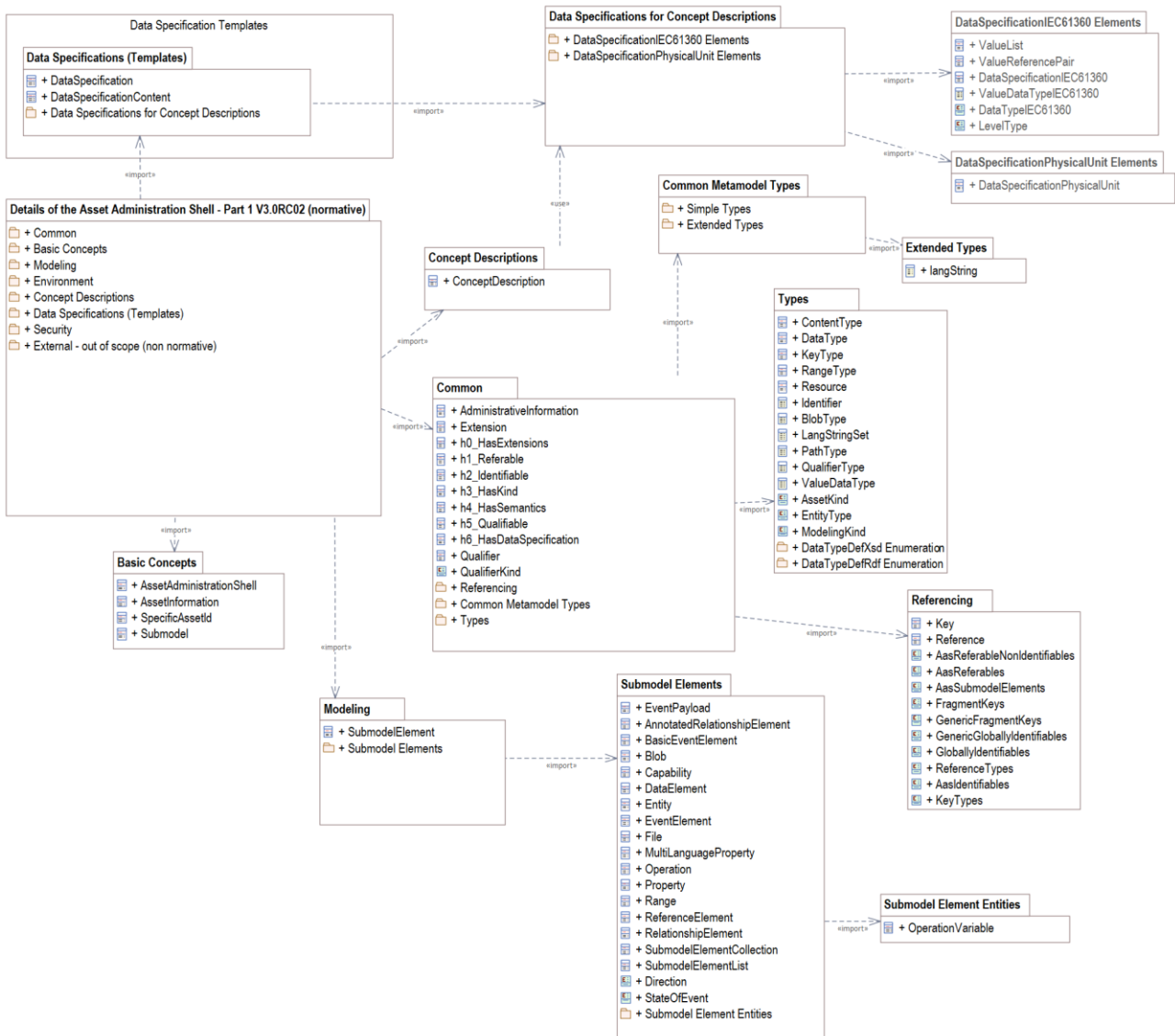
Submodel elements and the submodels themselves may have additional qualifiers (*Qualifiable*). Per *Qualifiable* there might be more than one qualifier. For details see Clause 5.7.2.7.

For every AAS, security aspects need to be considered. In this document 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. For details see Clause 7.

Figure 13 gives a complete picture of all elements defined in the metamodel excluding security. Information on the Security part is found in Clause 7.

Note: The abstract classes are numbered h0_, h1_ etc. (e.g. h1_Referable) but Aliases are defined for them without this prefix. The reason for this naming is that in the tooling used for UML modelling (Enterprise Architect) no order for inherited classes can be defined, they are ordered in an alphabetical way. For some serializations the order is important (for example for XML).

Figure 13 Metamodel package overview



5.7 Metamodel Specification Details: Designators (normative)

5.7.1 Introduction

In this clause the classes of the metamodel are specified in detail. In Annex B the template used to describe the classes and relationships is explained. In Annex D some of the diagrams are shown together with all its inherited attributes to give a complete overview.

For understanding the specifications, it is crucial to understand the common attributes first (Clause 5.7.2). They are reused throughout the specifications of the other classes (“inherits from”) and define important concepts like identifiable, qualifiable etc. They are abstract, i.e. there is no object instance of such classes.

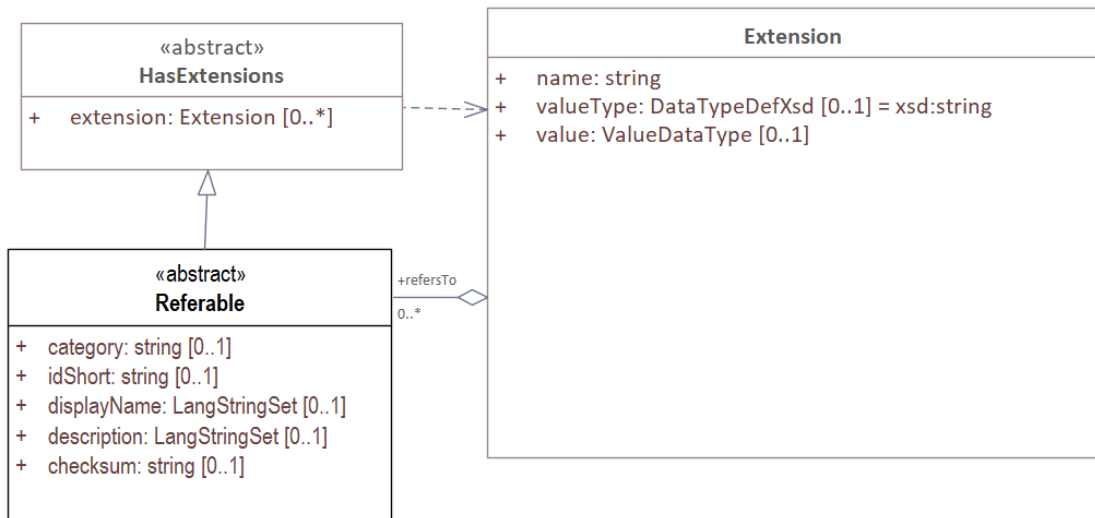
Another important concept is the concept of referencing and how a reference is represented in the UML diagrams and the tables. This is explained in Clause 5.7.9 and Annex D ii.

Constraints that are no invariants of classes are specified in Clause 5.7.12.3.

5.7.2 Common Attributes

5.7.2.1 Extensions (HasExtensions)

Figure 14 Metamodel of HasExtensions



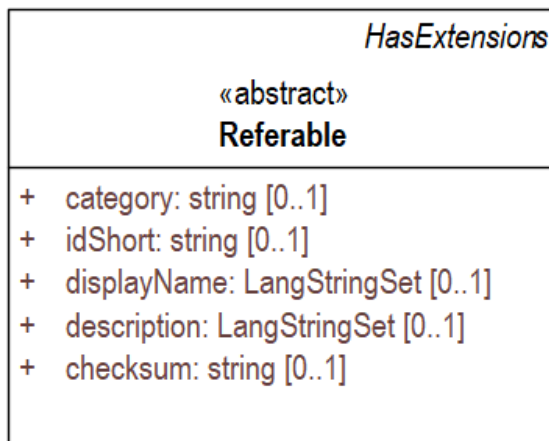
Class:	HasExtensions <<abstract>>		
Explanation:	Element that can be extended by proprietary extensions. Note: Extensions are proprietary, i.e. they do not support global interoperability.		
Inherits from:	--		
Attribute	Explanation	Type	Card.
extension	An extension of the element.	Extension	0..*

Class:	Extension		
Explanation:	Single extension of an element.		
Inherits from:	HasSemantics		
Attribute	Explanation	Type	Card.
name	Name of the extension.	string	1

Class:	Extension		
valueType	Type of the value of the extension. Default: xs:string	DataTypeDefXsd	0..1
value	Value of the extension	ValueDataType	0..1
refersTo	Reference to an element the extension refers to.	ModelReference<Referable>	0..*

5.7.2.2 Referable Attributes

Figure 15 Metamodel of Referables



The metamodel distinguishes between elements that are identifiable, referable or none of both.

Referable elements can be referenced via the *idShort*. For details on how to do referencing see Clause 5.7.9.

Not every element of the metamodel is referable. There are elements that are just attributes of a referable.

For non-identifiable referables the *idShort* shall be unique in its name space (Constraint AASd-022). A name space is defined as follows in this context: The parent element(s) an element is part of and that is either referable or identifiable is the name space of the element. Examples: A submodel is the name space for the properties contained in it. The name space of a submodel element being contained in a submodel element collection is the submodel element collection.

Class:	Referable <<abstract>>		
Explanation:	An element that is referable by its <i>idShort</i> . This ID is not globally unique. This ID is unique within the name space of the element.		
Inherits from:	HasExtensions		
Attribute	Explanation	Type	Card.
category	The category is a value that gives further meta information w.r.t. the class of the element. It affects the expected existence of attributes and the applicability of constraints. Note: The category is not identical to the semantic definition (<i>HasSemantics</i>) of an element. The category e.g. could denote that the element is a measurement	string	0..1

Class:	Referable <<abstract>>		
	value whereas the semantic definition of the element would denote that it is the measured temperature.		
idShort	<p>In case of identifiable this attribute is a short name of the element. In case of referable this ID is an identifying string of the element within its name space.</p> <p><u>Constraint AASd-027:</u> <i>idShort</i> of <i>Referables</i> shall have a maximum length of 128 characters.</p> <p>Note: In case the element is a property and the property has a semantic definition (<i>HasSemantics/semanticId</i>) conformant to IEC61360 the <i>idShort</i> is typically identical to the short name in English – if available.</p>	string	0..1
displayName	<p>Display name. Can be provided in several languages.</p> <p>If no display name is defined in the language requested by the application, then the display name is selected in the following order if available:</p> <ul style="list-style-type: none"> - the preferred name in the requested language of the concept description defining the semantics of the element - If there is a default language list defined in the application, then the corresponding preferred name in the language is chosen according to this order. - the English preferred name of the concept description defining the semantics of the element - the short name of the concept description - the idShort of the element 	LangStringSet	0..1
description	<p>Description or comments on the element.</p> <p>The description can be provided in several languages.</p> <p>If no description is defined, then the definition of the concept description that defines the semantics of the element is used.</p> <p>Additional information can be provided, e.g. if the element is qualified and which qualifier types can be expected in which context or which additional data specification templates are provided.</p>	LangStringSet	0..1
checksum	<p>Checksum to be used to determine if an Referable (including its aggregated child elements) has changed.</p> <p>The checksum is calculated by the user's tool environment. The checksum has no semantic meaning for an Asset Administration Shell model and there is no requirement for Asset Administration Shell tools to manage the checksum.</p>	string	0..1

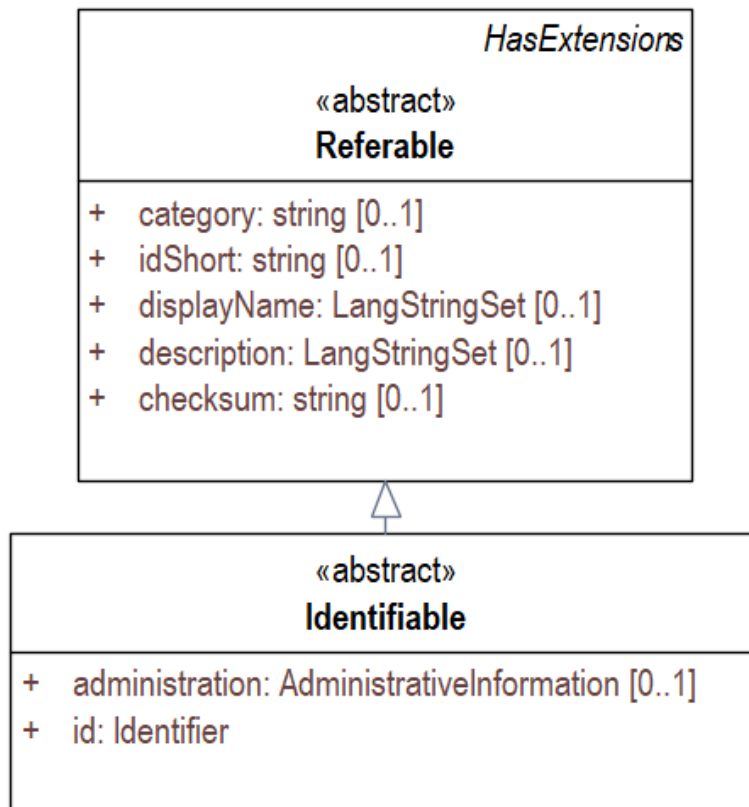
In Table 5 predefined categories are described.

Table 5 Categories for Elements with Value

Category:	Applicable to, Examples	Explanation:
CONSTANT	Property ReferenceElement	An element with category CONSTANT is an element with a value that does not change over time. In ECLASS this kind of category has the category "Coded Value".
PARAMETER	Property MultiLanguageProperty Range SubmodelElementCollection	An element with category PARAMETER is an element that is once set and then typically does not change over time. This is for example the case for configuration parameters.
VARIABLE	Property SubmodelElementList	An element with category VARIABLE is an element that is calculated during runtime, i.e. its value is a runtime value.

5.7.2.3 Identifiable Attributes

Figure 16 Metamodel of Identifiables



An identifiable element is a referable with a globally unique identifier (*Identifier*). To reference an identifiable only the global ID (*Identifiable/id*) shall be used because the *idShort* is not unique for an identifiable. Identifiables may have administrative information like version etc.

Referable elements not being identifiable can be referenced, but for doing so the context of the element is needed. A referable not being identifiable has a short identifier (*idShort*) that is unique just in its context, its name space.

Information about identification can be found in Clause 5.4. In Clause 5.4.4 constraints and recommendation on when to use which type of identifier can be found.

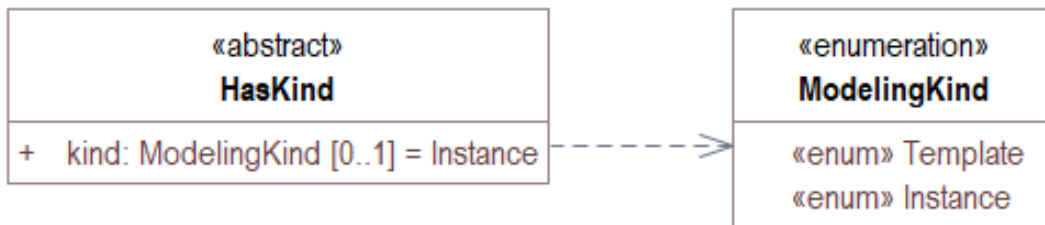
Examples for identifiers can be found in Clause 5.4.3 Identifiers for Assets and Administration Shells.

See Clause 5.4.4 for information which identifier types are supported.

Class:	Identifiable <<abstract>>		
Explanation:	An element that has a globally unique identifier.		
Inherits from:	Referable		
Attribute	Explanation	Type	Card.
administration	Administrative information of an identifiable element.	AdministrativeInformation	0..1
	Note: Some of the administrative information like the version number might need to be part of the identification.		
id	The globally unique identification of the element.	Identifier	1

5.7.2.4 Template or Instance of Model Element Attributes (HasKind)

Figure 17 Metamodel of HasKind



Class:	HasKind		
Explanation:	An element with a kind is an element that can either represent a template or an instance. Default for an element is that it is representing an instance.		
Inherits from:	--		
Attribute	Explanation	Type	Card.
kind	Kind of the element: either type or instance. Default Value = <i>Instance</i>	ModelingKind	0..1

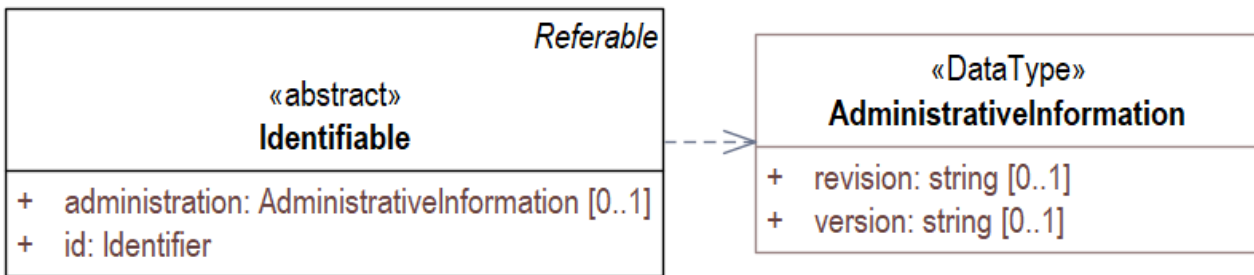
The kind enumeration is used to denote whether an element is of kind Template or Instance.

Enumeration:	ModelingKind
Explanation:	Enumeration for denoting whether an element is a template or an instance.

Enumeration:	ModelingKind
Set of:	--
Literal	Explanation
Template	Software element which specifies the common attributes shared by all instances of the template. [SOURCE: IEC TR 62390:2005-01, 3.1.25] modified
Instance	Concrete, clearly identifiable component of a certain template. Note: It becomes an individual entity of a template, for example a device model, by defining specific property values. Note: In an object-oriented view, an instance denotes an object of a template (class). [SOURCE: IEC 62890:2016, 3.1.16 65/617/CDV] modified

5.7.2.5 Administrative Information Attributes

Figure 18 Metamodel of Administrative Information



Every *Identifiable* may have administrative information. Administrative information includes for example

- Information about the version of the element
- Information about who created or who made the last change to the element
- Information about the languages available in case the element contains text, for translating purposes also the master or default language may be defined

In the first versions of the AAS metamodel only version information is defined for administrative information. In later versions additional attributes may be added.

Version corresponds in principle to the *version_identifier* according to IEC 62832 but is not used for concept identifiers only (IEC TS 62832-1) but for all identifiable elements. Version and revision together correspond to the version number according to IEC 62832.

AdministrativeInformation allows the usage of templates (*HasDataSpecification*) but there are no predefined templates in this version of the metamodel for administrative information.

Note: Two submodels with the same semanticId but different administrative information (this especially includes different versions), shall have different IDs (*SubmodelId*). The idShort or numeric identifier typically do not change, i.e. they are identical. The same holds for other identifiables (*IdentifiableId*).

Note: Since submodels with different version shall have different identifiers it is possible that an AAS has two submodels with the same *semanticId* but different versions, i.e. different administrative meta-information.

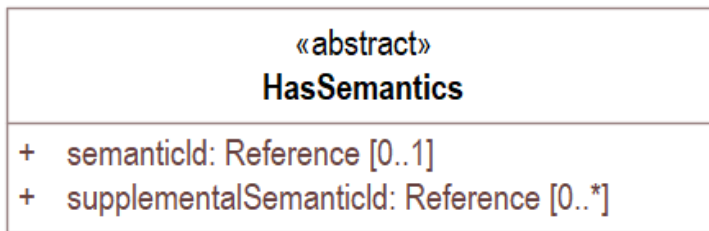
Note: Some of the administrative information like the version number might need to be part of the identification. This is similar to the handling of identifiers for concept descriptions using IRDIs. In ECLASS the IRDI 0173-1#02-AO677#002 contains the version information #002.

Note: The process of versioning of or adding other administrative information to elements is done by external version or configuration management software and not by the AAS itself.

Class:	AdministrativeInformation <<DataType>>		
Explanation:	Administrative metainformation for an element like version information. <u>Constraint AASd-005:</u> If <i>AdministrativeInformation/version</i> is not specified than also <i>AdministrativeInformation/revision</i> shall be unspecified. This means, a revision requires a version. if there is no version there is no revision neither. Revision is optional.		
Inherits from:	HasDataSpecification		
Attribute	Explanation	Type	Card.
version	Version of the element.	string	0..1
revision	Revision of the element.	string	0..1

5.7.2.6 Semantic References Attributes (HasSemantics)

Figure 19 Metamodel of Semantic References (HasSemantics)

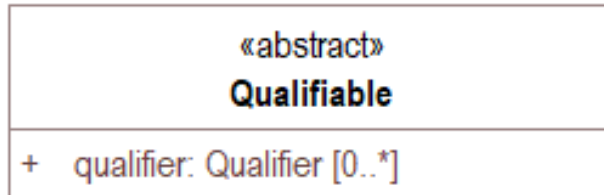


Class:	HasSemantics <<abstract>>		
Explanation:	Element that can have a semantic definition plus some supplemental semantic definitions. <u>Constraint AASd-118:</u> If there is a supplemental semantic ID (<i>HasSemantics/supplementalSemanticId</i>) defined then there shall be also a main semantic ID (<i>HasSemantics/semanticId</i>).		
Inherits from:	--		
Attribute	Explanation	Type	Card.
semanticId	Identifier of the semantic definition of the element. It is called semantic ID or also main semantic ID of the element. It is recommended to use a global reference.	Reference	0..1
supplementalSemanticId	Identifier of a supplemental semantic definition of the element. It is called supplemental semantic ID of the element.	Reference	0..*

Class:	HasSemantics <<abstract>>		
	It is recommended to use a global reference.		

5.7.2.7 Qualifiable Attributes

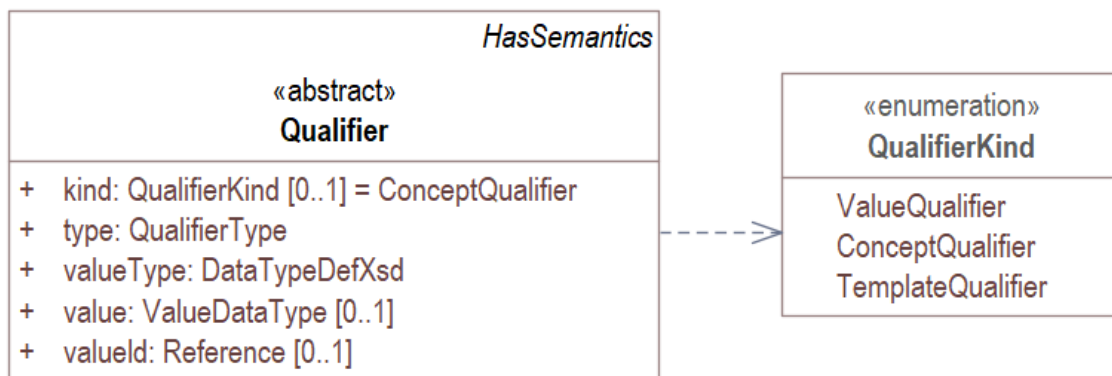
Figure 20 Metamodel of Qualifiables



Class:	Qualifiable <<abstract>>		
Explanation:	The value of a qualifiable element may be further qualified by one or more qualifiers.		
Inherits from:	--		
Attribute	Explanation	Type	Card.
qualifier	Additional qualification of a qualifiable element.	Qualifier	0..*

5.7.2.8 Qualifier Attributes

Figure 21 Metamodel of Qualifiers



For qualifiable elements, additional qualifiers may be defined. Besides other qualifiers, a level qualifier defining the level type minimal value, maximum value, typical value and nominal value can be found in IEC 62569-1. Additional qualifier types are defined in DIN SPEC 92000 like for example expressions semantics and expression logic.

Class:	Qualifier
Explanation:	A qualifier is a type-value-pair that makes additional statements w.r.t. the value of the element.

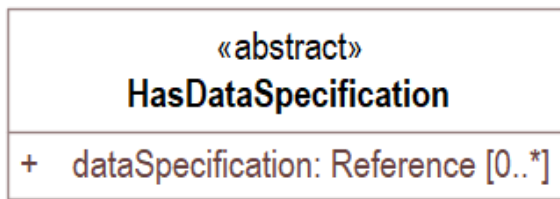
Class:	Qualifier		
	<p><u>Constraint AASd-006</u>: If both, the <i>value</i> and the <i>valueId</i> of a <i>Qualifier</i> are present then the value needs to be identical to the value of the referenced coded value in <i>Qualifier/valueId</i>.</p> <p><u>Constraint AASd-020</u>: The value of <i>Qualifier/value</i> shall be consistent to the data type as defined in <i>Qualifier/valueType</i>.</p>		
Inherits from:	HasSemantics		
Attribute	Explanation	Type	Card.
kind	The qualifier kind describes the kind of the qualifier that is applied to the element. Default: ConceptQualifier	QualifierKind	0..1
type	The qualifier <i>type</i> describes the type of the qualifier that is applied to the element.	QualifierType	1
valueType	Data type of the qualifier value.	DataTypeDefXsd	1
value	The qualifier value is the value of the qualifier.	ValueDataType	0..1
valueId	Reference to the global unique ID of a coded value. It is recommended to use a global reference.	Reference	0..1

It is recommended to add a *semanticId* for a *Qualifier*.

Enumeration:	QualifierKind
Explanation:	Enumeration for kinds of qualifiers.
Set of:	--
Literal	Explanation
ValueQualifier	qualifies the value of the element and can change during run-time Value qualifiers are only applicable to elements with kind=„Instance”
ConceptQualifier	qualifies the semantic definition the element is referring to (<i>HasSemantics/semanticId</i>)
TemplateQualifier	qualifies the elements within a specific submodel on concept level. Template qualifiers are only applicable to elements with kind=„Template”

5.7.2.9 Used Templates for Data Specification Attributes (HasDataSpecification)

Figure 22 Metamodel of HasDataSpecification

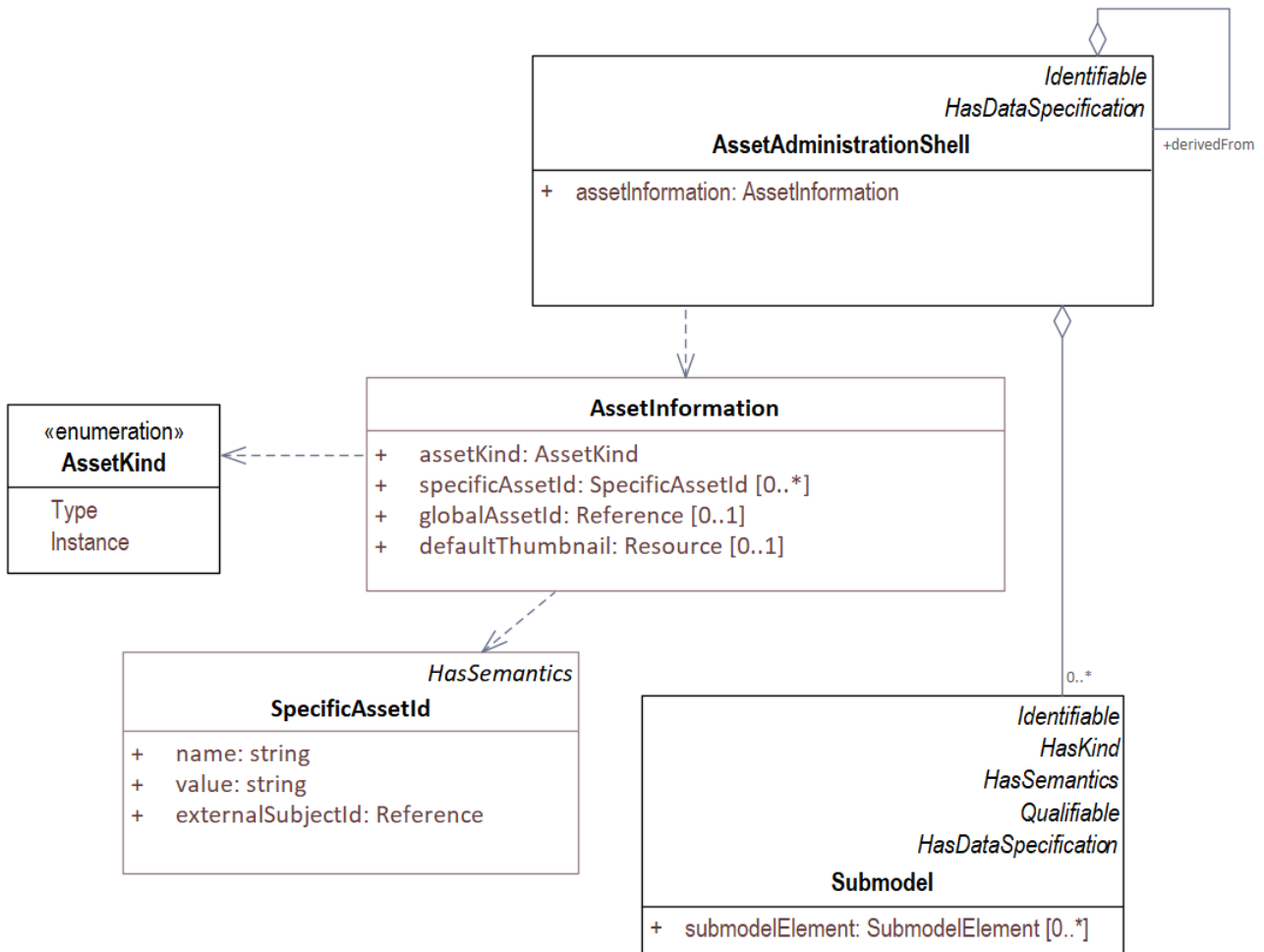


Class:	HasDataSpecification <<abstract>>		
Explanation:	Element that can be extended by using data specification templates. A data specification template defines a named set of additional attributes an element may or shall have. The data specifications used are explicitly specified with their global ID.		
Inherits from:	--		
Attribute	Explanation	Type	Card.
dataSpecification	Global reference to the data specification template used by the element. This is a global reference.	Reference	0..*

For more details on data specifications see Clause 6.

5.7.3 Asset Administration Shell Attributes

Figure 23 Metamodel AssetAdministrationShell



An Administration Shell is uniquely identifiable since it inherits from *Identifiable*.

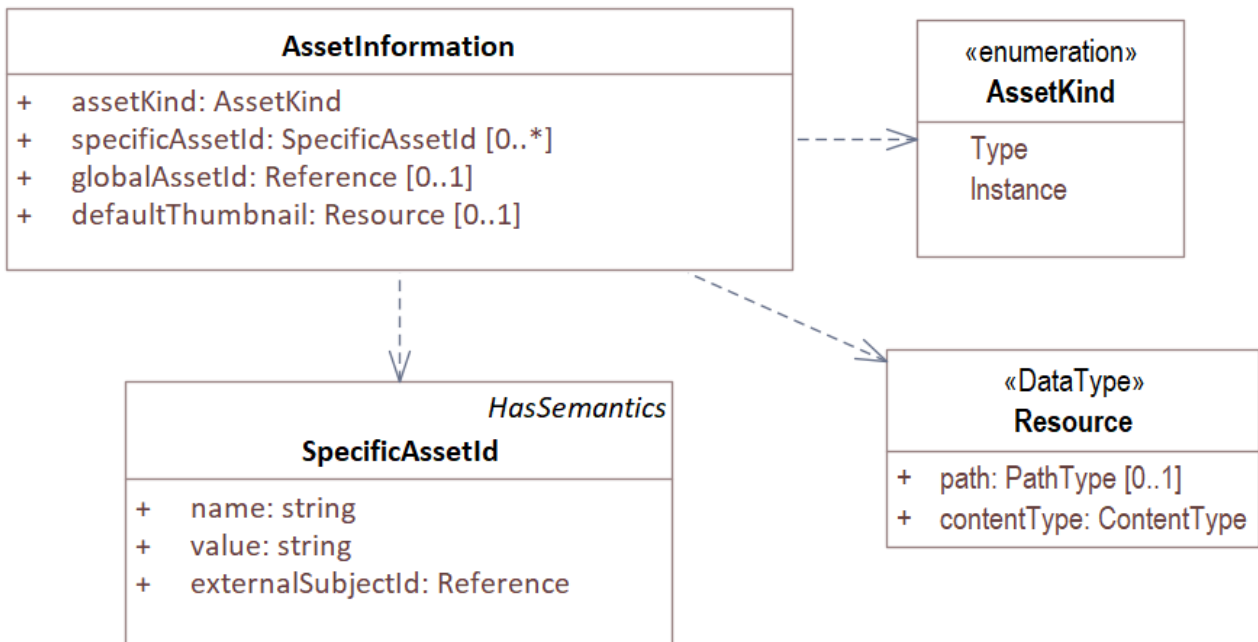
The *derivedFrom* attribute is used to establish a relationship between two Asset Administration Shells that are derived from each other. For more detailed information on the *derivedFrom* concept see Clause 5.2 Types and Instances.

Class:	AssetAdministrationShell		
Explanation:	An Asset Administration Shell.		
Inherits from:	Identifiable; HasDataSpecification		
Attribute	Explanation	Type	Card.
derivedFrom	The reference to the AAS the AAS was derived from.	ModelReference<AssetAdministrationShell>	0..1
assetInformation	Meta information about the asset the AAS is representing.	AssetInformation	1
submodel	Reference to a submodel of the AAS.	ModelReference<Submodel>	0..*

Class:	AssetAdministrationShell		
	<p>A submodel is a description of an aspect of the asset the AAS is representing.</p> <p>The asset of an AAS is typically described by one or more submodels.</p> <p>Temporarily no submodel might be assigned to the AAS.</p>		

5.7.4 Asset Information Attributes

Figure 24 Metamodel of Asset Information



Class:	AssetInformation		
Explanation:	<p>In <i>AssetInformation</i> identifying meta data of the asset that is represented by an AAS is defined.</p> <p>The asset may either represent an asset type or an asset instance.</p> <p>The asset has a globally unique identifier plus – if needed – additional domain specific (proprietary) identifiers. However, to support the corner case of very first phase of lifecycle where a stabilised/constant global asset identifier does not already exist, the corresponding attribute “globalAssetId” is optional.</p>		
Inherits from:	--		
Attribute	Explanation	Type	Card.
assetKind	Denotes whether the <i>Asset</i> is of kind “Type” or “Instance”.	AssetKind	1
globalAssetId	Global identifier of the asset the AAS is representing.	Reference	0..1

Class:	AssetInformation		
	<p>This attribute is required as soon as the AAS is exchanged via partners in the life cycle of the asset. In a first phase of the life cycle the asset might not yet have a global ID but already an internal identifier. The internal identifier would be modelled via “<i>specificAssetId</i>”.</p> <p>This is a global reference.</p>		
specificAssetId	Additional domain specific, typically proprietary identifier for the asset like e.g. serial number etc.	SpecificAssetId	0..*
defaultThumbnail	Thumbnail of the asset represented by the Asset Administration Shell. Used as default.	Resource	0..1

Note: Besides this asset information there still might be an identification submodel with further information. Specific asset IDs mainly serve the purpose for supporting discovery of AASs for an asset.
 Note: Keys for specificAssetIds do not need to be unique.
 Note: SemanticIds for the single specificAssetIds do not need to be unique.

Class:	Resource <<DataType>>		
Explanation:	Resource represents an address to a file (a locator). The value is an URI that can represent an absolute or relative path.		
Inherits from:	--		
Attribute	Explanation	Type	Card.
path	<p>Path and name of the resource (with file extension).</p> <p>The path can be absolute or relative.</p>	PathType	1
contentType	<p>Content type of the content of the file.</p> <p>The content type states which file extensions the file can have.</p>	ContentType	0..1

Enumeration:	AssetKind
Explanation:	Enumeration for denoting whether an asset is a type asset or an instance asset.
Set of:	--
Literal	Explanation
Type	<p>hardware or software element which specifies the common attributes shared by all instances of the type</p> <p>[SOURCE: IEC TR 62390:2005-01, 3.1.25]</p>
Instance	concrete, clearly identifiable component of a certain type

Enumeration:	AssetKind
	<p>Note 1: It becomes an individual entity of a type, for example a device, by defining specific property values.</p> <p>Note 2: In an object-oriented view, an instance denotes an object of a class (of a type).</p>
	[SOURCE: IEC 62890:2016, 3.1.16] 65/617/CDV

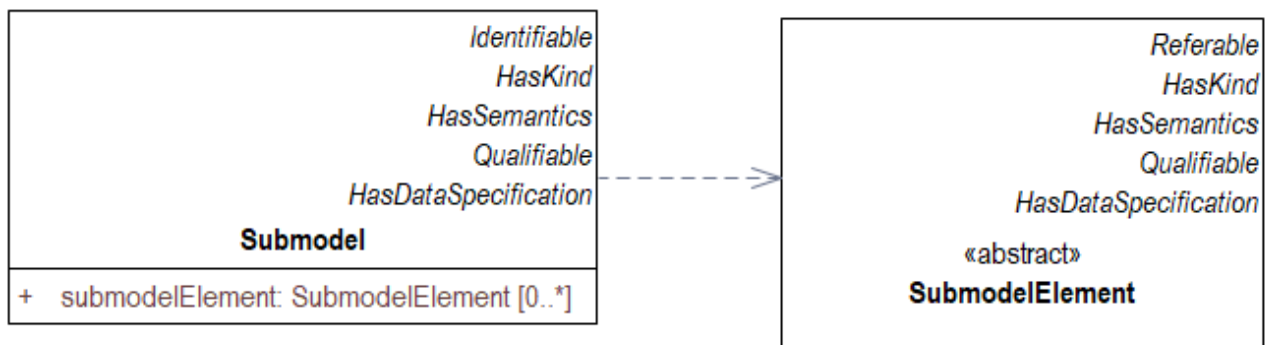
For more information on types and instances see Clause 5.2.

Class:	SpecificAssetId		
Explanation:	A specific asset ID describes a generic supplementary identifying attribute of the asset. The specific asset ID is not necessarily globally unique.		
Inherits from:	HasSemantics		
Attribute	Explanation	Type	Card.
name	Name of the identifier	string	1
value	The value of the specific asset identifier with the corresponding name.	string	1
externalSubjectId	<p>The (external) subject the specific asset ID belongs to or has meaning to.</p> <p style="background-color: #D9E1F2;">This is a global reference.</p>	Reference	1

For more information on the concept of subject see Clause 7 on Attribute Based Access Control (ABAC).

5.7.5 Submodel Attributes

Figure 25 Metamodel of Submodel



Class:	Submodel		
Explanation:	<p>A submodel defines a specific aspect of the asset represented by the AAS.</p> <p>A submodel is used to structure the digital 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.</p>		
Inherits from:	Identifiable; HasKind; HasSemantics; Qualifiable; HasDataSpecification		
Attribute	Explanation	Type	Card.
submodelElement	A submodel consists of zero or more submodel elements.	SubmodelElement	0..*

A submodel is handled like a *SubmodelElementCollection*. The difference is that it is identifiable and a predefined unit of information within the AAS.

It is recommended to add a *semanticId* for a submodel.

A submodel can be qualified (*Qualifiable*).

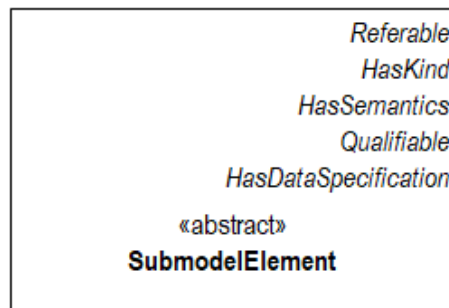
Submodel elements are qualifiable elements, i.e. one or more qualifiers may be defined for each of them.

Submodels and submodel elements may also have data specification templates defined for them. So far, no standardized data specification templates for submodels and submodel elements are defined.

In case the submodel is of *kind=Template* (modelling kind as inherited by *HasKind*) then the submodel elements within the submodel are presenting submodel element templates. In case the submodel is of *kind=Instance* then its submodel elements represent submodel element instances.

5.7.6 Submodel Element Attributes

Figure 26 Metamodel of Submodel Element



Submodel element are qualifiable elements, i.e. one or more qualifiers may be defined for each of them.

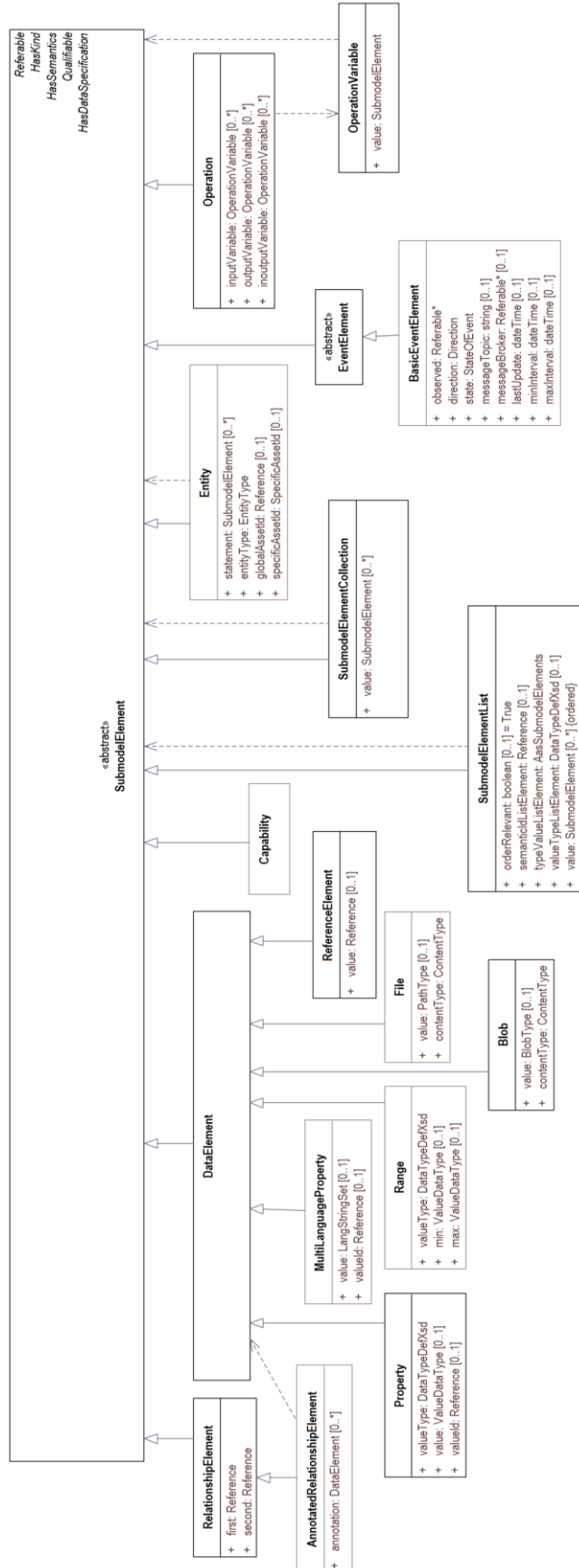
Submodel elements may also have data specification templates defined for them. A template might for example be defined to mirror some of the attributes like *preferredName* and *unit* of a property concept definition if there is no corresponding concept description available. Otherwise there only is the property definition referenced by *semanticId* available for the property: the lookup of the attributes has to be realized online in a different way and is not available offline.

In case the submodel is of *kind=Template* then the submodel elements within the submodel are presenting submodel element types. In case the submodel is of *kind=Instance* then its submodel elements represent submodel element instances.

Class:	SubmodelElement <<abstract>>		
Explanation:	<p>A submodel element is an element suitable for the description and differentiation of assets.</p> <p>It is recommended to add a <i>semanticId</i> to a <i>SubmodelElement</i>.</p>		
Inherits from:	Referable; HasKind; HasSemantics; Qualifiable; HasDataSpecification		
Attribute	Explanation	Type	Card.

5.7.7 Overview of Submodel Element Types

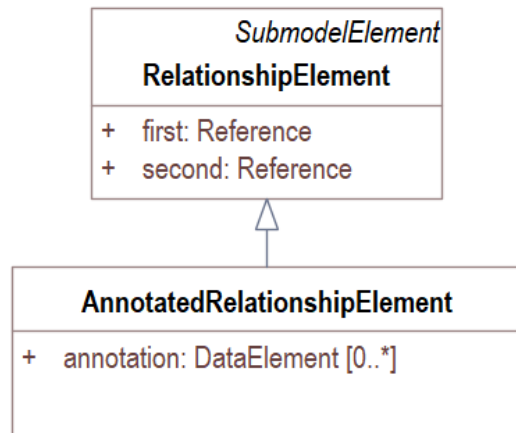
Figure 27 Metamodel Overview for Submodel Element Subtypes



Submodel elements include data properties as well as operations, events and other elements needed to describe a model for an asset (see Figure 27).

5.7.7.1 Annotated Relationship Element Attributes

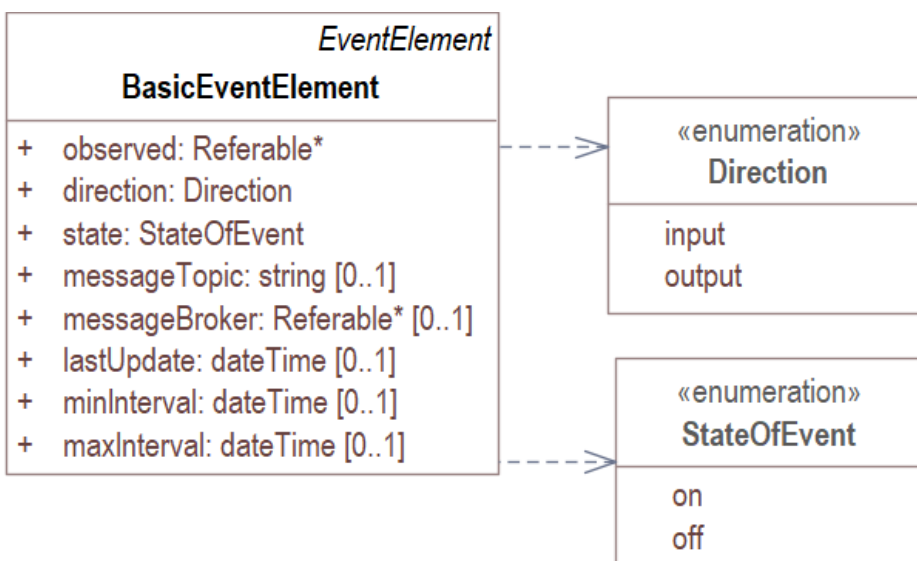
Figure 28 Metamodel of Annotated Relationship Elements



Class:	AnnotatedRelationshipElement		
Explanation:	An annotated relationship element is a relationship element that can be annotated with additional data elements.		
Inherits from:	RelationshipElement		
Attribute	Explanation	Type	Card.
annotation	A data element that represents an annotation that holds for the relationship between the two elements.	DataElement	0..*

5.7.7.2 Basic Event Element Attributes

Figure 29 Metamodel of Basic Event Element



Class:	BasicEventElement		
Explanation:	A basic event element.		
Inherits from:	EventElement		
Attribute	Explanation	Type	Card.
observed	Reference to the Referable, which defines the scope of the event. Can be AAS, Submodel or SubmodelElement. Reference to a referable, e.g. a data element or a submodel, that is being observed.	ModelReference<Referable>	1
Direction	Direction of event. Can be { Input, Output }.	Direction	1
State	State of event. Can be { On, Off }.	StateOfEvent	1
messageTopic	Information for the outer message infrastructure for scheduling the event to the respective communication channel.	string	0..1
messageBroker	Information, which outer message infrastructure shall handle messages for the <i>EventElement</i> . Refers to a <i>Submodel</i> , <i>SubmodelElementList</i> , <i>SubmodelElementCollection</i> or <i>Entity</i> , which contains <i>DataElements</i> describing the proprietary specification for the message broker. Note: for different message infrastructure, e.g. OPC UA or MQTT or AMQP, this proprietary specification could be standardized by having respective Submodels.	ModelReference<Referable>	0..1
lastUpdate	Timestamp in UTC, when the last event was received (input direction) or sent (output direction).	dateTime	0..1
minInterval	For input direction, reports on the maximum frequency, the software entity behind the respective Referable can handle input events. For output events, specifies the maximum frequency of outputting this event to an outer infrastructure. Might be not specified, that is, there is no minimum interval.	dateTime	0..1
maxInterval	For input direction: not applicable.	dateTime	0..1

Class:	BasicEventElement		
	For output direction: maximum interval in time, the respective Referable shall send an update of the status of the event, even if no other trigger condition for the event was not met. Might be not specified, that is, there is no maximum interval.		

Enumeration:	Direction
Explanation:	Direction
Set of:	--
Literal	Explanation
input	Input direction.
output	Output direction.

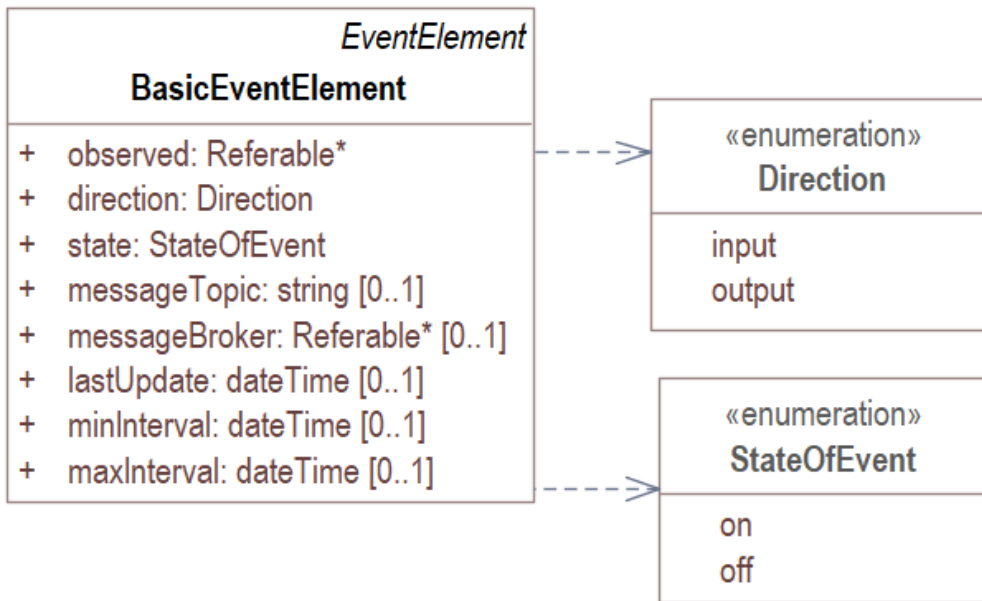
Enumeration:	StateOfEvent
Explanation:	State of an event
Set of:	--
Literal	Explanation
on	Event is on.
off	Event is off.

Events sent or received by AAS always comply to a general format. Exception: events exchanged in the course of an Industrie 4.0 interaction pattern.

In the following the payload of such an event is specified.

Note: the payload is not part of the information model as exchanged via the AASX package format but used in reactive Asset Administration Shells.

Figure 30 Metamodel Event Payload



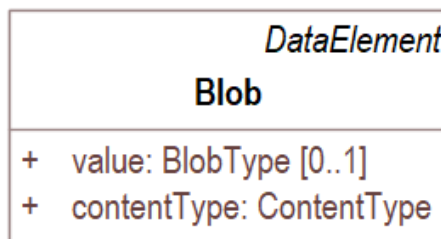
Class:	EventPayload		
Explanation:	Defines the necessary information of an event instance sent out or received.		
Inherits from:	-		
Attribute	Explanation	Type	Card.
source	Reference to the source event element, including identification of AAS, Submodel, SubmodelElements.	ModelReference<Referable>	1
sourceSemanticId	semanticId of the source event element, if available It is recommended to use a global reference.	Reference	0..1
observableReference	Reference to the referable, which defines the scope of the event. Can be AssetAdministrationShell, Submodel or SubmodelElement.	ModelReference<Referable>	1
observableSemanticId	semanticId of the referable which defines the scope of the event, if available. It is recommended to use a global reference.	Reference	0..1

Class:	EventPayload		
topic	Information for the outer message infrastructure for scheduling the event to the respective communication channel.	string	0..1
subjectId	Subject, who/which initiated the creation. This is a global reference.	Reference	0..1
timestamp	Timestamp in UTC, when this event was triggered.	dateTimeStamp	1
payload	Event specific payload.	string	0..1

For more information on the concept of subject see Clause 7 on Attribute Based Access Control (ABAC).

5.7.7.3 Blob Attributes

Figure 31 Metamodel of Blobs

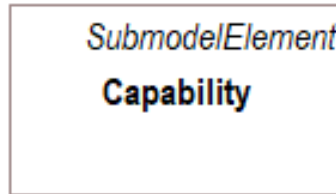


For information on content type see Clause 5.7.7.8 on submodel element “File”.

Class:	Blob		
Explanation:	A BLOB is a data element that represents a file that is contained with its source code in the value attribute.		
Inherits from:	DataElement		
Attribute	Explanation	Type	Card.
value	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.	BlobType	0..1
contentType	Content type of the content of the BLOB. The content type (MIME type) states which file extensions the file can have. Valid values are content types like e.g. “application/json”, “application/xls”, “image/jpg” The allowed values are defined as in RFC2046.	ContentType	1

5.7.7.4 Capability Attributes

Figure 32 Metamodel of Capabilities



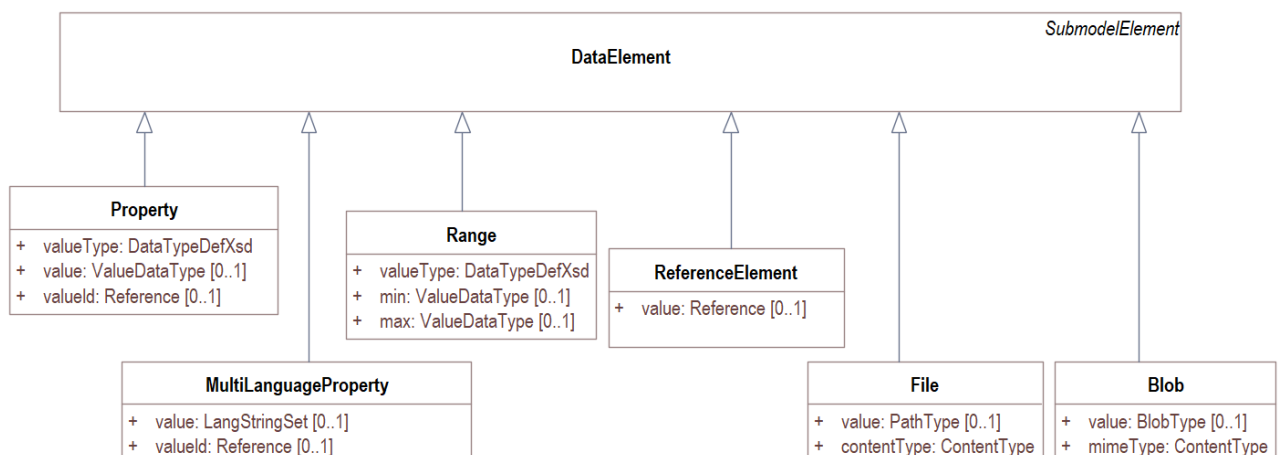
Class:	Capability		
Explanation:	A capability is the implementation-independent description of the potential of an asset to achieve a certain effect in the physical or virtual world.		
Inherits from:	SubmodelElement		
Attribute	Explanation	Type	Card.

Note: The *semanticId* of a capability is typically an ontology. Thus, reasoning on capabilities is enabled.

For information and examples how to apply the concept of capability and how to map it to one or more skills implementing the capability please refer to [36]. The mapping is done via a relationship element with the corresponding semantics. A skill is typically a property or an operation. In more complex cases the mapping can also be a collection or a complete submodel.

5.7.7.5 Data Element and Overview of Data Element Types

Figure 33 Metamodel of Data Elements



A data element is a submodel element that is not further composed out of other submodel elements.

A data element is a submodel element that has a value or a predefined number of values like range data elements.

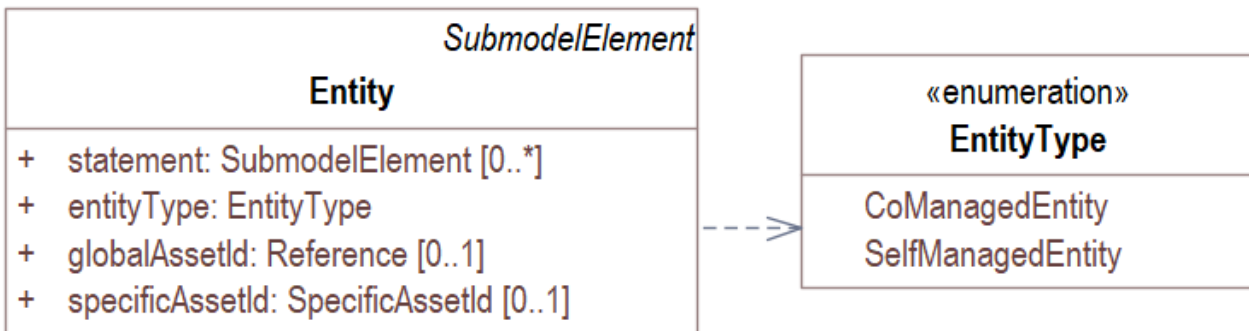
A controlled value is a value whose meaning is given in an external source (see “ISO/TS 29002-10:2009(E)”).

The type of value differs for different subtypes of data elements. Data Elements include properties and file handling and reference elements, see Figure 33.

Class:	DataElement <<abstract>>		
Explanation:	<p>A data element is a submodel element that is not further composed out of other submodel elements.</p> <p>A data element is a submodel element that has a value. The type of value differs for different subtypes of data elements.</p> <p><u>Constraint AASd-090</u>: For data elements <i>category</i> (inherited by <i>Referable</i>) shall be one of the following values: CONSTANT, PARAMETER or VARIABLE. Default: VARIABLE</p>		
Inherits from:	SubmodelElement		
Attribute	Explanation	Type	Card.

5.7.7.6 Entity Attributes

Figure 34 Metamodel of Entities

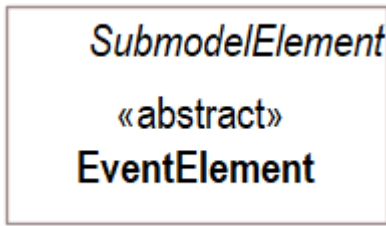


Class:	Entity		
Explanation:	<p>An entity is a submodel element that is used to model entities.</p> <p><u>Constraint AASd-014:</u> Either the attribute <i>globalAssetId</i> or <i>specificAssetId</i> of an <i>Entity</i> must be set if <i>Entity/entityType</i> is set to “<i>SelfManagedEntity</i>”. They are not existing otherwise.</p>		
Inherits from:	SubmodelElement		
Attribute	Explanation	Type	Card.
statement	Describes statements applicable to the entity by a set of submodel elements, typically with a qualified value.	SubmodelElement	0..*
entityType	Describes whether the entity is a co-managed entity or a self-managed entity.	EntityType	1
globalAssetId	<p>Global identifier of the asset the entity is representing.</p> <p>This is a global reference.</p>	Reference	0..1
specificAssetId	Reference to a specific asset ID representing a supplementary identifier of the asset represented by the Asset Administration Shell.	SpecificAssetId	0..1

Enumeration:	EntityType
Explanation:	Enumeration for denoting whether an entity is a self-managed entity or a co-managed entity.
Set of:	--
Literal	Explanation
CoManagedEntity	For co-managed entities there is no separate AAS. Co-managed entities need to be part of a self-managed entity.
SelfManagedEntity	<p>Self-Managed Entities have their own AAS but can be part of the bill of material of a composite self-managed entity.</p> <p>The asset of an I4.0 Component is a self-managed entity per definition.</p>

5.7.7.7 Event Attributes

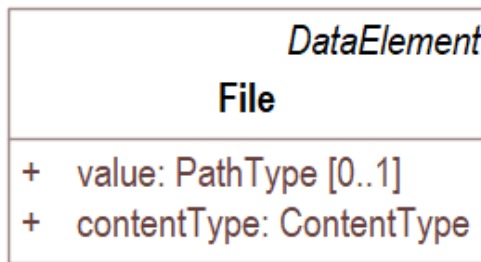
Figure 35 Metamodel of Events



Class:	EventElement <<abstract>>		
Explanation:	An event element.		
Inherits from:	SubmodelElement		
Attribute	Explanation	Type	Card.

5.7.7.8 File Attributes

Figure 36 Metamodel of File Submodel Element



A media type (also MIME type and content type) [...] is a two-part identifier for file formats and format contents transmitted on the Internet. The Internet Assigned Numbers Authority (IANA) is the official authority for the standardization and publication of these classifications. Media types were originally defined in Request for Comments 2045 in November 1996 as a part of MIME (Multipurpose Internet Mail Extensions) specification, for denoting type of email message content and attachments.¹²

Class:	File		
Explanation:	A File is a data element that represents an address to a file (a locator). The value is an URI that can represent an absolute or relative path.		
Inherits from:	DataElement		
Attribute	Explanation	Type	Card.
value	Path and name of the file (with file extension). The path can be absolute or relative.	PathType	0..1
contentType	Content type of the content of the file.	ContentType	1

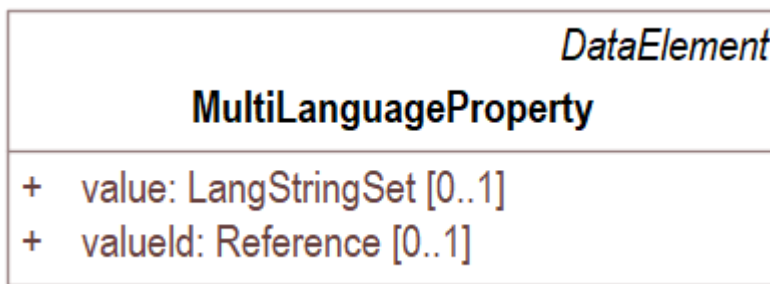
¹² Wikipedia.org, date: 2018-04-09

Class:	File		
	The content type states which file extensions the file can have.		

For handling of supplementary external files in exchanging AAS specification in AASX package format see also Clause 8.3 Conventions for the Asset Administration Shell Package File Format (AASX). An absolute path is used in the case that the file exists independently of the AAS. A relative path, relative to the package root should be used if the file is part of the serialized package of the AAS.

5.7.7.9 Multi Language Property Attributes

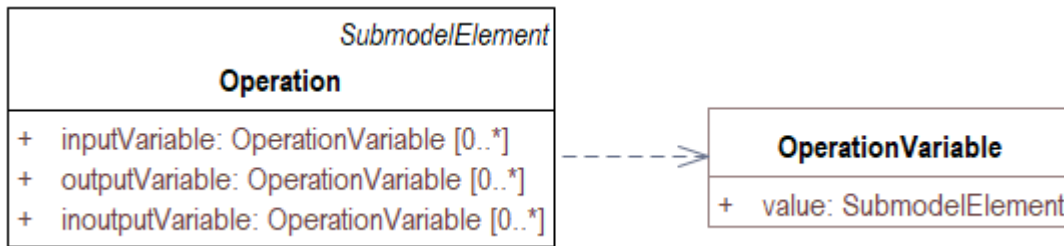
Figure 37 Metamodel of Multi Language Properties



Class:	MultiLanguageProperty		
Explanation:	A property is a data element that has a multi-language value. <i>Constraint AASd-012:</i> If both, the <i>MultiLanguageProperty/value</i> and the <i>MultiLanguageProperty/valueId</i> are present then for each string in a specific language the meaning must be the same as specified in <i>MultiLanguageProperty/valueId</i> .		
Inherits from:	DataElement		
Attribute	Explanation	Type	Card.
value	The value of the property instance.	LangStringSet	0..1
valueId	Reference to the global unique ID of a coded value. It is recommended to use a global reference.	Reference	0..1

5.7.7.10 Operation Attributes

Figure 38 Metamodel of Operations



Class:	Operation		
Explanation:	An operation is a submodel element with input and output variables.		
Inherits from:	SubmodelElement		
Attribute	Explanation	Type	Card.
inputVariable	Input parameter of the operation.	OperationVariable	0..*
outputVariable	Output parameter of the operation.	OperationVariable	0..*
inoutVariable	Parameter that is input and output of the operation.	OperationVariable	0..*

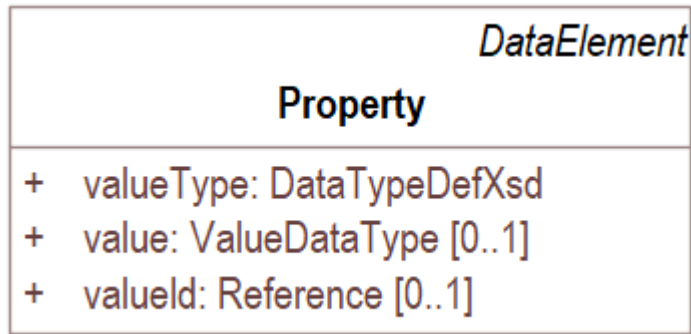
Class:	OperationVariable		
Explanation:	The value of an operation variable is a submodel element that is used as input and/or output variable of an operation.		
Inherits from:			
Attribute	Explanation	Type	Card.
value	Describes an argument or result of an operation via a submodel element	SubmodelElement	1

Note: Operations typically specify the behavior of a component in terms of procedures. Hence, operations enable the specification of services with procedure-based interactions [32].

Note: OperationVariable is introduced as separate class to enable future extensions, e.g. for adding a default value, cardinality (option/mandatory).

5.7.7.11 Property Attributes

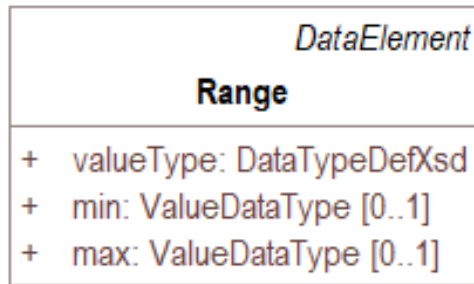
Figure 39 Metamodel of Properties



Class:	Property		
Explanation:	<p>A property is a data element that has a single value.</p> <p><u>Constraint AASd-007:</u> If both, the <i>Property/value</i> and the <i>Property/valueId</i> are present then the value of <i>Property/value</i> needs to be identical to the value of the referenced coded value in <i>Property/valueId</i>.</p>		
Inherits from:	DataElement		
Attribute	Explanation	Type	Card.
valueType	Data type of the value	DataTypeDefXsd	1
value	The value of the property instance.	ValueDataType	0..1
valueId	<p>Reference to the global unique ID of a coded value.</p> <p style="background-color: #e0f0ff;">It is recommended to use a global reference.</p>	Reference	0..1

5.7.7.12 Range Attributes

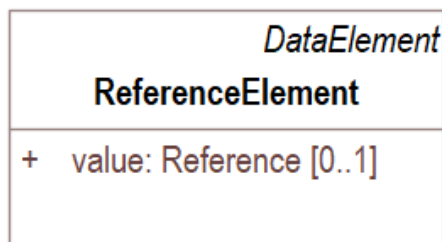
Figure 40 Metamodel of Ranges



Class:	Range		
Explanation:	A range data element is a data element that defines a range with min and max.		
Inherits from:	DataElement		
Attribute	Explanation	Type	Card.
valueType	Data type of the min und max	DataTypeDefXsd	1
min	The minimum value of the range. If the min value is missing, then the value is assumed to be negative infinite.	ValueDataType	0..1
max	The maximum value of the range. If the max value is missing, then the value is assumed to be positive infinite.	ValueDataType	0..1

5.7.7.13 Reference Element Attributes

Figure 41 Metamodel of Reference Elements



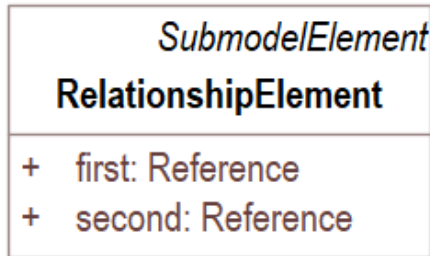
Class:	ReferenceElement		
Explanation:	A reference element is a data element that defines a logical reference to another element within the same or another AAS or a reference to an external object or entity.		
Inherits from:	DataElement		
Attribute	Explanation	Type	Card.
value	Global reference to an external object or entity or a logical reference to another element within the same	Reference	0..1

Class:	ReferenceElement		
	or another AAS (i.e. a model reference to a <i>Referable</i>).		

For more information on references see Clause 5.7.9.

5.7.7.14 Relationship Element Attributes

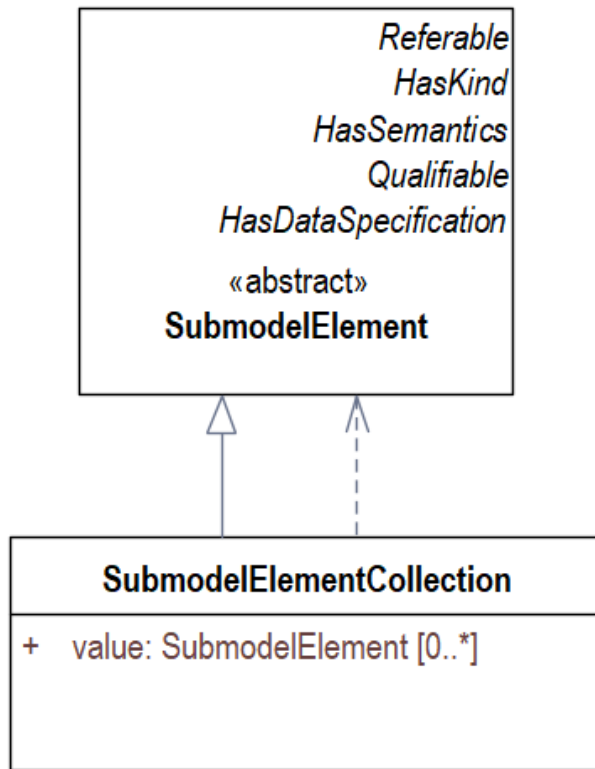
Figure 42 Metamodel of Relationship Elements



Class:	RelationshipElement		
Explanation:	A relationship element is used to define a relationship between two elements being either referable (model reference) or external (global reference).		
Inherits from:	SubmodelElement		
Attribute	Explanation	Type	Card.
first	Reference to the first element in the relationship taking the role of the subject.	Reference	1
second	Reference to the second element in the relationship taking the role of the object.	Reference	1

5.7.7.15 Submodel Element Collection Attributes

Figure 43 Metamodel of Submodel Element Collections



Submodel Element Collections are used for entities with a fixed set of properties with unique names within the struct. Each property within the collection should have a clearly defined semantics. A property of a struct can be any submodel element with a value.

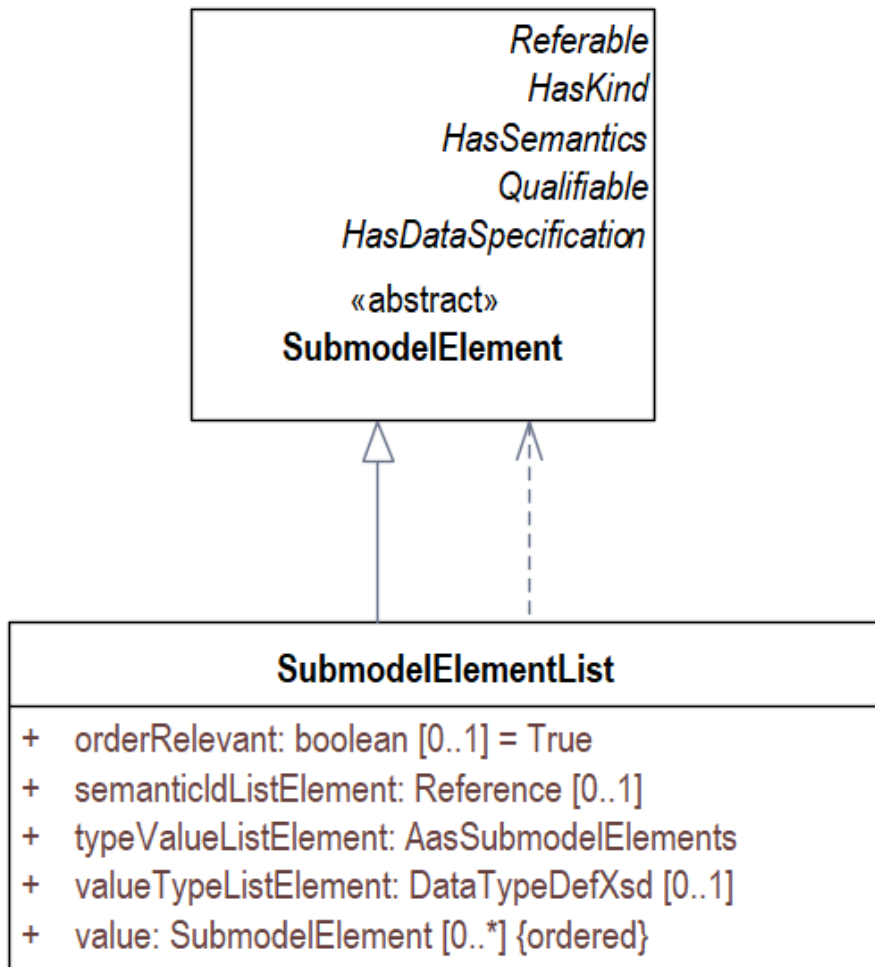
It is not required that the different elements of a submodel element struct have different semanticIds. However, in these cases the usage of a SubmodelElementList should be considered.

Example: For an asset there typically are many different documents available like the operating instructions, the safety instructions etc. Each single document has a predefined set of predefined properties like title, version, author etc. They logically belong to a document. So a single document is represented by a *SubmodelElementCollection*. The set of all documents is represented by a *SubmodelElementList*. So in this example we have a *SubmodelElementList* of *SubmodelElementCollections*.

Class:	SubmodelElementCollection		
Explanation:	A submodel element collection is a kind of struct, i.e. a logical encapsulation of multiple named values. It has a fixed number of submodel elements.		
Inherits from:	SubmodelElement		
Attribute	Explanation	Type	Card.
value	Submodel element contained in the collection.	SubmodelElement	0..*

5.7.7.16 Submodel Element List Attributes

Figure 44 Metamodel of Submodel Element Lists



Submodel Element Lists are used for sets (i.e. unordered collections without duplicates), ordered lists (i.e. ordered collections that may contain duplicates), bags (i.e. unordered collections that may contain duplicates) as well as for ordered sets (i.e. ordered collections without duplicates). They are realized via ordered collections of submodel elements.

Submodel Element Lists are also used to create multi-dimensional arrays. For a two-dimensional array `list[3][5]` with *Property* values it would be realized like this: The first submodel element list would contain 3 *SubmodelElementList* elements. In each of these 3 *SubmodelElementList* 5 single *Property* elements would be contained. The *semanticId* of the contained properties would be the same for all lists in the first list, i.e. *semanticIdListElement* would be identical for all three lists contained in the first list. The *semanticId* of the three contained lists would differ depending on the dimension it represents. In case of complex values in the array a *SubmodelElementCollection* would be used as values in the leaf lists.

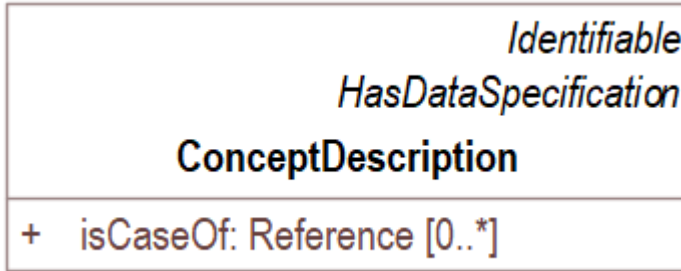
Similarly a table with 3 columns can be represented. In this case a *SubmodelElementCollection* with 3 *SubmodelElementLists* would be contained. In this case the *semanticId* as well as the *semanticIdListElement* for the three columns differ.

See Clause 5.4.6 matching strategies for semantic ids are explained.

Class:	SubmodelElementList		
Explanation:	<p>A submodel element list is an ordered list of submodel elements.</p> <p>The numbering starts with Zero (0).</p> <p><u>Constraint AASd-107:</u> If a first level child element in a <i>SubmodelElementList</i> has a <i>semanticId</i> it shall be identical to <i>SubmodelElementList/semanticIdListElement</i>.</p> <p><u>Constraint AASd-114:</u> If two first level child elements in a <i>SubmodelElementList</i> have a <i>semanticId</i> then they shall be identical.</p> <p><u>Constraint AASd-115:</u> If a first level child element in a <i>SubmodelElementList</i> does not specify a <i>semanticId</i> then the value is assumed to be identical to <i>SubmodelElementList/semanticIdListElement</i>.</p> <p><u>Constraint AASd-108:</u> All first level child elements in a <i>SubmodelElementList</i> shall have the same submodel element type as specified in <i>SubmodelElementList/typeValueListElement</i>.</p> <p><u>Constraint AASd-109:</u> If <i>SubmodelElementList/typeValueListElement</i> equal to <i>Property</i> or <i>Range</i> <i>SubmodelElementList/valueTypeListElement</i> shall be set and all first level child elements in the <i>SubmodelElementList</i> shall have the value type as specified in <i>SubmodelElementList/valueTypeListElement</i>.</p>		
Inherits from:	SubmodelElement		
Attribute	Explanation	Type	Card.
orderRelevant	<p>Defines whether order in list is relevant. If <i>orderRelevant</i> = False then the list is representing a set or a bag.</p> <p>Default: True</p>	boolean	0..1
value	Submodel element contained in the list.	SubmodelElement	0..*
semanticIdListElement	<p>Semantic ID the submodel elements contained in the list match to.</p> <p>It is recommended to use a global reference.</p>	Reference	0..1
typeValueListElement	The submodel element type of the submodel elements contained in the list.	SubmodelElementElements	1
valueTypeListElement	The value type of the submodel element contained in the list.	DataTypeDefXsd	0..1

5.7.8 Concept Description Attributes

Figure 45 Metamodel of Concept Descriptions



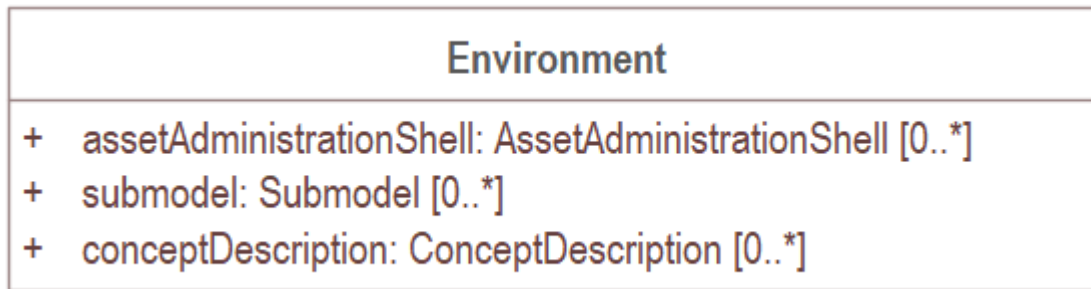
Class:	ConceptDescription		
Explanation:	<p>The semantics of a property or other elements that may have a semantic description is defined by a concept description.</p> <p>The description of the concept should follow a standardized schema (realized as data specification template).</p> <p><u>Constraint AASd-051</u>: A <i>ConceptDescription</i> shall have one of the following categories: VALUE, PROPERTY, REFERENCE, DOCUMENT, CAPABILITY, RELATIONSHIP, COLLECTION, FUNCTION, EVENT, ENTITY, APPLICATION_CLASS, QUALIFIER, VIEW. Default: PROPERTY.</p>		
Inherits from:	Identifiable; HasDataSpecification		
Attribute	Explanation	Type	Card.
isCaseOf	<p>Reference to an external definition the concept is compatible to or was derived from.</p> <p>It is recommended to use a global reference.</p> <p>Note: Compare to is-case-of relationship in ISO 13584-32 & IEC EN 61360</p>	Reference	0..*

Different types of submodel elements require different attributes for describing the semantics of them. This is why a concept description has at least one data specification template associated with it. Within this template the attributes needed to define the semantics are defined.

See Clause 5.7.12.3 for predefined data specification templates to be used.

5.7.9 Environment

Figure 46 Metamodel for Environment



Note w.r.t. file exchange: There is exactly one environment independent on how many files the contained elements are splitted. If the file is splitted then there shall be no element with the same identifier in two different files.

Class:	Environment		
Explanation:	Container for the sets of different identifiables.		
Inherits from:	Reference		
Attribute	Explanation	Type	Card.
assetAdministrationShell	Asset administration shell	AssetAdministrationShell	0..*
submodel	Submodel	Submodel	0..*
conceptDescription	Concept description	ConceptDescription	0..*

5.7.10 Referencing in Asset Administration Shells

5.7.10.1 Overview

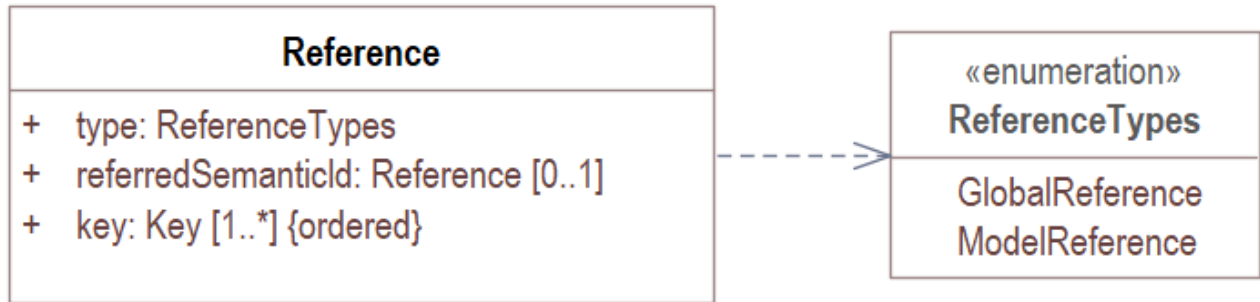
Up to now two kinds of references are distinguished: references to external objects or entities (global reference) and references to model element of the same or another Asset Administration Shell (model reference). Model references are also used for meta model inherent relationships like submodels of an AssetAdministrationShell etc. (notation see Annex C).

A global reference is a unique identifier. The identifier can be a concatenation of different identifiers, for example representing an IRDI path etc.

Note: References should not be mixed up with locators. Even URLs (URL = uniform resource locator) can be used as identifiers and do not necessarily describe a resource that can be accessed.

5.7.10.2 Reference Attributes

Figure 47 Metamodel of Reference



Class:	Reference		
Explanation:	<p>Reference to either a model element of the same or another AAS or to an external entity. A reference is an ordered list of keys.</p> <p>A model reference is an ordered list of keys, each key referencing an element. The complete list of keys may for example be concatenated to a path that then gives unique access to an element.</p> <p>A global reference is a reference to an external entity.</p>		
Inherits from:	--		
Attribute	Explanation	Type	Card.
type	Type of the reference. Denotes, whether reference is a global reference or a model reference.	ReferenceTypes	1
referredSemanticId	<i>SemanticId</i> of the referenced model element (<i>Reference/type=ModelReference</i>). For global references there typically is no semantic ID. It is recommended to use a global reference.	Reference	0..1
key <<ordered>>	Unique reference in its name space.	Key	1..*

5.7.10.3 Key Attributes

In Figure 48 a logical model of key types is presented. Depending on the context different enumerations can be derived. In the context of references the enumeration is “KeyTypes”.

Figure 48 Logical Model for Keys of References

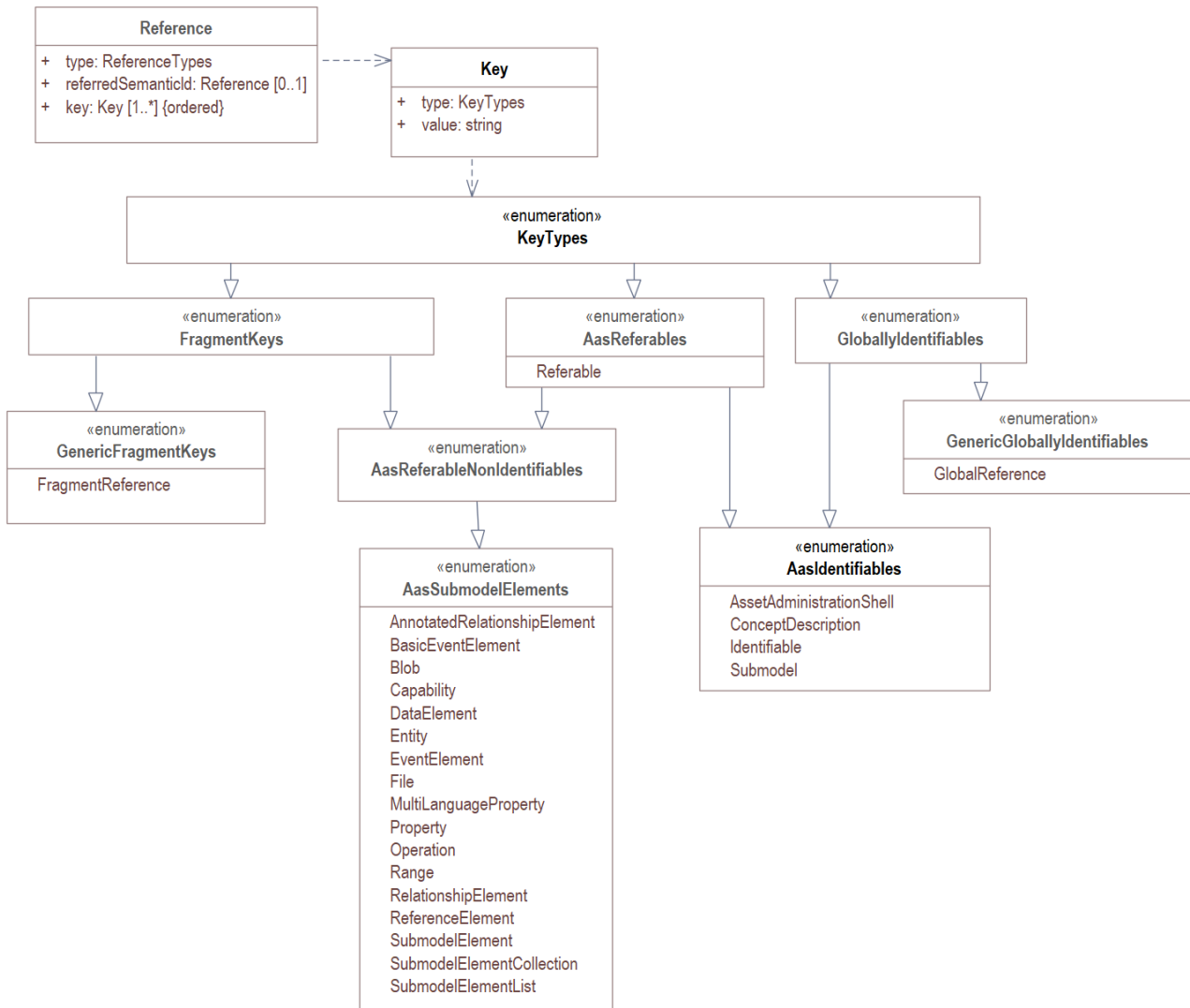
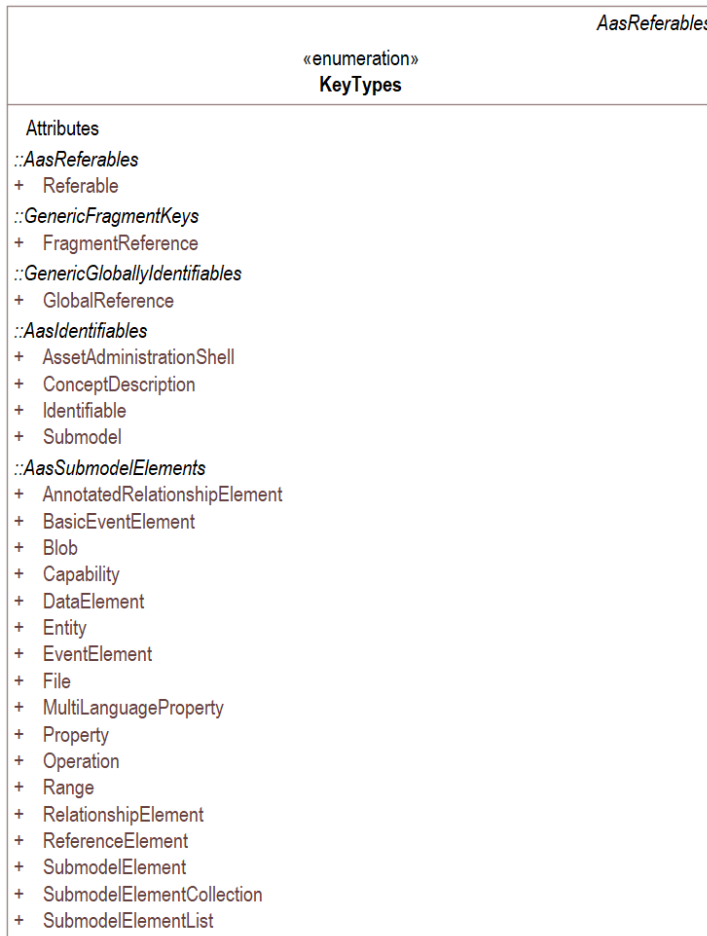


Figure 49 Metamodel of KeyTypes Enumeration



Class:	Key		
Explanation:	A key is a reference to an element by its ID.		
Inherits from:	--		
Attribute	Explanation	Type	Card.
type	Denotes which kind of entity is referenced. In case <i>type = FragmentId</i> the key represents a bookmark or a similar local identifier within its parent element as specified by the key that precedes this key. In all other cases the key references a model element of the same or of another AAS. The name of the model element is explicitly listed.	KeyElements	1
value	The key value, for example an IRDI or an URI	string	1

Example for using a *FragmentId* as type of a key: a reference to an element within a file that is part of an Asset Administration Shell *aasx* package.

Enumeration:	KeyTypes
Explanation:	Enumeration of different key value types within a key.
Set of:	FragementKeys, GloballyIdentifiables
Literal	Explanation
Referable	Referable Note: Referable is abstract, i.e. if a key uses "Referable" the reference may be an Asset Administration Shell, a Property etc.
FragmentReference	Bookmark or a similar local identifier of a subordinate part of a primary resource
GlobalReference	Global reference
AssetAdministrationShell	Asset administration shell
ConceptDescription	Concept Description
Identifiable	Identifiable Note: Identifiable is abstract, i.e. if a key uses "Identifiable" the reference may be an Asset Administration Shell, a Submodel or a Concept Description.
Submodel	Submodel
AnnotatedRelationshipElement	Annotated relationship element
BasicEventElement	Basic event element
Blob	Blob
Capability	Capability
DataElement	Data Element. Note: Data Element is abstract, i.e. if a key uses "DataElement" the reference may be a Property, a File etc.
Entity	Entity
EventElement	Event Note: Event element is abstract.
File	File
MultiLanguageProperty	Property with a value that can be provided in multiple languages
Operation	Operation
Property	Property
Range	Range with min and max
ReferenceElement	Reference
RelationshipElement	Relationship
SubmodelElement	Submodel element

Enumeration:	KeyTypes
Explanation:	Enumeration of different key value types within a key.
Set of:	FragementKeys, GloballyIdentifiables
Literal	Explanation
	Note: Submodel Element is abstract, i.e. if a key uses "SubmodelElement" the reference may be a Property, a SubmodelElementList, an Operation etc.
SubmodelElementCollection	Struct of submodel elements
SubmodelElementList	List of submodel elements

Enumeration:	FragmentKeys
Explanation:	Enumeration of different fragment key value types within a key.
Set of:	AASReferableNonIdentifiables, GenericFragmentKeys
Literal	Explanation
FragmentReference	Bookmark or a similar local identifier of a subordinate part of a primary resource
AnnotatedRelationshipElement	Annotated relationship element
BasicEventElement	Basic event element
Blob	Blob
Capability	Capability
DataElement	Data Element. Note: Data Element is abstract, i.e. if a key uses "DataElement" the reference may be a Property, a File etc.
Entity	Entity
EventElement	Event Note: Event element is abstract.
File	File
MultiLanguageProperty	Property with a value that can be provided in multiple languages
Operation	Operation
Property	Property
Range	Range with min and max
ReferenceElement	Reference
RelationshipElement	Relationship
SubmodelElement	Submodel element

Enumeration:	FragmentKeys
Explanation:	Enumeration of different fragment key value types within a key.
Set of:	AASReferableNonIdentifiabes, GenericFragmentKeys
Literal	Explanation
	Note: Submodel Element is abstract, i.e. if a key uses "SubmodelElement" the reference may be a Property, a SubmodelElementList, an Operation etc.
SubmodelElementCollection	Struct of submodel elements
SubmodelElementList	List of submodel elements

Enumeration:	GloballyIdentifiabes
Explanation:	Enumeration of different key value types within a key.
Set of:	AASIdentifiabes, GenericGloballyIdentifiabes
Literal	Explanation
GlobalReference	Global reference
AssetAdministrationShell	Asset administration shell
ConceptDescription	Concept Description
Identifiable	Identifiable Note: Identifiable is abstract, i.e. if a key uses "Identifiable" the reference may be an Asset Administration Shell, a Submodel or a Concept Description.
Submodel	Submodel

Enumeration:	AasReferableNonIdentifiabes
Explanation:	Enumeration of different fragment key value types within a key.
Set of:	AasSubmodelElements
Literal	Explanation
AnnotatedRelationshipElement	Annotated relationship element
BasicEventElement	Basic event element
Blob	Blob
Capability	Capability
DataElement	Data Element. Note: Data Element is abstract, i.e. if a key uses "DataElement" the reference may be a Property, a File etc.
Entity	Entity
EventElement	Event

Enumeration:	AasReferableNonIdentifiables
Explanation:	Enumeration of different fragment key value types within a key.
Set of:	AasSubmodelElements
Literal	Explanation
	Note: Event element is abstract.
File	File
MultiLanguageProperty	Property with a value that can be provided in multiple languages
Operation	Operation
Property	Property
Range	Range with min and max
ReferenceElement	Reference
RelationshipElement	Relationship
SubmodelElement	Submodel element Note: Submodel Element is abstract, i.e. if a key uses "SubmodelElement" the reference may be a Property, a SubmodelElementList, an Operation etc.
SubmodelElementCollection	Struct of submodel elements
SubmodelElementList	List of submodel elements

Enumeration:	AasSubmodelElements
Explanation:	Enumeration of different fragment key value types within a key.
Set of:	--
Literal	Explanation
AnnotatedRelationshipElement	Annotated relationship element
BasicEventElement	Basic event element
Blob	Blob
Capability	Capability
DataElement	Data Element. Note: Data Element is abstract, i.e. if a key uses "DataElement" the reference may be a Property, a File etc.
Entity	Entity
EventElement	Event Note: Event element is abstract.

Enumeration:	AasSubmodelElements
Explanation:	Enumeration of different fragment key value types within a key.
Set of:	--
Literal	Explanation
File	File
MultiLanguageProperty	Property with a value that can be provided in multiple languages
Operation	Operation
Property	Property
Range	Range with min and max
ReferenceElement	Reference
RelationshipElement	Relationship
SubmodelElement	Submodel element Note: Submodel Element is abstract, i.e. if a key uses "SubmodelElement" the reference may be a Property, a SubmodelElementList, an Operation etc.
SubmodelElementCollection	Struct of submodel elements
SubmodelElementList	List of submodel elements

Enumeration:	AasReferables
Explanation:	Enumeration of referables
Set of:	AasReferableNonIdentifiables, AasIdentifiables
Literal	Explanation
Referable	Referable Note: Referable is abstract, i.e. if a key uses "Referable" the reference may be an Asset Administration Shell, a Property etc.
AssetAdministrationShell	Asset administration shell
ConceptDescription	Concept Description
Identifiable	Identifiable Note: Identifiable is abstract, i.e. if a key uses "Identifiable" the reference may be an Asset Administration Shell, a Submodel or a Concept Description.
Submodel	Submodel
AnnotatedRelationshipElement	Annotated relationship element
BasicEventElement	Basic event element
Blob	Blob
Capability	Capability

Enumeration:	AasReferables
Explanation:	Enumeration of referables
Set of:	AasReferableNonIdentifiables, AasIdentifiables
Literal	Explanation
DataElement	Data Element. Note: Data Element is abstract, i.e. if a key uses "DataElement" the reference may be a Property, a File etc.
Entity	Entity
EventElement	Event Note: Event element is abstract.
File	File
MultiLanguageProperty	Property with a value that can be provided in multiple languages
Operation	Operation
Property	Property
Range	Range with min and max
ReferenceElement	Reference
RelationshipElement	Relationship
SubmodelElement	Submodel element Note: Submodel Element is abstract, i.e. if a key uses "SubmodelElement" the reference may be a Property, a SubmodelElementList, an Operation etc.
SubmodelElementCollection	Struct of submodel elements
SubmodelElementList	List of submodel elements

Enumeration:	GenericFragmentKeys
Explanation:	Enumeration of different fragment key value types within a key.
Set of:	--
Literal	Explanation
FragmentReference	Bookmark or a similar local identifier of a subordinate part of a primary resource

Enumeration:	AasIdentifiables
Explanation:	Enumeration of different key value types within a key.
Set of:	--
Literal	Explanation
AssetAdministrationShell	Asset administration shell
ConceptDescription	Concept Description

Enumeration:	AasIdentifiables
Explanation:	Enumeration of different key value types within a key.
Set of:	--
Literal	Explanation
Identifiable	Identifiable Note: Identifiable is abstract, i.e. if a key uses "Identifiable" the reference may be an Asset Administration Shell, a Submodel or a Concept Description.
Submodel	Submodel

Enumeration:	GenericGloballyIdentifiables
Explanation:	Enumeration of different key value types within a key.
Set of:	--
Literal	Explanation
GlobalReference	Global reference

5.7.10.4 Constraints

Constraint AASd-121: For *References* the *type* of the first *key* of *Reference/keys* shall be one of *GloballyIdentifiables*.

Constraint AASd-122: For global references, i.e. *References* with *Reference/type* = *GlobalReference*, the *type* of the first *key* of *Reference/keys* shall be one of *GenericGloballyIdentifiables*.

Constraint AASd-123: For model references, i.e. *References* with *Reference/type* = *ModelReference*, the *type* of the first *key* of *Reference/keys* shall be one of *AasIdentifiables*.

Constraint AASd-124: For global references, i.e. *References* with *Reference/type* = *GlobalReference*, the last *key* of *Reference/keys* shall be either one of *GenericGloballyIdentifiables* or one of *GenericFragmentKeys*.

Constraint AASd-125: For model references, i.e. *References* with *Reference/type* = *ModelReference*, with more than one *key* in *Reference/keys* the *type* of the keys following the first *key* of *Reference/keys* shall be one of *FragmentKeys*.

Note: Constraint AASd-125 ensures that the shortest path is used.

Constraint AASd-126: For model references, i.e. *References* with *Reference/type* = *ModelReference*, with more than one *key* in *Reference/keys* the *type* of the last *Key* in the reference key chain may be one of *GenericFragmentKeys* or no *key* at all shall have a value out of *GenericFragmentKeys*.

Constraint AASd-127: For model references, i.e. *References* with *Reference/type* = *ModelReference*, with more than one *key* in *Reference/keys* a *key* with *type* *FragmentReference* shall be preceded by a *key* with *type* *File* or *Blob*. All other AAS fragments, i.e. *type* values out of *AasSubmodelElements*, do not support fragments.

Note: Which kind of fragments are supported depends on the content type and the specification of allowed fragment identifiers for the corresponding resource being referenced via the reference.

Constraint AASd-128: For model references, i.e. *References* with *Reference/type* = *ModelReference*, the *Key/value* of a *Key* preceded by a *Key* with *Key/type*=*SubmodelElementList* is an integer number denoting the position in the array of the submodel element list.

Examples for valid references:

(Submodel)http://example.com/aas/1/1/1234859590

(GlobalReference)http://example.com/specification.html

Examples for valid global references:

(GlobalReference)http://example.com/ressource

(GlobalReference)0173-1#02-EXA123#001

(GlobalReference)http://example.com/specification.html (FragmentReference)Hints

Note: *(GlobalReference)http://example.com/specification.html (FragmentReference)Hints* represents the path with fragment identifier *http://example.com/specification.html#Hints*

Examples for valid model references:

(AssetAdministrationShell)http://example.com/aas/1/0/12348

(Submodel)http://example.com/aas/1/1/1234859590

(Submodel)http://example.com/aas/1/1/1234859590, (File)Specification

(ConceptDescription)0173-1#02-BAA120#008

(Submodel)http://example.com/aas/1/1/1234859590, (SubmodelElementList)Documents, (SubmodelElementCollection)0, (MultiLanguageProperty)Title

(Submodel)http://example.com/aas/1/1/1234859590, (SubmodelElementCollection)Manual, (MultiLanguageProperty)Title

Note: *(SubmodelElementCollection)0, (MultiLanguageProperty)Title* may be semantically and content-wise identical to *(SubmodelElementCollection)Manual, (MultiLanguageProperty)Title*. The difference is that in the first submodels more than one document is allowed and thus a submodel element list is defined: elements in a list are numbered. However, it is possible to define a display name also in this case. So the display name of the *SubmodelElementCollection* should be the same in both bases, e.g. "Users Manual".

(Submodel)http://example.com/aas/1/1/1234859590, (File)Specification, (FragmentReference)Hints

Assuming the *File* has the value using the absolute path (and not a relative path) *http://example.com/specification.html* then the first reference points to the same information as the global reference *(GlobalReference)http://example.com/specification.html, (FragmentReference)Hints*.

(Submodel)http://example.com/aas/1/1/1234859590, (Blob)Specification, (FragmentReference)Hints

Examples for invalid model references:

(GlobalReference)http://example.com/aas/1/1/1234859590

(Property)0173-1#02-BAA120#008

(Submodel)http://example.com/aas/1/1/1234859590, (EventElement)Event, (FragmentReference)Comment

(AssetAdministrationShell)<http://example.com/aas/1/0/12348>,

(Submodel)<http://example.com/aas/1/1/1234859590>, **(Property)***Temperature*

is not a valid model reference because *AssetAdministrationShell* and *Submodel* both are globally identifiable.

5.7.10.5 Matching Algorithm for References

In Clause 5.4.6 matching strategies for semantic identifiers were discussed. In this Clause matching strategies based on the reference concept is explained in more details and thus also covers other kind of identifying elements.

For examples the string serialization of references as defined in Clause 9.2.3 is used for easier understanding.

Exact match:

- A global reference A matches a global reference B if all values of all keys are identical. Note: it is assumed as unlikely that a fragment value is identical to a global reference value and thus references something different.
- A model reference A matches a model reference B if all values of all keys are identical. Note: the key type can be ignored since the fragment keys are always unique (e.g. all idShorts of Submodel elements in a submodel or all submodel elements in a submodel element list or collection).
- A global reference A matches a model reference B and vice versa if all values of all keys are identical. Note: Since Identifiables of the AAS are globally unique, model references are special cases of global references. The only difference is the handling of key types that are predefined for AAS element. Other key types could be predefined, e.g. for IRDI paths etc. but so far only generic key types are supported.

Note: If the key types are not identical although all key values are following the correct order of the key chain then at least of the references is buggy and a warning should be raised.

5.7.11 Templates, Inheritance, Qualifiers and Categories

On a first glance there seem to be some overlapping between the concept of data specification templates, extensions, inheritance, qualifiers and categories. In this clause the commonalities and differences are explained and hints for good practices are given.

In general extension of the metamodel by inheritance is foreseen. As an alternative also templates might be used.

- Extensions can be used to add proprietary and/or temporary information to an element. Extensions do not support interoperability. They can be used as work-around for missing properties in the standard. In this case the same extensions are attaches to all elements of a specific class (e.g. to Properties). However, in general extensions can be attached in a quite arbitrary way. Properties are defined in a predefined way as key values pairs (key named "name" in this case).
- Templates in contrast to extensions aim to enable interoperability between the partners that agree on the template. Templates should only be used if different instances of the class follow different schemas and the templates for the schemas are not known at design time of the metamodel. Templates might also be used if the overall metamodel is not yet stable enough or a tool does support templates but not (yet) the complete metamodel. Typically all instances of a specific class having the same category provide the same attribute values conformant to the template. In contrast to extensions the attributes in the template have speaking names.
- However: when using non-standardized proprietary data specification templates interoperability cannot be ensured and thus should be avoided whenever possible.
- In case all instances of a class follow the same schema then inheritance and/or categories should be used.
- Categories can be used if all instances of a class follow the same schema but have different constraints depending on its category. Such a constraint might specify that an optional attribute is mandatory for

this category (like for example the unit that is mandatory for properties representing physical values). Realizing the same via inheritance would lead to multiple inheritance what is to be omitted¹³.

- Qualifiers are used if the structure and its semantics of the element is the same independent of its qualifiers. It is only the quality or the meaning of the value for the element that differs.

5.7.12 Primitive and Simple Data Types

5.7.12.1 Predefined Simple Data Types

The metamodel of the Asset Administration Shell uses the following basic data types as defined in the XML Schema Definition (XSD)¹⁴. Their definition is outside the scope of this document.

The meaning and format of xsd types is specified in <https://www.w3.org/XML/Schema>. The simple type “langString” is specified in the Resource Description Framework (RDF)¹⁵.

Source	Basic Data Type	Value Range	Sample Values
xsd	string	Character string (but not all Unicode character strings)	"Hello world", "Καλημέρα κόσμε", "コンニチハ"
xsd	boolean	true, false	true, false
xsd	byte	-128...+127 (8 bit)	-1, 0, 127
rdf	langString	Strings with language tags	"Hello"@en, "Hallo"@de. Note that this is written in RDF/Turtle syntax, and that only "Hello" and "Hallo" are the actual values.

Simple data types start with a small letter.

In addition to these types there is a special handling of data elements with varying types (i.e. with primitive data type “ValueDataType”). They are discussed in Clause 5.7.12.2 and Clause 5.7.12.3.

5.7.12.2 Primitive Data Types

Types that are used for specific data specification templates are listed in the corresponding clause of the data specification.

Table 6 lists the Primitives used in the metamodel. Primitive data types start with a capital letter.

Table 6 Primitive DataTypes Used in Metamodel

Primitive	Explanation	Value Examples
BlobType	Group of <i>bytes</i> to represent file content (binaries and non-binaries)	<pre><?xml version="1.0" encoding="UTF-8"?> <schema elementFormDefault="qualified" targetNamespace="http://www.admin-shell.io/aas/2/0" xmlns="http://www.w3.org/2001/XMLSchema" xmlns:aas="http://www.admin-shell.io/aas/2/0" xmlns:abac="http://www.admin-shell.io/aas/abac/2/0" xmlns:IEC61360="http://www.admin-shell.io/IEC61360/2/0"> <import namespace="http://www.admin-shell.io/aas/abac/2/0" schemaLocation="AAS_ABAC.xsd"/></pre>

¹³ Exception: In this specification multiple inheritance is used but only in case of inheriting from abstract classes.

¹⁴ <https://www.w3.org/XML/Core/>, former <https://www.w3.org/XML/Schema>

¹⁵ see: <https://www.w3.org/TR/rdf11-concepts/>

Primitive	Explanation	Value Examples
		<p>MZ _____ÿÿ_____, _____@_____</p> <p>_____€_____°_____´</p> <p>Í! ,_LÍ!This program cannot be run in DOS</p> <p>mode.\$ _____PE__L__Rö\^ _____à_</p>
Identifier	<i>string</i>	<p>https://cust/123456</p> <p>0173-1#02-BAA120#008</p>
LangStringSet	<p><i>Array of elements of type langString</i></p> <p>langString is a RDF data type. A langString is a string value tagged with a language code.</p> <p>It depends on the serialization rules for a technology how this is realized.</p>	<p>In xml:</p> <pre><aas:langString lang="EN">This is a multi-language value in English</aas:langString></pre> <pre><aas:langString lang="DE"> Das ist ein Multi-Language-Wert in Deutsch </aas:langString></pre> <p>In rdf:</p> <pre>"This is a multi-language value in English"@en ;</pre> <pre>"Das ist ein Multi-Language-Wert in Deutsch"@de</pre> <p>In JSON:</p> <pre>{ "language": "EN", "text": " This is a multi-language value in English." }</pre> <pre>{ "language": "DE", "text": " Das ist ein Multi-Language-Wert in Deutsch." }</pre>
ContentType	<p><i>string</i></p> <p>Any content type as in RFC2046.</p> <p>A media type (also MIME type and content type) [...] is a two-part identifier for file formats and format contents transmitted on the Internet. The Internet Assigned Numbers Authority (IANA) is the official authority for the standardization and publication of these classifications. Media types were originally defined in Request for</p>	<p>application/pdf</p> <p>image/jpeg</p>

Primitive	Explanation	Value Examples
	Comments 2045 in November 1996 as a part of MIME (Multipurpose Internet Mail Extensions) specification, for denoting type of email message content and attachments. ¹⁶	
PathType	<i>string</i> Any string conformant to RFC8089 ¹⁷ , the “file” URI scheme (for relative and absolute file paths)	./Specification.pdf file:c:/local/Specification.pdf file://host.example.com/path/to/file
QualifierType	<i>string</i>	ExpressionSemantic life cycle qual
ValueDataType	<i>any xsd atomic type as specified via DataTypeDefXsd</i>	“This is a string value” 10 1.5 2020-04-01 True

5.7.12.3 Enumeration for Submodel Element Value Types

Enumerations are primitive data types. Most of the enumerations are defined in the context of the class they are used. In this clause enumerations for submodel element value types¹⁸ are defined.

The predefined types used to define the type of values of properties and other values use the names and the semantics of XML Schema Definition (XSD)¹⁹. Additionally, the type “langString” with the semantics as defined in the Resource Description Framework (RDF)²⁰ is used. “langString” is a string value tagged with a language code.

RDF²¹ recommends to not use the following xsd data types. This is, why they are excluded from the allowed data types.

- XSD BuildIn List Types are not supported (ENTITIES, IDREFS and NMTOKENS).
- XSD string BuildIn Types are not supported (normalizedString, token, language, NCName, ENTITY, ID, IDREF).
- The following XSD primitive types are not supported: NOTATION, QName.

¹⁶ Wikipedia.org, date: 2018-04-09

¹⁷ <https://datatracker.ietf.org/doc/html/rfc8089>

¹⁸ E.g. *Property/valueType*

¹⁹ See <https://www.w3.org/XML/Schema>

²⁰ see: <https://www.w3.org/TR/rdf11-concepts/>

²¹ See <https://www.w3.org/TR/rdf11-concepts/#xsd-datatypes>

Note: The following RDF types are not supported: HTML and XMLLiteral.

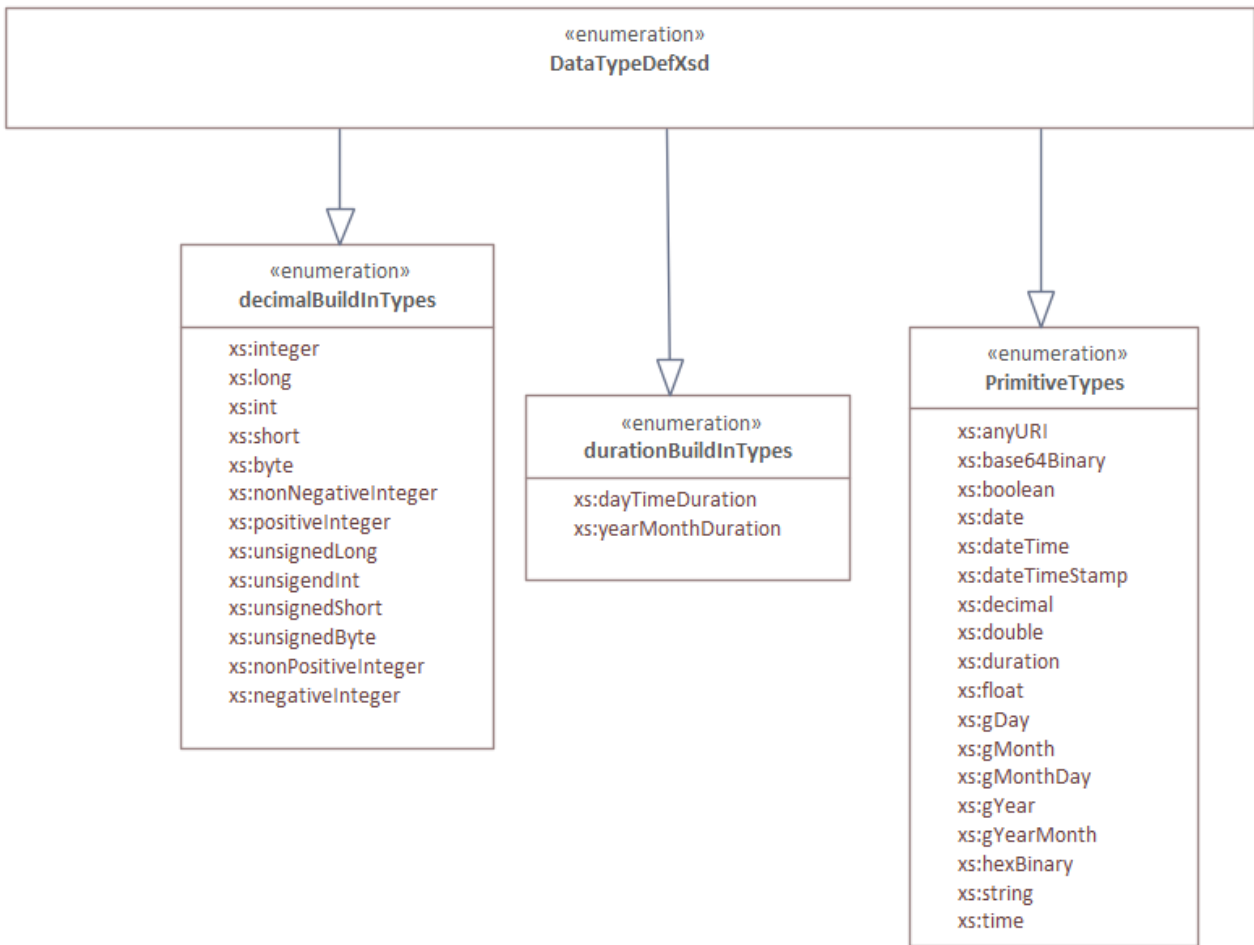
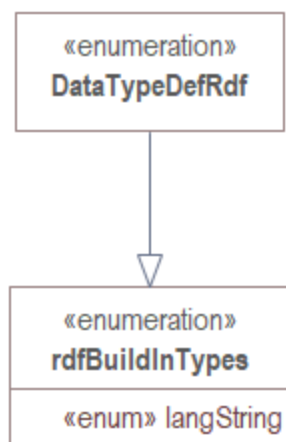


Figure 51 DefTypeDefRdf Enumeration



Example values and the value range of the different data types are given in Table 7. The left column “Data Type” shows the data types which can be used for submodel element values. The data types are defined according to the W3C XML Schema (<https://www.w3.org/TR/xmlschema-2/#built-in-datatypes> and <https://www.w3.org/TR/xmlschema-2/#built-in-derived>). “Value Range” further explains the possible range of data values for this data type. In the right column are related examples for values of the corresponding data type.

Table 7 Data Types with Examples²²

	Data Type	Value Range	Sample Values
Core Types	xs:string	Character string (but not all Unicode character strings)	"Hello world", "Καλημέρα κόσμε", "コンニチハ"
	xs:boolean	true, false	true, false
	xs:decimal	Arbitrary-precision decimal numbers	-1.23, 126789672374892739424.543233, +100000.00, 210
	xs:integer	Arbitrary-size integer numbers	-1, 0, 126789675432332938792837429837429837429, +100000
IEEE-floating-point numbers	xs:double	64-bit floating point numbers incl. ±Inf, ±0, NaN	-1.0, +0.0, -0.0, 234.567e8, -INF, NaN
	xs:float	32-bit floating point numbers incl. ±Inf, ±0, NaN	-1.0, +0.0, -0.0, 234.567e8, -INF, NaN
Time and data	xs:date	Dates (yyyy-mm-dd) with or without timezone	"2000-01-01", "2000-01-01Z", "2000-01-01+12:05"
	xs:time	Times (hh:mm:ss.sss...) with or without timezone	"14:23:00", "14:23:00.527634Z", "14:23:00+03:00"
	xs:dateTime	Date and time with or without timezone	"2000-01-01T14:23:00", "2000-01-01T14:23:00.66372+14:00"
	xs:dateTimeStamp	Date and time with required timezone	"2000-01-01T14:23:00.66372+14:00"
Recurring and partial dates	xs:gYear	Gregorian calendar year	"2000", "2000+03:00"
	xs:gMonth	Gregorian calendar month	"--04", "--04+03:00"
	xs:gDay	Gregorian calendar day of the month	"---04", "---04+03:00"
	xs:gYearMonth	Gregorian calendar year and month	"2000-01", "2000-01+03:00"
	xs:gMonthDay	Gregorian calendar month and day	"--01-01", "--01-01+03:00"
	xs:duration	Duration of time	"P30D", "-P1Y2M3DT1H", "PT1H5M0S"
	xs:yearMonthDuration	Duration of time (months and years only)	"P10M", "P5Y2M"
	xs:dayTimeDuration	Duration of time (days, hours, minutes, seconds only)	"P30D", "P1DT5H", "PT1H5M0S"
	xs:byte	-128...+127 (8 bit)	-1, 0, 127

²² See list of RDF-compatible XSD types with short description <https://www.w3.org/TR/rdf11-concepts/#xsd-datatypes>. Examples from <https://openmanufacturingplatform.github.io/sds-bamm-aspect-meta-model/bamm-specification/v1.0.0/datatypes.html>

Limited-range integer numbers	xs:short	-32768...+32767 (16 bit)	-1, 0, 32767
	xs:int	2147483648...+2147483647 (32 bit)	-1, 0, 2147483647
	xs:long	-9223372036854775808...+9223372036854775807 (64 bit)	-1, 0, 9223372036854775807
	xs:unsignedByte	0...255 (8 bit)	0, 1, 255
	xs:unsignedShort	0...65535 (16 bit)	0, 1, 65535
	xs:unsignedInt	0...4294967295 (32 bit)	0, 1, 4294967295
	xs:unsignedLong	0...18446744073709551615 (64 bit)	0, 1, 18446744073709551615
	xs:positiveInteger	Integer numbers >0	1, 7345683746578364857368475638745
	xs:nonNegativeInteger	Integer numbers ≥0	0, 1, 7345683746578364857368475638745
	xs:negativeInteger	Integer numbers <0	-1, -23487263847628376482736487263847
xs:nonPositiveInteger	Integer numbers ≤0	-1, 0, -93845837498573987498798987394	
Encoded binary data	xs:hexBinary	Hex-encoded binary data	"6b756d6f77617368657265"
	xs:base64Binary	Base64-encoded binary data	"a3Vtb3dhc2hlcmU="
Miscellaneous types	xs:anyURI	Absolute or relative URIs and IRIs	"http://customer.com/demo/aas/1/1/1234859590", "urn:example:company:1.0.0"
	rdf:langString	Strings with language tags	"Hello"@en, "Hallo"@de. Note that this is written in RDF/Turtle syntax, and that only "Hello" and "Hallo" are the actual values.

Enumeration:	DataTypeDefXsd
Explanation:	Enumeration listing all xsd anySimpleTypes For more details see https://www.w3.org/TR/rdf11-concepts/#xsd-datatypes
Set of:	DecimalBuildInTypes, durationBuildInTypes, PrimitiveTypes
Literal	Explanation
xs:integer	see: https://www.w3.org/TR/xmlschema11-2/#integer
xs:long	see: https://www.w3.org/TR/xmlschema11-2/#long
xs:int	see: https://www.w3.org/TR/xmlschema11-2/#int
xs:short	see: https://www.w3.org/TR/xmlschema11-2/#short
xs:byte	see: https://www.w3.org/TR/xmlschema11-2/#byte

Enumeration:	DataTypeDefXsd
xs:nonNegativeInteger	see: https://www.w3.org/TR/xmlschema11-2/#nonNegativeInteger
xs:positiveInteger	see: https://www.w3.org/TR/xmlschema11-2/#positiveInteger
xs:unsignedLong	see: https://www.w3.org/TR/xmlschema11-2/#unsignedLong
xs:unsignedInt	see: https://www.w3.org/TR/xmlschema11-2/#unsignedInt
xs:unsignedShort	see: https://www.w3.org/TR/xmlschema11-2/#unsignedShort
xs:unsignedByte	see: https://www.w3.org/TR/xmlschema11-2/#unsignedShort
xs:nonPositiveInteger	see: https://www.w3.org/TR/xmlschema11-2/#nonPositiveInteger
xs:negativeInteger	see: https://www.w3.org/TR/xmlschema11-2/#negativeInteger
xs:daytimeDuration	see: https://www.w3.org/TR/xmlschema11-2/#dayTimeDuration
xs:yearMonthDuration	see: https://www.w3.org/TR/xmlschema11-2/#yearMonthDuration
xs:anyURI	see: https://www.w3.org/TR/xmlschema-2/#anyURI
xs:base64Binary	see: https://www.w3.org/TR/xmlschema-2/#base64Binary
xs:boolean	see: https://www.w3.org/TR/xmlschema-2/#boolean
xs:date	see: https://www.w3.org/TR/xmlschema-2/#date
xs:dateTime	see: https://www.w3.org/TR/xmlschema-2/#dateTime
xs:decimal	see: https://www.w3.org/TR/xmlschema-2/#decimal
xs:double	see: https://www.w3.org/TR/xmlschema-2/#double
xs:duration	see: https://www.w3.org/TR/xmlschema-2/#duration
xs:float	see: https://www.w3.org/TR/xmlschema-2/#float
xs:gDay	see: https://www.w3.org/TR/xmlschema-2/#gDay
xs:gMonth	see: https://www.w3.org/TR/xmlschema-2/#gMonth
xs:gMonthDay	see: https://www.w3.org/TR/xmlschema-2/#gMonthDay
xs:gYear	see: https://www.w3.org/TR/xmlschema-2/#gYear
xs:gYearMonth	see: https://www.w3.org/TR/xmlschema-2/#gYearMonth
xs:hexBinary	see: https://www.w3.org/TR/xmlschema-2/#hexBinary
xs:string	see: https://www.w3.org/TR/xmlschema-2/#string
xs:time	see: https://www.w3.org/TR/xmlschema-2/#time

Enumeration:	decimalBuildInTypes
--------------	---------------------

Explanation:	Enumeration listing all xsd build in decimal types
Set of:	--
Literal	Explanation
xs:integer	see: https://www.w3.org/TR/xmlschema11-2/#integer
xs:long	see: https://www.w3.org/TR/xmlschema11-2/#long
xs:int	see: https://www.w3.org/TR/xmlschema11-2/#int
xs:short	see: https://www.w3.org/TR/xmlschema11-2/#short
xs:byte	see: https://www.w3.org/TR/xmlschema11-2/#byte
xs:nonNegativeInteger	see: https://www.w3.org/TR/xmlschema11-2/#nonNegativeInteger
xs:positiveInteger	see: https://www.w3.org/TR/xmlschema11-2/#positiveInteger
xs:unsignedLong	see: https://www.w3.org/TR/xmlschema11-2/#unsignedLong
xs:unsignedInt	see: https://www.w3.org/TR/xmlschema11-2/#unsignedInt
xs:unsignedShort	see: https://www.w3.org/TR/xmlschema11-2/#unsignedShort
xs:unsignedByte	see: https://www.w3.org/TR/xmlschema11-2/#unsignedShort
xs:nonPositiveInteger	see: https://www.w3.org/TR/xmlschema11-2/#nonPositiveInteger
xs:negativeInteger	see: https://www.w3.org/TR/xmlschema11-2/#negativeInteger

Enumeration:	durationBuildInTypes
Explanation:	Enumeration listing all xsd build in types with respect to duration
Set of:	--
Literal	Explanation
xs:daytimeDuration	see: https://www.w3.org/TR/xmlschema11-2/#dayTimeDuration
xs:yearMonthDuration	see: https://www.w3.org/TR/xmlschema11-2/#yearMonthDuration

Enumeration:	PrimitiveTypes
Explanation:	Enumeration listing all xsd primitive types
Set of:	--
Literal	Explanation
xs:anyURI	see: https://www.w3.org/TR/xmlschema-2/#anyURI
xs:base64Binary	see: https://www.w3.org/TR/xmlschema-2/#base64Binary
xs:boolean	see: https://www.w3.org/TR/xmlschema-2/#boolean
xs:date	see: https://www.w3.org/TR/xmlschema-2/#date
xs:dateTime	see: https://www.w3.org/TR/xmlschema-2/#dateTime
xs:decimal	see: https://www.w3.org/TR/xmlschema-2/#decimal
xs:double	see: https://www.w3.org/TR/xmlschema-2/#double
xs:duration	see: https://www.w3.org/TR/xmlschema-2/#duration
xs:float	see: https://www.w3.org/TR/xmlschema-2/#float

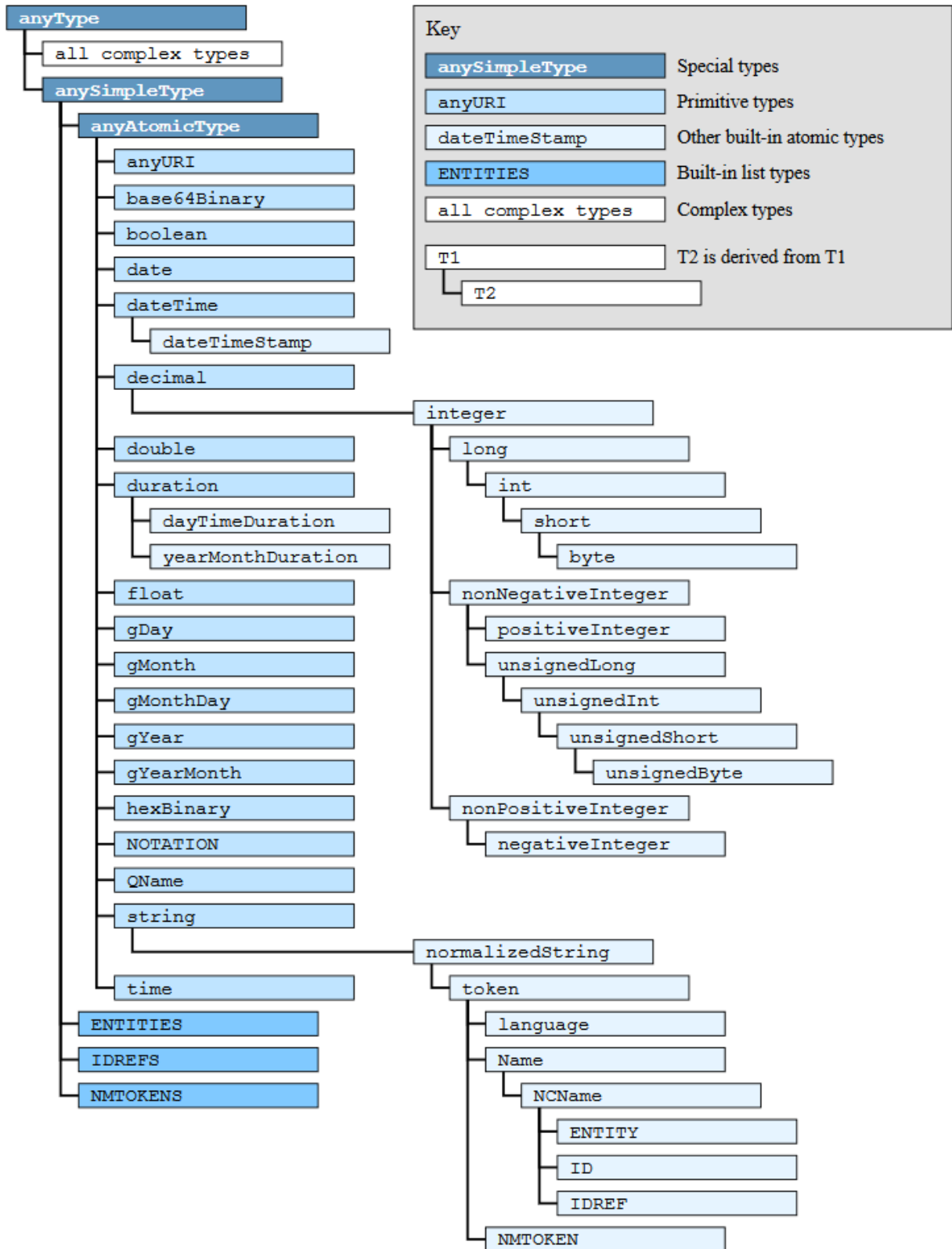
Enumeration:	PrimitiveTypes
xs:gDay	see: https://www.w3.org/TR/xmlschema-2/#gDay
xs:gMonth	see: https://www.w3.org/TR/xmlschema-2/#gMonth
xs:gMonthDay	see: https://www.w3.org/TR/xmlschema-2/#gMonthDay
xs:gYear	see: https://www.w3.org/TR/xmlschema-2/#gYear
xs:gYearMonth	see: https://www.w3.org/TR/xmlschema-2/#gYearMonth
xs:hexBinary	see: https://www.w3.org/TR/xmlschema-2/#hexBinary
xs:string	see: https://www.w3.org/TR/xmlschema-2/#string
xs:time	see: https://www.w3.org/TR/xmlschema-2/#time

Enumeration:	DataTypeDefRdf
Explanation:	Enumeration listing all RDF types
Set of:	--
Literal	Explanation
rdf:langString	String with a language tag

RDF requires IETF BCP 47²³ language tags, i.e. simple two-letter language tags for Locales like “de” conformant to ISO 639-1 are allowed as well as language tags plus extension like “de-DE” for country code, dialect etc. like in “en-US” or “en-GB” for English (United Kingdom) and English (United States). IETF language tags are referencing ISO 639, ISO 3166 and ISO 15924.

²³ see <https://tools.ietf.org/rfc/bcp/bcp47.txt>

Figure 52 Built-In Types of XML Schema Definition 1.1 (XSD)



5.7.13 Cross Constraints and Invariants

5.7.13.1 Introduction

In this clause constraints that cannot be assigned to a single class, i.e. that are no class invariants, are documented.

A class invariant is a constraint that must be true for all instances of a class at any time.

5.7.13.2 Constraints for Referables and Identifiables

Constraint AASd-002: *idShort* of *Referables* shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. I.e. [a-zA-Z][a-zA-Z0-9_]+.

Constraint AASd-117: *idShort* of non-identifiable *Referables* not equal to *SubmodelElementList* shall be specified (i.e. *idShort* is mandatory for all *Referables* except for *SubmodelElementLists* and all *Identifiables*).

Constraint AASd-120: *idShort* of submodel elements within a *SubmodelElementList* shall not be specified.

Constraint AASd-022: *idShort* of non-identifiable referables shall be unique in its namespace.

Constraint AASd-003: *idShort* of *Referables* shall be matched case-sensitive.

5.7.13.3 Constraints for Qualifiers

Constraint AASd-021: Every qualifiable can only have one qualifier with the same *Qualifier/type*.

Constraint AASd-119: If any *Qualifier/kind* value of a *Qualifiable/qualifier* is equal to *TemplateQualifier* and the qualified element inherits from "hasKind" then the qualified element shall be of kind *Template* (*HasKind/kind* = "Template").

5.7.13.4 Constraints for Extensions

Constraint AASd-077: The name of an extension within *HasExtensions* needs to be unique.

5.7.13.5 Constraints for Asset Related Information

Constraint AASd-116: "globalAssetId" (case-insensitive) is a reserved key. If used as value for *SpecificAssetId/name* *SpecificAssetId/value* shall be identical to *AssetInformation/globalAssetId*.

5.7.13.6 Constraints for Types

Constraint AASd-100: An attribute with data type "string" is not allowed to be empty.

6 Predefined Data Specification Templates

6.1 General

A data specification template serves to specify which additional attributes shall be added to an element instance that are not part of the meta model. Typically, data specification templates have a specific scope. For example, templates for concept descriptions differ from templates for operations etc. More than one data specification template can be defined and used for an element instance. Which templates are used for an element instance is defined via *HasDataSpecification*.

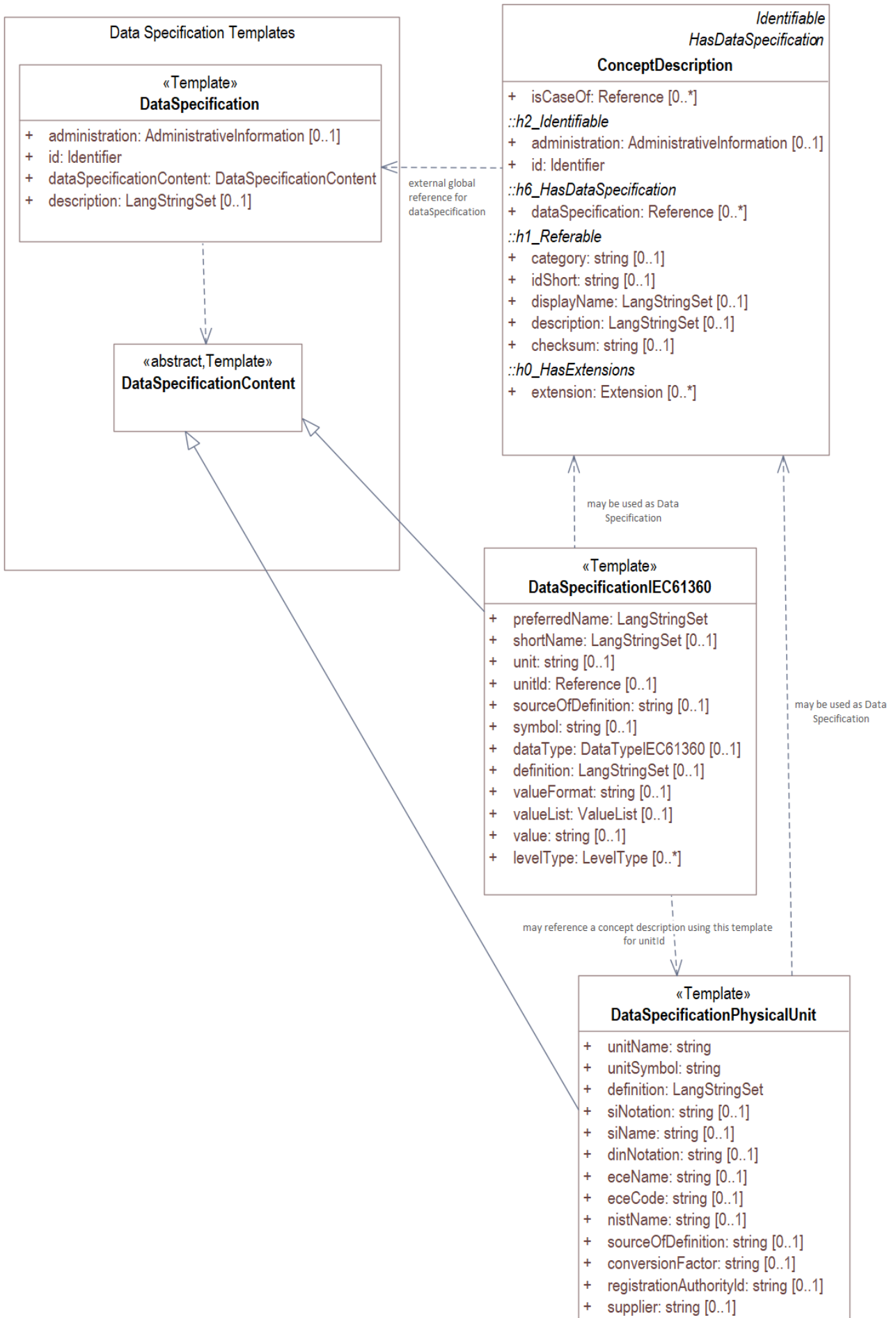
There are two data specification templates defined so far:

- One for defining concept descriptions for properties conformant to IEC61360
- And one for defining concept descriptions for physical units conformant to IEC61360

The template introduced for describing the concept of a property, a value list or a value is based on IEC 61360. Figure 53 shows how concept descriptions and the predefined data specification templates are related to each other.

Note: The abstract classes are numbered h0_, h1_ etc. but Aliases are defined for them without this prefix. The reason for this naming is that in the tooling used for UML modelling (Enterprise Architect) no order for inherited classes can be defined, they are ordered in an alphabetical way.

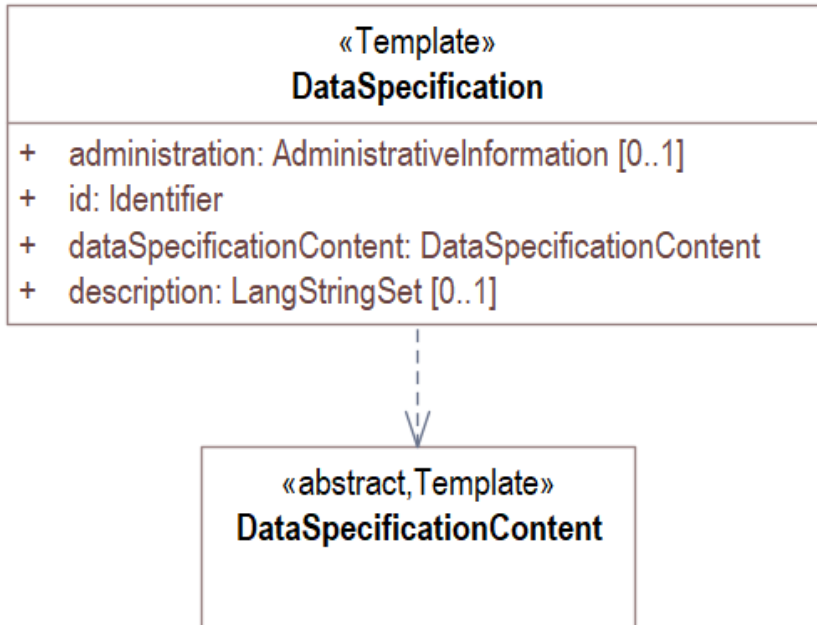
Figure 53 Data Specification Template IEC61360



6.2 Data Specification Template Specification Details: Designators

6.2.1.1 Data Specification Template Attributes

Figure 54 Data Specification Templates



Note: The Data Specification Templates do not belong to the metamodel of the Asset Administration Shell. In serializations that choose specific templates the corresponding data specification content may be directly incorporated.

It is required that a data specification template has a global unique ID so that it can be referenced via *HasDataSpecification/dataSpecification*.

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. In UML these are two separated classes.

Class:	DataSpecification <<Template>>		
Explanation:	Data Specification Template		
Inherits from:	--		
Attribute	Explanation	Type	Card.
administration	Administrative information of an identifiable element. Note: Some of the administrative information like the version number might need to be part of the identification.	AdministrativeInformation	0..1
id	The globally unique identification of the element.	Identifier	1

Class:	DataSpecification <<Template>>		
dataSpecificationContent	The content of the template without meta data	DataSpecificationContent	1
description	Description how and in which context the data specification template is applicable. The description can be provided in several languages.	LangStringSet	0..1

Class:	DataSpecificationContent <<Template>><<abstract>>		
Explanation:	Data specification content is part of a data specification template and defines which additional attributes shall be added to the element instance that references the data specification template and meta information about the template itself.		
Inherits from:	--		
Attribute	Explanation	Type	Card.

6.3 Predefined Template for IEC61360 Properties, Value Lists and Values

6.3.1 General

The data specification template IEC61360 introduces additional attributes to a concept description and defines a property, a value list or a value based on IEC 61360.

Figure 55 to Figure 58 show examples from ECLASS, ECLASS following the IEC 61360 standard to give an impression how it looks like in existing dictionaries.

Figure 55 Example Property from ECLASS

class Data Specification conformant to IEC61360-1 2017-07 and ISO13584-42 2010-12-15	
Property	02-BAA120 Max. rotation speed
short name	-
Format	INTEGER_MEASURE
Unit of measure	1/min
Definition:	Greatest permissible rotation speed with which the motor or feeding unit may be operated
Values:	

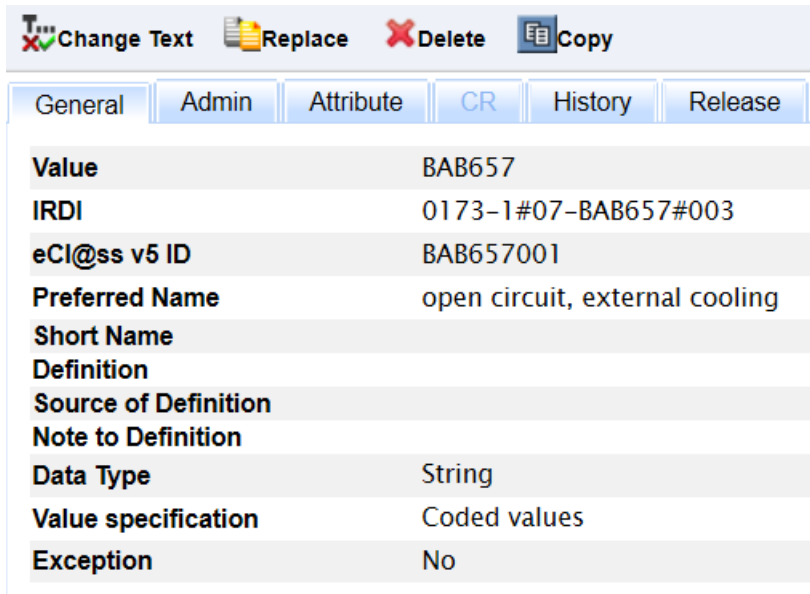
Figure 56 Example Property Description with Value List from ECLASS

Property	02-BAE122 Cooling type
short name	-
Format	STRING
Definition:	Summary of various types of cooling, for use as search criteria that limit a selection
Values:	
	0173-1#07-BAB649#001 - Air-air heat exchanger
	0173-1#07-BAB650#001 - Air-water heat exchanger
	0173-1#07-BAB592#001 - alien
	0173-1#07-BAB611#001 - closed, external air-cooling
	0173-1#07-BAB610#001 - closed, internal air-cooling
	0173-1#07-BAB591#003 - free cooling
	0173-1#07-BAB702#003 - Heat exchanger against other cooling medium
	0173-1#07-BAB657#003 - open circuit, external cooling
	0173-1#07-BAB656#003 - open circuit, internal cooling
	0173-1#07-BAB535#003 - other form of cooling with primary air coolant
	0173-1#07-BAB536#003 - other primary non-air coolant
	0173-1#07-BAB674#003 - self

Figure 57 Example Value Description from ECLASS

Value	0173-1#07-BAB657#003
Classification	open circuit, external cooling
short name	
Definition:	

Figure 58 Example Value Description from ECLASS Advanced



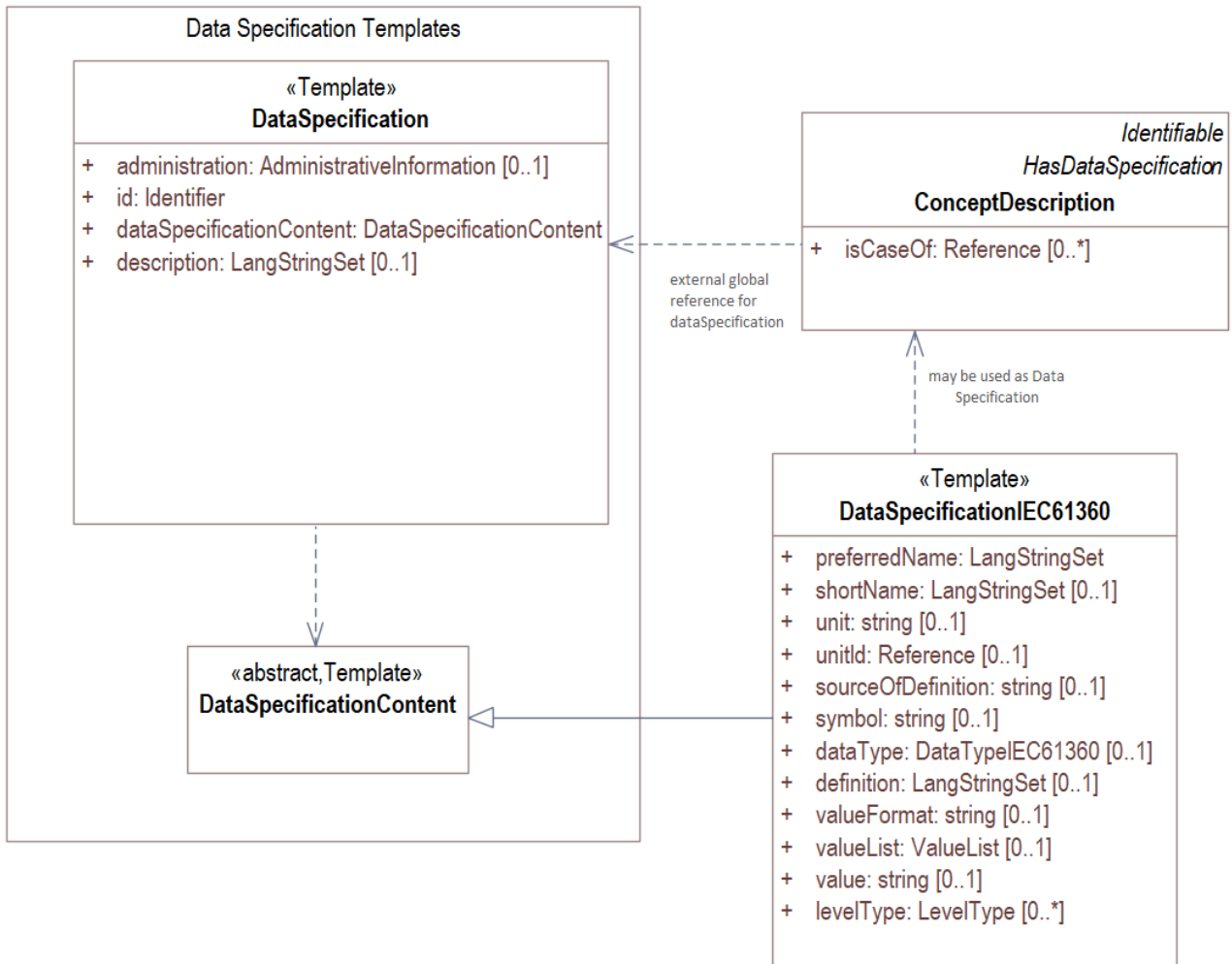
The screenshot shows a software interface with a toolbar at the top containing icons for 'Change Text', 'Replace', 'Delete', and 'Copy'. Below the toolbar is a tabbed menu with 'General', 'Admin', 'Attribute', 'CR', 'History', and 'Release'. The 'CR' tab is active, displaying a table of value descriptions.

Value	BAB657
IRDI	0173-1#07-BAB657#003
eCl@ss v5 ID	BAB657001
Preferred Name	open circuit, external cooling
Short Name	
Definition	
Source of Definition	
Note to Definition	
Data Type	String
Value specification	Coded values
Exception	No

6.3.2 Overview of Data Specification Template IEC61360

Figure 60 shows how a the data specification template IEC61360 is used to describe additional attributes of a concept description.

Figure 59 Concept Descriptions for Properties Conformant to IEC61360

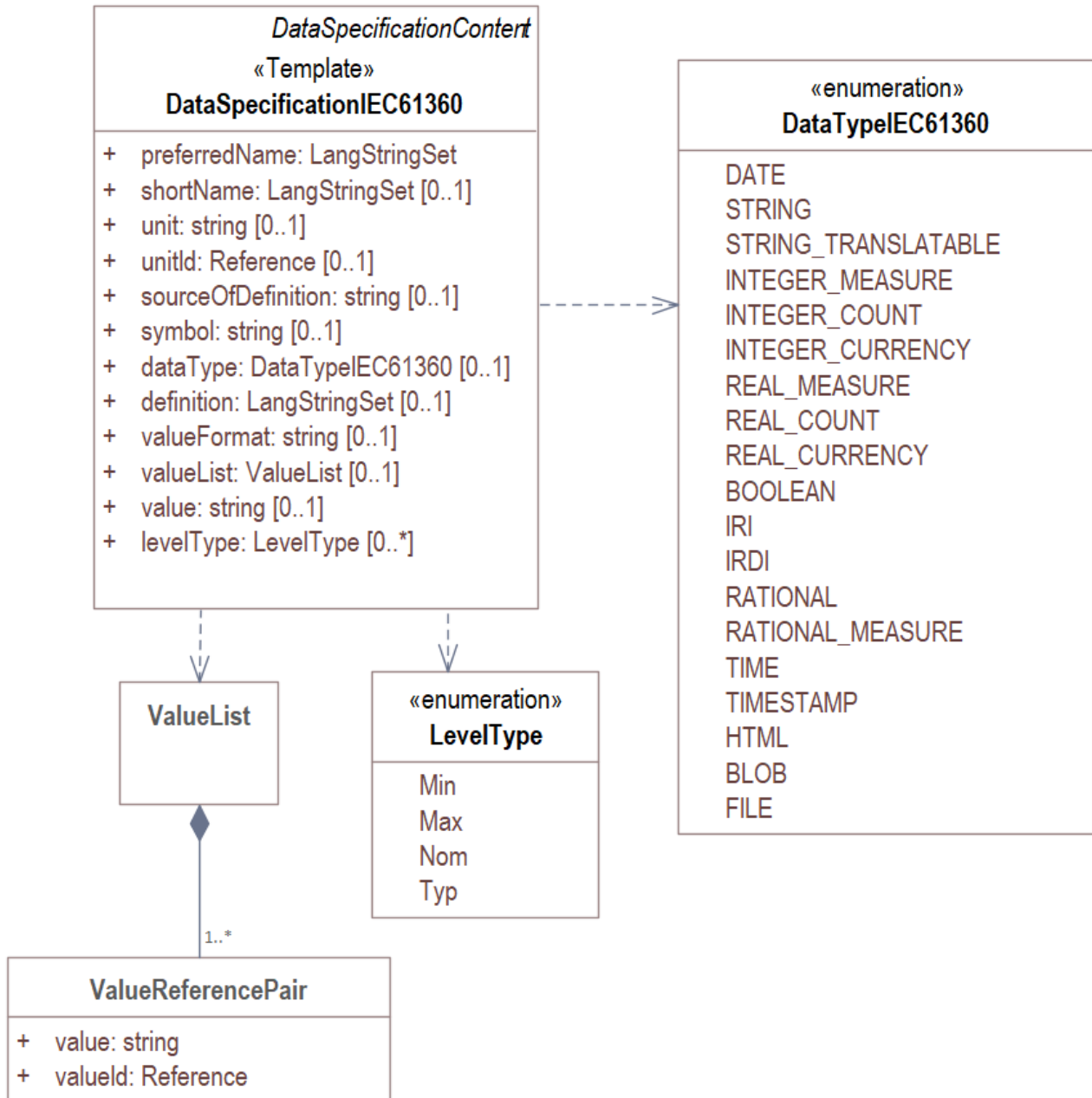


6.3.3 Data Specification IEC61360 Template Specification Details: Designators

6.3.3.1 Data Specification IEC61360 Template Attributes

Although the IEC61360 attributes listed in this template are defined for properties and values and value lists only it is also possible to use the template for other definitions as long as no specific data specifications for concept descriptions for these elements are available. This is shown in the tables Table 8, Table 9, Table 10 and Table 11.

Figure 60 Metamodel of Data Specification IEC6136



Class:	DataSpecificationIEC6136 <<Template>>		
Explanation:	Content of data specification template for concept descriptions for properties, values and value lists conformant to IEC 61360. <u>Constraint AASc-010:</u> If <i>DataSpecificationIEC61360/value</i> is not empty then <i>DataSpecificationIEC61360/valueList</i> shall be empty and vice versa. <u>Constraint AASc-009:</u> If <i>DataSpecificationIEC61360/dataType</i> one of: <i>INTEGER_MEASURE</i> , <i>REAL_MEASURE</i> , <i>RATIONAL_MEASURE</i> , <i>INTEGER_CURRENCY</i> , <i>REAL_CURRENCY</i> , then <i>DataSpecificationIEC61360/unit</i> or <i>DataSpecificationIEC61360/unitId</i> shall be defined.		
Inherits from:	DataSpecificationContent		
Attribute	Explanation	Type	Card.
preferredName	Preferred name	LangStringSet	1

Class:	DataSpecificationIEC61360 <<Template>>		
	<u>Constraint AASc-002:</u> <i>DataSpecification-IEC61360/preferredName</i> shall be provided at least in English.		
shortName	Short name	LangStringSet	0..1
unit	Unit	string	0..1
unitId	<p>Unique unit ID</p> <p>unit and unitId need to be consistent if both attributes are set</p> <p>It is recommended to use a global reference.</p> <p>Although the unitId is a global reference there might exist a <i>ConceptDescription</i> with data specification <i>DataSpecificationPhysicalUnit</i> with the same ID.</p>	Reference	0..1
sourceOf-Definition	Source of definition	string	0..1
symbol	Symbol	string	0..1
dataType	Data Type	DataTypeIEC61360	0..1
definition	Definition in different languages	LangStringSet	0..1
valueFormat	Value Format	string	0..1
valueList	List of allowed values	ValueList	0..1
value	Value	string	0..1
levelType	Set of levels	LevelType	0..1

Note: IEC61360 requires also a globally unique identifier for a concept description. This ID is not part of the data specification template. Instead the *ConceptDescription/id* as inherited via *Identifiable* is used. Same holds for administrative information like the version and revision.

ConceptDescription/idShort and *DataSpecificationIEC61360/shortName* are very similar. However, in this case the decision was to add *shortName* explicitly to the data specification.

Same holds for *ConceptDescription/displayName* and *DataSpecificationIEC61360/displayName*.

Same holds for *ConceptDescription/description* and *DataSpecificationIEC61360/definition*.

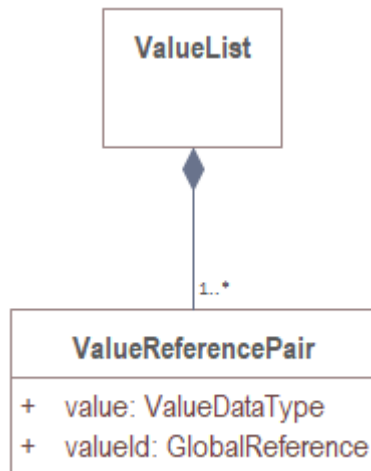
Enumeration:	DataTypeIEC61360
Explanation:	Enumeration of simple data types for a IEC61360 concept description using the data specification template <i>DataSpecificationIEC61360</i>
Set of:	--
Literal	Explanation

DATE	<p>values containing a calendar date, conformant to ISO 8601:2004</p> <p>Format yyyy-mm-dd</p> <p>Example from IEC 61360-1:2017: "1999-05-31" is the [DATE] representation of: "31 May 1999".</p>
STRING	values consisting of sequence of characters but cannot be translated into other languages
STRING_TRANSLATABLE	values containing string but shall be represented as different string in different languages
INTEGER_MEASURE	values containing values that are measure of type INTEGER. In addition such a value comes with a physical unit.
INTEGER_COUNT	values containing values of type INTEGER but are no currencies or measures
INTEGER_CURRENCY	values containing values of type INTEGER that are currencies
REAL_MEASURE	values containing values that are measures of type REAL. In addition such a value comes with a physical unit.
REAL_COUNT	values containing numbers that can be written as a terminating or non-terminating decimal; a rational or irrational number but are no currencies or measures
REAL_CURRENCY	values containing values of type REAL that are currencies
BOOLEAN	values representing truth of logic or Boolean algebra (TRUE, FALSE)
IRI	<p>values containing values of type STRING conformant to Rfc 3987</p> <p>In IEC61360-1 (2017) only URI is supported. An IRI type allows in particular to express an URL or an URI.</p>
IRDI	<p>values conforming to ISO/IEC 11179 series global identifier sequences</p> <p>IRDI can be used instead of the more specific data types ICID or ISO29002_IRDI.</p> <p>ICID values are value conformant to an IRDI, where the delimiter between RAI and ID is "#" while the delimiter between DI and VI is confined to "##"</p> <p>ISO29002_IRDI values are values containing a global identifier that identifies an administrated item in a registry. The structure of this identifier complies with identifier syntax defined in ISO/TS 29002-5. The identifier shall fulfil the requirements specified in ISO/TS 29002-5 for an "international registration data identifier" (IRDI).</p>
RATIONAL	values containing values of type rational
RATIONAL_MEASURE	values containing values of type rational. In addition such a value comes with a physical unit.
TIME	<p>values containing a time, conformant to ISO 8601:2004 but restricted to what is allowed in the corresponding type in xml.</p> <p>Format hh:mm (ECLASS)</p> <p>Example from IEC 61360-1:2017: "13:20:00-05:00" is the [TIME] representation of: 1.20 p.m. for Eastern Standard Time, which is 5 hours behind Coordinated Universal Time (UTC).</p>

TIMESTAMP	values containing a time, conformant to ISO 8601:2004 but restricted to what is allowed in the corresponding type in xml. Format yyyy-mm-dd hh:mm (ECLASS)
FILE	values containing an address to a file. The values are of type URI and can represent an absolute or relative path. IEC61360 does not support the file type.
HTML	Values containing string with any sequence of characters, using the syntax of HTML5 (see W3C Recommendation 28:2014)
BLOB	values containing the content of a file. Values may be binaries. <i>HTML conformant to HTML5</i> is a special blob. In IEC61360 <i>binary</i> is for a sequence of bits, each bit being represented by “0” and “1” only. A binary is a blob but a blob may also contain other source code.

“ValueList” lists all the allowed values for a concept description for which the allowed values are listed in an enumeration. The value list is a set of value reference pairs.

Figure 61 ValueList



Class:	ValueList		
Explanation:	A set of value reference pairs.		
Inherits from:	--		
Attribute	Explanation	Type	Card.
valueReferencePair	A pair of a value together with its global unique ID.	ValueReferencePair	1..*

Class:	ValueReferencePair
Explanation:	A value reference pair within a value list. Each value has a global unique ID defining its semantic.

Class:	ValueReferencePair		
Inherits from:	--		
Attribute	Explanation	Type	Card.
value	the value of the referenced concept definition of the value in <i>valueId</i> .	string	1
valueId	Global unique ID of the value. It is recommended to use a global reference.	Reference	1

A mapping of IEC61360 data types to xsd data types is provided here:
https://wiki.eclass.eu/wiki/Datatype_to_XSD_mapping²⁵:

IEC61360/ECLASS	XSD
Integer (Measure)	integer
Integer (count)	integer
Integer (currency)	integer
Real (measure)	double
String translatable	langString
Rational (measure)	double
Rational (count)	double
Real (count)	double
Real (currency)	double
Time	Time
Date	date
Timestamp	dateTime
Boolean	boolean
Url	anyURI

6.3.3.2 Identifier for DataSpecificationIEC61360

Conformant to the rules in Clause 9.2.4 the following data specification template needs to be referenced via the ID

“<https://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0>”

²⁴ Link not working any longer (access 2022-05-15)

²⁵ Link not working any longer (access 2022-05-15)

(in *hasDataSpecification/dataSpecification*).

This namespace has the qualifier “IEC:” Examples: *IEC:DataSpecificationIEC61360/preferredName* or *IEC:DataSpecificationIEC61360/levelType/Min* or *IEC:LevelType/Min*

Note: The data specification template is not identical to the data specification content as shown in Figure 60.

6.3.4 Category of Concept Descriptions

For the meaning of the content attributes of the IEC61360 data specification template please refer to IEC 61360 and/or ECLASS.

The data specification template can be used to describe properties and values, both.

See Figure 59 for how data specification templates are related to concept descriptions (showing all inherited attributes as well). In a similar way data specification templates for other elements in the information model can be defined and used. In Figure 62 all used categories are listed.

The following tables recommend using specific categories to distinguish which kind of concept is described. They also give advice which attributes need to be filled for which category of concept description.

These tables are not part of the specification because in a way the existing data specification template for describing concept descriptions for properties and coded values is misused to also describe other concepts. In later versions of the standards and this specification, there might be data specifications for concept descriptions better suited for the purpose.

Figure 62 Categories of Concept Descriptions (non normative)

«enumeration» ConceptDescriptionCategories
«enum» APPLICATION_CLASS
«enum» CAPABILITY
«enum» COLLECTION
«enum» DOCUMENT
«enum» ENTITY
«enum» EVENT
«enum» FUNCTION
«enum» PROPERTY
«enum» VALUE
«enum» RANGE
«enum» QUALIFIER_TYPE
«enum» REFERENCE
«enum» RELATIONSHIP

Table 8 IEC61360 Data Specification Template for Properties and Ranges

Attribute ²⁶	Property	Property	Property	Multi-Language-Property	Range
Category of Concept Description	VALUE	PROPERTY	PROPERTY	PROPERTY	PROPERTY
Category of Submodel-Element	CONSTANT	VARIABLE	PARAMETER	--	--
preferredName ²⁷	m	m	m	m	m
shortName	(m)	(m)	(m)	(m)	(m)
unit	(m)	(m)	(m)	--	(m)
unitId	(m)	(m)	(m)	--	(m)
sourceOfDefinition	o	o	o	o	o
symbol	o	o	o	--	--
dataType	m ²⁸	m ²⁹	m ³⁰	stringTranslatable	integer* or real* or rational*
definition	(m)	m	m	m	m
valueFormat	o	o	o	--	o
valueList	--	o	o	--	--
value	m	--	--	--	--
valueId	o	--	--	--	--
levelType	--	--	--	--	{Min, Max}

²⁶ m= mandatory, o = optional, (m) = conditionally mandatory or recommended to be added

²⁷ Mandatory in at least one language. Preferable an English preferred name should always be defined.

²⁸ All data types except stringTranslatable, Iri, Irdi, file, blob, Icid and Iso29002Irdi.

²⁹ All data types except stringTranslatable, Iri, Irdi, file, blob, Icid and Iso29002Irdi.

³⁰ All data types except stringTranslatable, Iri, Irdi, file, blob, Icid and Iso29002Irdi..

Table 9 IEC612360 Data Spec. Template for other Data Elements, Relationships Elements and Capabilities

Attribute ³¹	Reference-Element	File ³²	Blob ³²	Capability ³²	Relationship-Element ³²	AnnotatedRelationship-Element ³²
Category of Concept Description	REFERENCE	DOCUMENT	DOCUMENT	CAPABILITY	RELATIONSHIP	RELATIONSHIP
Category of Submodel-Element	--	--	--	--	--	--
preferredName ³³	m	m	m	m	m	m
shortName	(m)	(m)	(m)	(m)	(m)	(m)
unit	--	--	--	--	--	--
unitId	--	--	--	--	--	--
sourceOf-Definition	o	o	o	o	o	o
symbol	--	--	--	--	--	--
dataType	string or Iri or Irdi or Icid or iso29002 Irdi	file	blob or html5	--	--	--
definition	m	m	m	m	m	m
valueFormat	--	--	--	--	--	--
valueList	--	--	--	--	--	--
value	--	--	--	--	--	--
valueId	--	--	--	--	--	--
levelType	--	--	--	--	--	--

³¹ m= mandatory, o = optional, (m) = conditionally mandatory or recommended to be added

³² Template only used until explicit template for defining the corresponding types of elements are available.

³³ Mandatory in at least one language. Preferable an English preferred name should always be defined.

Table 10 IEC612360 Data Specification Template for other Submodel Elements

Attribute	SubmodelElementList	SubmodelElementCollection	Operation ³²	EventElement	Entity
Category of Concept Description	COLLECTION	ENTITY	FUNCTION	EVENT	ENTITY
Category of Submodel-Element	--	--	--	--	--
preferredName ³⁴	m	m	m	m	m
shortName	(m)	(m)	(m)	(m)	(m)
unit	--	--	--	--	--
unitId	--	--	--	--	--
sourceOfDefinition	o	o	o	o	o
symbol	--	--	--	--	--
dataType	--	--	--	--	--
definition	m	m	m	m	m
valueFormat	--	--	--	--	--
valueList	--	--	--	--	--
value	--	--	--	--	--
valueId	--	--	--	--	--
levelType	--	--	--	--	--

³⁴ Mandatory in at least one language. Preferable an English preferred name should always be defined.

Table 11 Other Elements with semanticId

Attribute ³¹	Submodel	Qualifier
category	APPLICATION_CLASS	QUALIFIER_TYPE
preferredName	m	m
shortName	(m)	(m)
unit	--	--
unitId	--	--
sourceOfDefinition	o	o
symbol	--	--
dataType	--	m
definition	m	m
valueFormat	--	o
valueList	--	o
value	--	--
valueId	--	--
levelType	--	--

6.4 Predefined Templates for Unit Concept Descriptions

6.4.1 General

The data specification template IEC61360 introduces additional attributes to a concept description of a physical unit and is based on IEC 61360.

Figure 63 shows an example from ECLASS, ECLASS following the IEC 61360 standard to give an impression how it looks like in existing dictionaries.

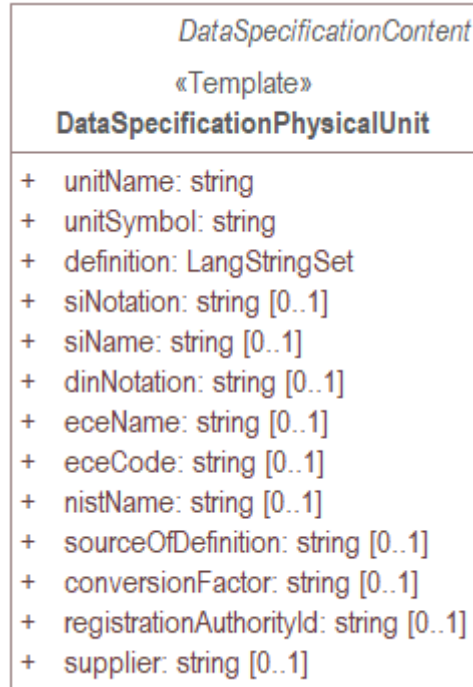
Figure 63 Example of a concept description for a unit: 1/min (from ECLASS)

class Data Specification Units conformant to IEC61360-1 and ISO13584-42 (NEW)	
ID	0173-1#05-AAA650#002
Name	1/min
Primary language	German
Structured Naming	min ⁻¹
Short Name	1/min
Definition	reciprocal of the unit minute
Source	NIST Special Publication 811:1995
Comment	
SI Notation	1/min
SI-Name	reciprocal minute
DIN Notation	min ⁻¹
ECE Name	reciprocal minute
ECE-Code	C94
NIST-Name	revolutions per minute
Conversion factor	1.0/60
Registration authority ID	0173-1
Supplier	ECL

6.4.2 Data Specification Physical Unit Template Specification Details: Designators

6.4.2.1 Data Specification Template Physical Unit Attributes

Figure 64 Metamodel of Data Specification Physical Unit



Class:	DataSpecificationPhysicalUnit <<Template>>		
Explanation:	Content of data specification template for concept descriptions for physical units conformant to IEC 61360.		
Inherits from:	DataSpecificationContent		
Attribute	Explanation	Type	Card.
unitName	Name of the physical unit	string	1
unitSymbol	Symbol for the physical unit	string	1
definition	Definition in different languages	LangStringSet	1
siNotation	Notation of SI physical unit	string	0..1
siName	Name of SI physical unit	string	0..1
dinNotation	Notation of physical unit conformant to DIN	string	0..1
eceName	Name of physical unit conformant to ECE	string	0..1
eceCode	Code of physical unit conformant to ECE	string	0..1
nistName	Name of NIST physical unit	string	0..1
sourceOfDefinition	Source of definition	string	0..1
conversionFactor	Conversion factor	string	0..1
registrationAuthorityId	Registration authority ID	string	0..1

Class:	DataSpecificationPhysicalUnit <<Template>>		
supplier	Supplier	string	0..1

6.4.2.2 Identifier for Data Specification Physical Unit

Conformant to the rules in Clause 9.2.4 the following data specification template should be referenced via the ID

`"https://admin-shell.io/DataSpecificationTemplates/DataSpecificationPhysicalUnit/3/0/RC02"`

(in *hasDataSpecification/dataSpecification*).

The recommendation is to use "IEC:" as namespace qualifier as already discussed in Clause 9.2.4.

Examples: *IEC:DataSpecificationPhysicalUnit/unitName* or *IEC:DataSpecificationPhysicalUnit/definition*

Units are used in data specification templates for properties when defining the *unitId* (*IEC:/DataSpecificationIEC61250/unitId*, see Clause 6.3.3). The unit value corresponds then to the *unitName* as specified in the concept description referenced via *unitId*.

The data specification template for concept descriptions for units (see Clause 6.4) is defined conformant to IEC61360-1 and ISO13854-42 and is following the xml schema UnitML. An example unit is shown in Figure 63.

6.5 Cross Constraints and Invariants for Predefined Data Specifications

6.5.1 General

In this clause constraints in the context of the predefined data specifications that cannot be assigned to a single class, i.e. that are no class invariants, are documented.

A class invariant is a constraint that must be true for all instances of a class at any time.

6.5.2 Constraints for DataSpecificationIEC61360

Constraint AASc-004: For a *ConceptDescription* with *category* *PROPERTY* or *VALUE* using data specification template IEC61360 (<http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/3/0/RC02>) - *DataSpecificationIEC61360/dataType* is mandatory and shall be one of: *DATE*, *STRING*, *STRING_TRANSLATABLE*, *INTEGER_MEASURE*, *INTEGER_COUNT*, *INTEGER_CURRENCY*, *REAL_MEASURE*, *REAL_COUNT*, *REAL_CURRENCY*, *BOOLEAN*, *RATIONAL*, *RATIONAL_MEASURE*, *TIME*, *TIMESTAMP*.

Constraint AASc-005: For a *ConceptDescription* with *category* *REFERENCE* using data specification template IEC61360 (<http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/3/0/RC02>) - *DataSpecificationIEC61360/dataType* shall be one of: *STRING*, *IRI*, *IRDI*.

Constraint AASc-006: For a *ConceptDescription* with *category* *DOCUMENT* using data specification template IEC61360 (<http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/3/0/RC02>) - *DataSpecificationIEC61360/dataType* shall be one of: *FILE*, *BLOB*, *HTML*.

Constraint AASc-007: For a *ConceptDescription* with *category* *QUALIFIER_TYPE* using data specification template IEC61360 (<http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/3/0/RC02>) - *DataSpecificationIEC61360/dataType* is mandatory and shall be defined.

Constraint AASc-008: For a *ConceptDescription* except for a *ConceptDescription* of category *VALUE* using data specification template IEC61360 (<http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/3/0/RC02>) - *DataSpecificationIEC61360/definition* is mandatory and shall be defined at least in English.

Constraint AASc-003: For a *ConceptDescription* with category VALUE using data specification template IEC61360 (<http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0>) *DataSpecificationIEC61360/value* shall be set.

Constraint AASd-050: If the *DataSpecificationContent DataSpecificationIEC61360* is used for an element then the value of *HasDataSpecification/dataSpecification* shall contain the global reference to the IRI of the corresponding data specification template <https://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/3/0/RC02>.

6.5.3 Constraints for DataSpecificationPhysicalUnit

Constraint AASd-050b: If the *DataSpecificationContent DataSpecificationPhysicalUnit* is used for an element then the value of *HasDataSpecification/dataSpecification* shall contain the global reference to the IRI of the corresponding data specification template <https://admin-shell.io/DataSpecificationTemplates/DataSpecificationPhysicalUnit0/3/0/RC02>.

7.1 General

As the AAS is a central point for data access, there is the need to support fine grained access control that supports multiple roles as well as separate access control policies for individual nodes or submodels in the AAS. Access Control is based on Identity Management and can only be successfully implemented in a secure environment. These aspects as well as concepts to support data usage control and data provenance tracking are going to be developed further in the future, hence not described in detail in this chapter. For this document, the focus lies on the supported access control model.

7.2 Passing Access Permissions

When having a look at the leading picture (Figure 2 in Clause 4.2) also security aspects have to be considered when transferring information from one value chain partner to the next.

When AAS content is passed from one partner to another, this is typically related to a change in the access control domain of the partners involved (supplier, integrator, operator), i.e. the scope of the validity of access control policies.

Therefore, for the example that the supplier passes on data to the integrator, the following typical steps are carried out:

- Step A1-A2: The supplier makes a choice which data is to be passed on (see Clause 10), and thus determines the content of the AASX package (see Clause 8).
- Step A2-A3: The AASX package is transferred to the integrator.
- Step A3-A4: The integrator receives the package and imports the content into his security domain. During this step, the integrator has to establish access rights according to the requirements in his own security domain.

ABAC is a very flexible approach, that also encompasses role-based access as a role can be considered as one attribute in this context. Other attributes might be the time-of-day, the location of the asset, the originating address and others.

In addition to the AAS content itself, also the defined access permissions have to be transferred between the partners due to the following two reasons:

- (1) Access permissions to information elements of an AAS must be established in each access control domain.
- (2) One partner must be able to pass a suggestion which access permissions should be established for the asset that is described in the AAS.

An example for the requirement (2):

A robot manufacturer suggests that for the robot the following roles shall be defined: machine setter, operator and a maintenance role. Note that the roles have to be expressed by means of attributes of the AAS representing the robot. He also suggests permissions for these roles, e.g. an installer (integrator) does have write-access to the program of the robot, but an operator does not.

The above example motivates that the access permission rules need to be passed from one access control domain to the other. The passing on of the access permission rules is implemented by following means:

- Definition of access permissions: The detailed access permissions (e.g. read, write, delete, create, invoke method etc.) are defined in a domain specific submodel (see *AccessControl/defaultPermissions* and *AccessControl/selectablePermissions* in Clause 7.4.4).
- Definition of the access permission rules, based on the defined access permissions. These are defined as part of access control (see Clause 7.4.4).

- Association of access permission rules to each information element (object) of the AAS. This means is realized by the information structure of the AAS itself (see *PermissionsPerObject* in Clause 7.4.5).

Effective access permissions are determined based on the access permission rules. Each submodel element in the AAS shall have rules that define its access permissions for each subject. The subject is assumed to be already authenticated.

If a submodel element does not have these rules, it will automatically use the table for the element where it is included ("inheritance from above"). The most upper object is the AAS itself, i.e. the AAS is the starting point for the inheritance.

As indicated before, subject identification, rule definitions and also permissions could be different for the receiving party as it may be in a different access control domain. When the receiving party establishes access permissions during step A3-A4, it must merge the passed-on access definitions (permissions and access permission rules) to the existing definitions in its access control domain.

In [19] examples and more background information on attribute access control and access control in general can be found. The classes and their attributes are defined in Clause 7.4.

7.3 Overview Metamodel of Administration Shell w.r.t. Security

Security-related attributes related to access control are part of the AAS information model. The objective of access control is to protect system resources (here: AAS content) against unauthorized access. The protection measures are specified in access control policies whose scope of validity is defined by security domains dedicated to access control.

Note: The implementation of access control in an I4.0 System needs support by dedicated infrastructure services, e.g. for identity management, digital certificate management, authentication and access control enforcement.

The underlying concept applied for access control is the concept of attribute-based access control (ABAC). In general, the ABAC request flow is described in [22]. Originally, ABAC relies upon the data-flow model and language model of the OASIS eXtensible Access Control Markup Language (XACML) specifications [54].

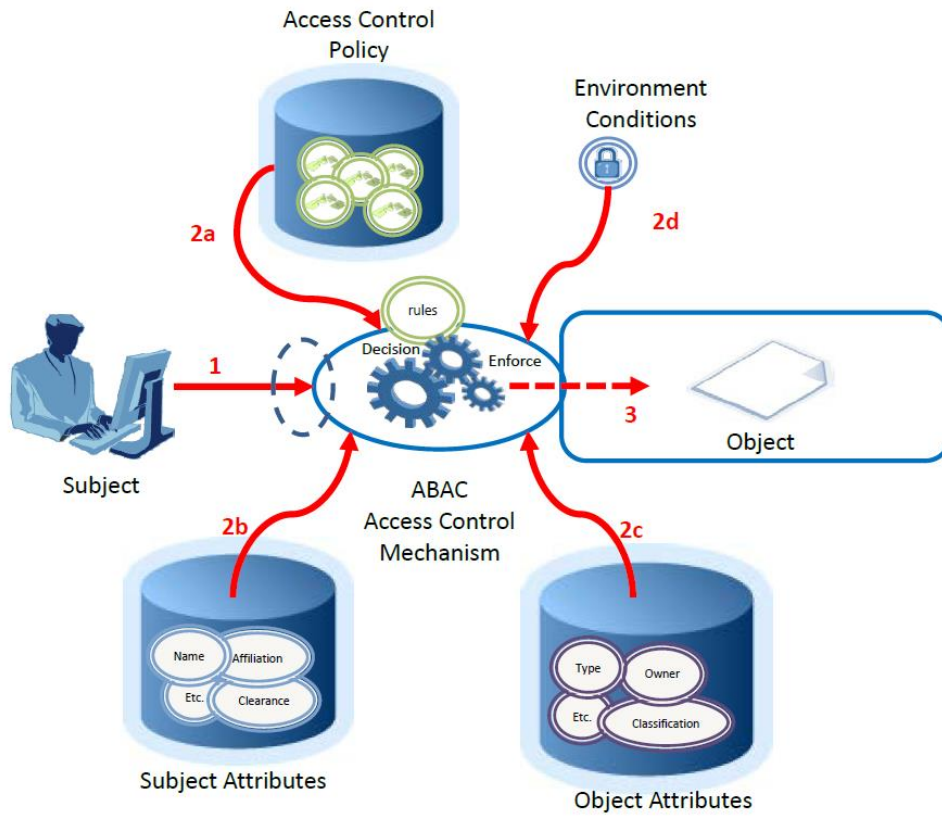
OASIS XACML includes concepts such as:

- Policy administration point (PAP): The system entity that creates a policy set.
- Policy decision point (PDP): The system entity that evaluates an applicable policy and renders an authorization decision.
- Policy enforcement point (PEP): The system entity that performs access control, by making decision requests and enforcing authorization decisions.
- Policy information point (PIP): The system entity that acts as a source of attribute values.

The general request flow is depicted in Figure 65:

- A subject is requesting access to an object (1). In the context of an AAS, an object is typically a submodel or a property or any other submodel element connected to the asset.
- The implemented access control mechanism of the AAS evaluates the access permission rules (2a) that include constraints that need to be fulfilled w.r.t. the subject attributes (2b), the object attributes (2c) and the environment conditions (2d).
- After the evaluation a decision is taken and enforced upon the object (3), i.e. the access to the submodel element is permitted or declined.

Figure 65 Attribute Based Access Control [22]



Note: Attribute in the context of ABAC is different from attributes of elements as defined in the metamodel.

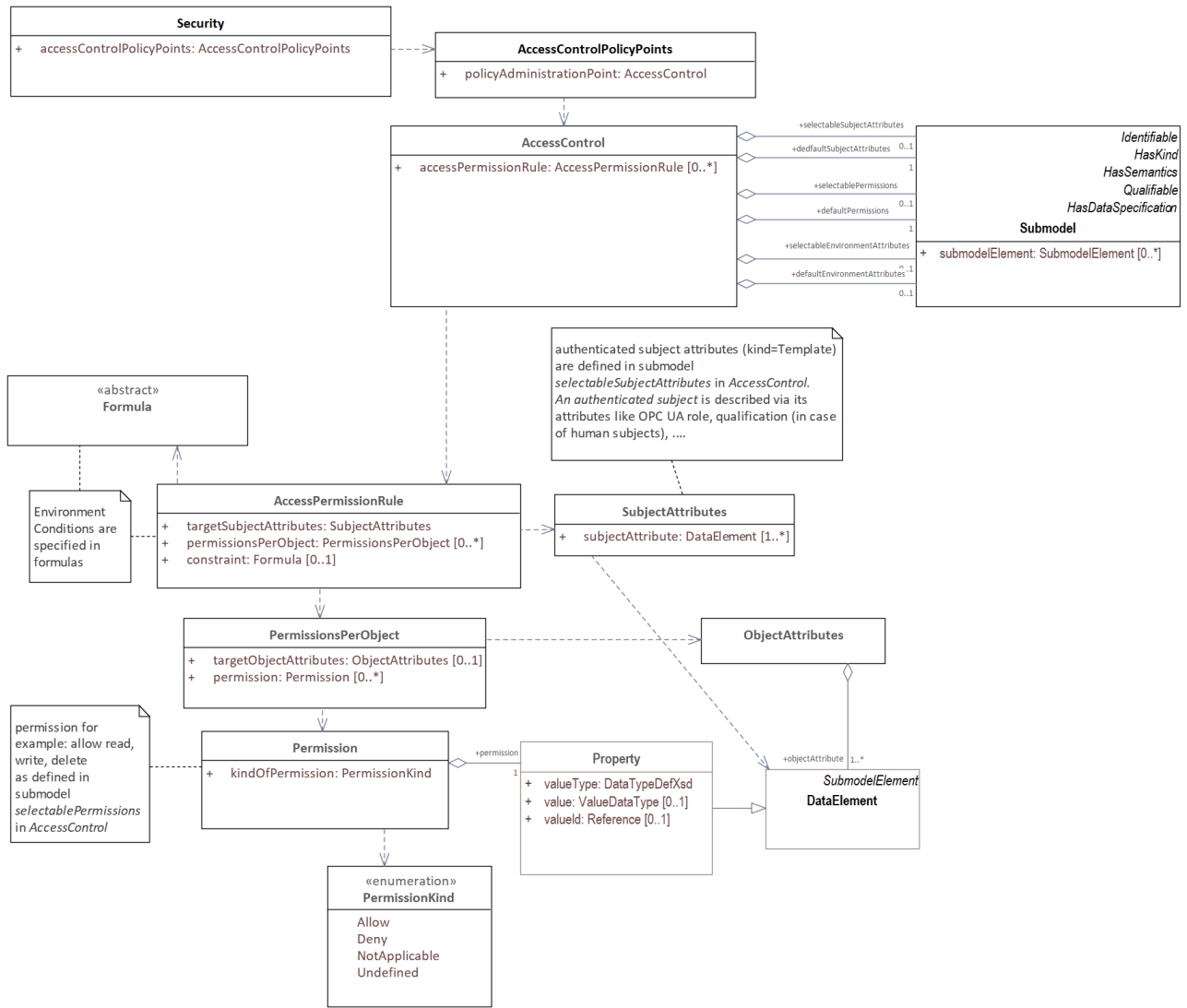
In Figure 66 an overview of the information model of the AAS w.r.t. security aspects is given.

An object in the context of ABAC corresponds typically to a submodel or to a submodel element. The object attributes again are modelled as submodel elements.

Subject Attributes need to be accessed either via an external policy information point (PIP) or they are defined as properties within a special submodel of the AAS. A typical subject attribute is its role. The role is the only subject attribute defined when ABAC is applied as role-based access control.

Optionally, environment conditions can be defined. In role-based access control, no environment conditions are defined. Environment conditions can be expressed via formula constraints. To be able to do so the values needed should be defined as property or reference to data within a submodel of the AAS.

Figure 66 Metamodel Overview for Access Control



By means of access control policies (e.g. in terms of access permission rules), it is defined which subject is allowed to access which objects³⁵ within the AAS. It is assumed that the subject is already authenticated. Objects can be any referable elements, i.e. they include identifiables like submodels and concept descriptions. More general it can be specified whether an authenticated subject is allowed or denied accessing an object a.s.o. “Access” might be one of the specified permissions on an element of the AAS. Which permissions are selectable is not defined by the metamodel of the AAS. The selectable permissions are defined via a submodel (*AccessControl/selectablePermissions*). The same holds for the subject attributes (*AccessControl/selectableSubjectAttributes*). The default subject attributes and default permissions are used if they are not overwritten by the owner of the AAS. As for permissions the used authenticated subject attributes are defined in submodel *AccessControl/selectableSubjectAttributes*.

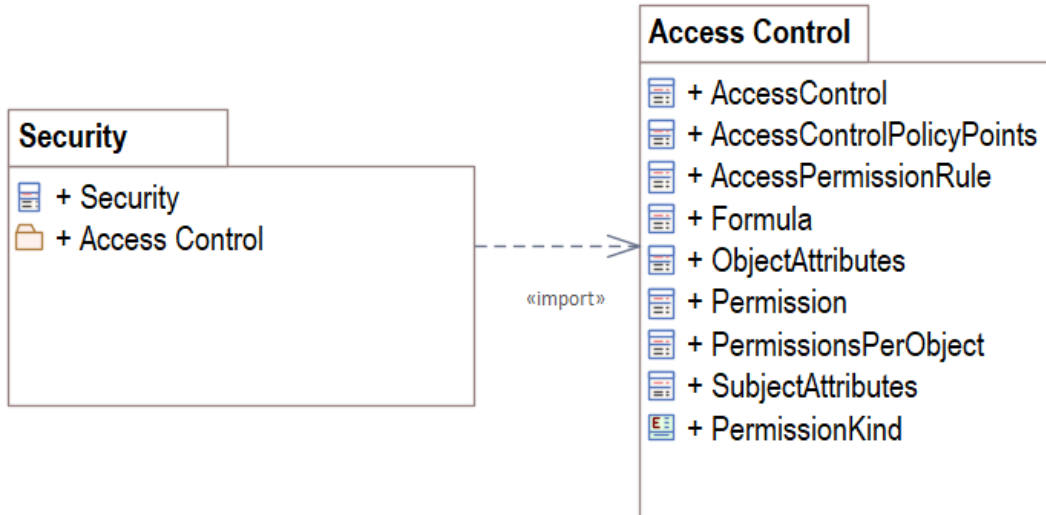
Access rights may be constrained further. For example, a policy rule may specify that the role “maintenance engineer” (to be more precise: an authenticated subject with subject attribute “role = ‘maintenance engineer’”) is only allowed to write configuration parameters if the machine (the asset) is not running. See Figure 73 in Clause 5.7.2.7 for a formal expression of this access rule based on the property “Status”.

³⁵ The term “object” is used because it is more generic and in future also other objects like for example attributes of classes may be included besides elements.

Object Attributes are handled in a different way. It is assumed that any property of the object in focus can additionally take over the role of an object attribute. Therefore, there is no special submodel for default or selectable object attributes.

Figure 67 gives an overview of all elements defined for security issues in the metamodel.

Figure 67 Security Overview Packages



7.4 Metamodel Specification Details: Designators

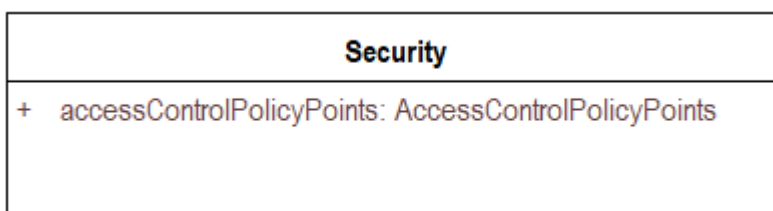
7.4.1 Introduction

In this clause the classes of the metamodel related to security are specified in detail. It is an extension of the metamodel as described in Clause 5.7.

For understanding the extension the basics and common abstract classes need to be understood (see especially Clause 5.7.2, Clause 5.7.9 and Clause 5.7.10.4).

7.4.2 Security Attributes

Figure 68 Metamodel of Security Attributes of AAS



Class:	Security		
Explanation:	Container for security relevant information of AAS.		
Inherits from:	--		
Attribute	Explanation	Type	Card.
accessControlPolicyPoints	Access control policy points of AAS.	AccessControlPolicyPoints	1

7.4.3 Access Control Policy Point Attributes

Figure 69 Metamodel of Access Control Policy Points

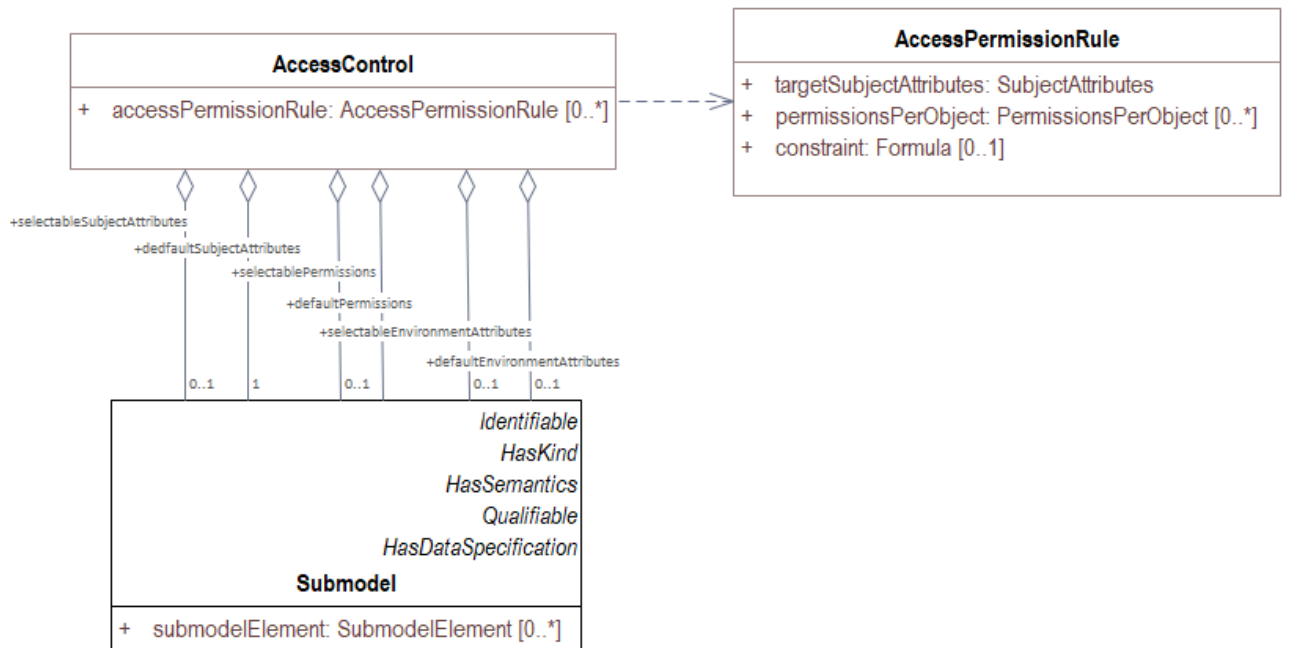


Class:	AccessControlPolicyPoints		
Explanation:	Container for access control policy points.		
Inherits from:	--		
Attribute	Explanation	Type	Card.
policyAdministrationPoint	The access control administration policy point of AAS.	AccessControl	1

The definition of the Policy Administration point is taken from [22]. The PAP is responsible for managing administering policies and also includes access control for the policies itself. Policies are deployed to the PDP for evaluation of access control decisions.

7.4.4 Access Control Attributes

Figure 70 Metamodel of Access Control

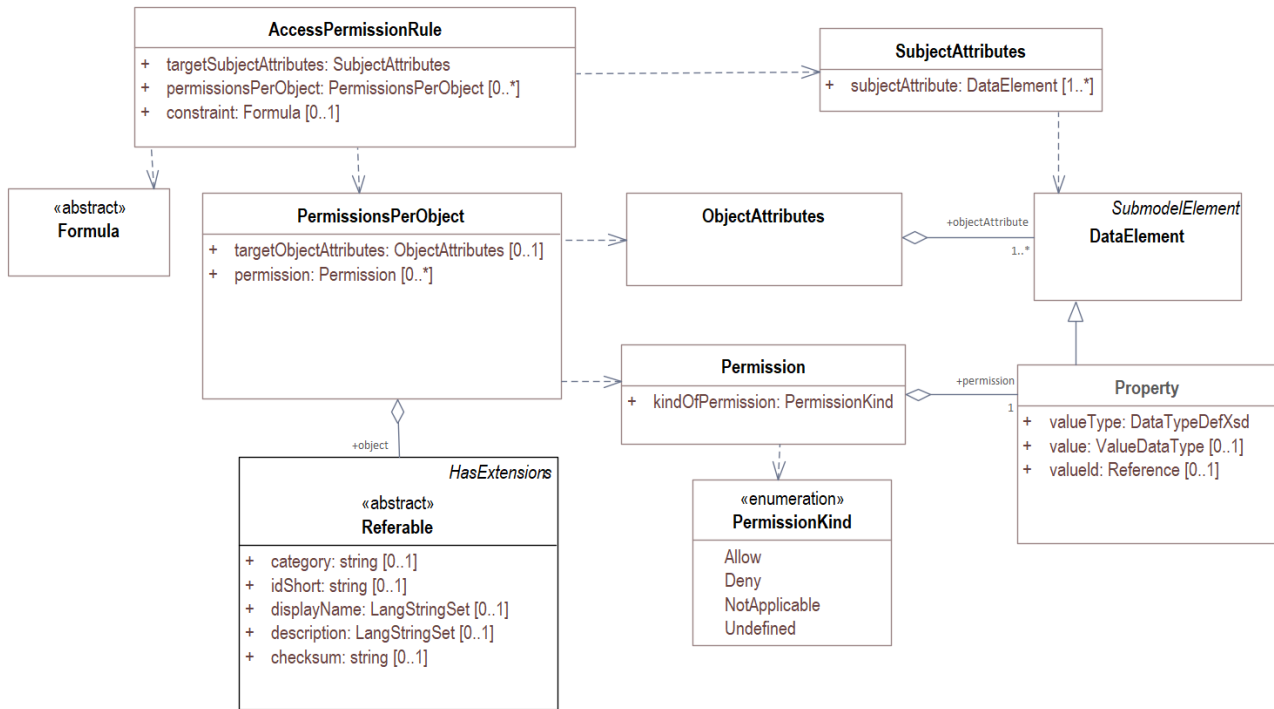


Class:	AccessControl		
Explanation:	Access Control defines the local access control policy administration point. Access Control has the major task to define the access permission rules.		
Inherits from:	--		
Attribute	Explanation	Type	Card.
accessPermissionRule	Access permission rules of the AAS describing the rights assigned to (already authenticated) subjects to access elements of the AAS.	AccessPermissionRule	0..*
selectableSubjectAttributes	Reference to a submodel defining the authenticated subjects that are configured for the AAS. They are selectable by the access permission rules to assign permissions to the subjects. Default: reference to the submodel referenced via <i>defaultSubjectAttributes</i> .	ModelReference<Submodel>	0..1
defaultSubjectAttributes	Reference to a submodel defining the default subjects' attributes for the AAS that can be used to describe access permission rules. The submodel is of <i>kind=Template</i> .	ModelReference<Submodel>	1

Class:	AccessControl		
selectablePermissions	Reference to a submodel defining which permissions can be assigned to the subjects. <u>Default:</u> reference to the submodel referenced via <i>defaultPermissions</i>	ModelReference<Submodel>	0..1
defaultPermissions	Reference to a submodel defining the default permissions for the AAS.	ModelReference<Submodel>	1
selectableEnvironmentAttributes	Reference to a submodel defining which environment attributes can be accessed via the permission rules defined for the AAS, i.e. attributes that are not describing the asset itself. <u>Default:</u> reference to the submodel referenced via <i>defaultEnvironmentAttributes</i>	ModelReference<Submodel>	0..1
defaultEnvironmentAttributes	Reference to a submodel defining default environment attributes, i.e. attributes that are not describing the asset itself. The submodel is of <i>kind=Template</i> . At the same type the values of these environment attributes need to be accessible when evaluating the access permission rules. This is realized as a policy information point.	ModelReference<Submodel>	0..1

7.4.5 Access Permission Rule Attributes

Figure 71 Metamodel of Access Permission Rule



Class:	AccessPermissionRule		
Explanation:	Table that defines access permissions per authenticated subject for a set of objects (referable elements).		
Inherits from:			
Attribute	Explanation	Type	Card.
targetSubjectAttributes	Target subject attributes that need to be fulfilled by the accessing subject to get the permissions defined by this rule.	SubjectAttributes	1
permissionsPerObject	Set of object-permission pairs that define the permissions per object within the access permission rule.	PermissionsPerObject	0..*
constraint	Constraint that needs to be validated to true so that access permission rule holds.	Formula	0..1

Class:	PermissionsPerObject		
Explanation:	Table that defines access permissions for a specified object. The object is any referable element in the AAS. Additionally, object attributes can be defined that further specify the kind of object the permissions apply to.		
Inherits from:	--		
Attribute	Explanation	Type	Card.

Class:	PermissionsPerObject		
object	Element to which permission shall be assigned.	ModelReference<Referable>	1
targetObjectAttributes	Target object attributes that need to be fulfilled so that the access permissions apply to the accessing subject.	ObjectAttributes	0..1
permission	Permissions assigned to the object. The permissions hold for all subjects as specified in the access permission rule.	Permission	0..*

Class:	ObjectAttributes		
Explanation:	A set of data elements that describe object attributes. These attributes need to refer to a data element within an existing submodel.		
Inherits from:	--		
Attribute	Explanation	Type	Card.
objectAttribute	Reference to a data element that further classifies an object.	ModelReference<DataElement>	1..*

Class:	Permission		
Explanation:	Description of a single permission.		
Inherits from:	--		
Attribute	Explanation	Type	Card.
permission	Reference to a property that defines the semantics of the permission.	ModelReference<Property>	1
kindOfPermission	Description of the kind of permission. Possible kind of permission also include the denial of the permission. Values: <ul style="list-style-type: none"> • Allow • Deny • NotApplicable • Undefined 	PermissionKind	1

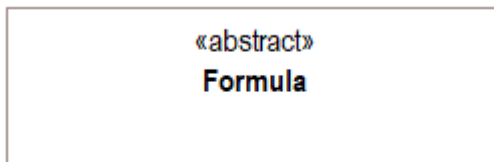
Class:	SubjectAttributes		
Explanation:	A set of data elements that further classifies a specific subject.		
Inherits from:	--		

Class:	SubjectAttributes		
Attribute	Explanation	Type	Card.
subjectAttribute	A data element that further classifies a specific subject.	DataElement	1..*

Enumeration:	PermissionKind
Explanation:	Enumeration of the kind of permissions that is given to the assignment of a permission to a subject.
Set of:	--
Literal	Explanation
Allow	Allow the permission given to the subject.
Deny	Explicitly deny the permission given to the subject.
NotApplicable	The permission is not applicable to the subject.
Undefined	It is undefined whether the permission is allowed, not applicable or denied to the subject.

7.4.6 Formula Attributes

Figure 72 Metamodel of Formulas



A formula may depend on referables that are used in the logical expression.

The value of the referenced elements needs to be accessible so that it can be evaluated in the formula to true or false in the corresponding logical expression it is used in.

In Figure 73 an example for a formula depending on the property “Status” is shown. However, up to now no formula language is defined for the AAS so the example is just showing one of the possibilities using a xsd mixed-Content Type and an attribute “depends” to describe which property values are needed to formulate the rule.

Note: With this exemplary mechanism it is not possible so far to formulate formulas containing complex objects (e.g. submodel element collections or relationship elements). It is restricted to data elements or other elements for which there is a serialization as a string available and defined.

Figure 73 Example Formula “Machine Status not Running” (non normative)

```

<aas:Formula>
  <aas:dependsOn>
    <Keys>
      <Key type="Submodel">https://myShell/Machine</Key>
      <Key type="Property">Status</Key> </Keys>
    </aas:dependsOn> != RUNNING
  </aas:Formula>

```

Class:	Formula <<abstract>>		
Explanation:	A formula is used to describe constraints by a logical expression.		
Inherits from:			
Attribute	Explanation	Type	Card.

7.4.7 Cross Constraints and Invariants

In this clause constraints that cannot be assigned to a single class, i.e. that are no class invariants, are documented.

A class invariant is a constraint that must be true for all instances of a class at any time.

Constraint AASs-010: The property referenced in *Permission/permission* shall have the category “CONSTANT”.

Constraint AASs-011: The property referenced in *Permission/permission* shall be part of the submodel that is referenced within the “*selectablePermissions*” attribute of “*AccessControl*”.

Constraint AASs-015: Every data element in *SubjectAttributes/subjectAttributes* shall be part of the submodel that is referenced within the “*selectableSubjectAttributes*” attribute of “*AccessControl*”.

8 Package File Format for the Asset Administration Shell (AASX)

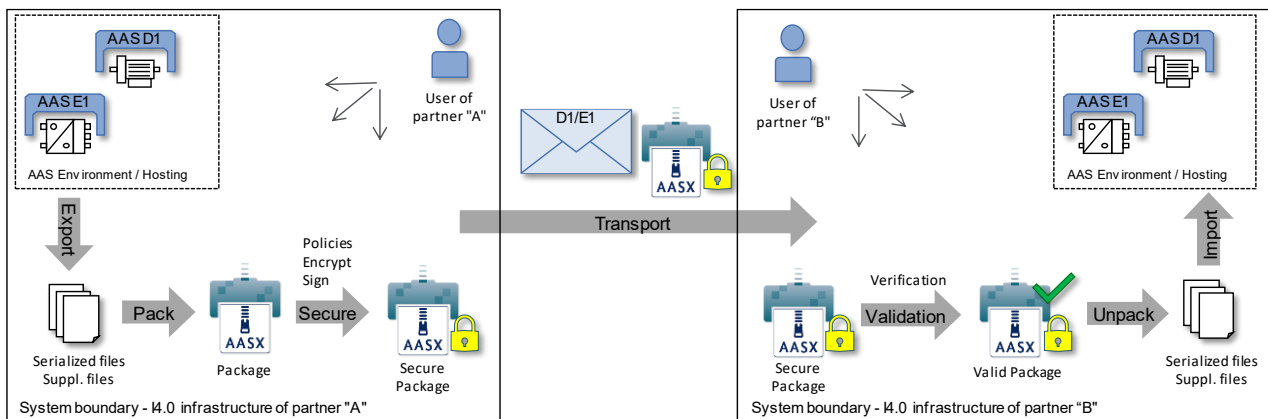
8.1 General

In some use cases it is necessary to exchange the full or partial structure of the Asset Administration Shell with or without associated values and/or make the information persistent (e.g. store it in a file server). This would mean that it is necessary to define a file format that can hold and store this information. Therefore, a package file format for the Asset Administration Shell (AASX) is defined based on the following requirements:

- Generic package file format to include the Asset Administration Shell structure, data and other related files
- Main use cases are the exchange between organizations/partners and storage/persistence of the Asset Administration Shells' information.
- Without any legal restriction and no royalties. Preferably based on an international standard with high guarantees of future maintainability of that format
- Existence of APIs to create, read and write this format
- Digital signatures & encryption capabilities must be provided
- Policies for authenticity and integration of package files³⁶

The following process in Figure 74 is defined for creating and consuming AASX packages.

Figure 74 Process for generating and consuming AASX packages



The process starts by serializing the existing AAS (e.g. D1 and E1) into files (according to the serialization mechanisms described in this document), as well as exporting other supplementary files (which are files mentioned in the structure of the AAS, such as manuals, CAD files, etc.). All of these files will be packaged together into the AASX ZIP file format and will be followed by several security steps that defines the policies for modifiability, encryption and digitally signing of the files inside the AASX. The final AASX can then be transported from the AASX producer (in this case partner A) to the AASX consumer (partner B), by digital media such as e-mail, USB-Sticks, etc. The consumer needs first to validate and verify the incoming AASX, unpack the contained files and then import them to generate the new AAS in the consumer environment. The process will be explained in detail in the following sub-sections.

8.2 Basic Concepts of the Open Packaging Conventions

The packaging model specified by the Open Packaging Conventions describes **packages**, **parts**, and **relationships**. Packages hold parts, which hold content and resources, such as **files**³⁷. Every file in a package

³⁶ Role-based policies to access this package is not defined, as this is a feature of the systems that host the AASs (see section 7)

³⁷ The term "file" will be used instead of "part".

has a unique URI-compliant file name along with a specified content-type expressed in the form of a MIME media type.

Relationships are defined to connect the package to files, and to connect various files in the package. The definition of the relationships (along with the files' names) is the **logical model** of the package. The resource that is a source of a relationship must be either the package itself or a data component (file) inside of the package. The target resource of a relationship can be any URI-addressable resource inside or outside of the package. It is possible to have more than one relationship that share the same target file (see example 9–6 in ISO/IEC 29500-2: 2012).

The **physical model** maps these logical concepts to a physical format. The result of this mapping is a physical package format (a ZIP archive format) in which files appear in a directory-like hierarchy (adapted from [27] and [28]).

8.3 Conventions for the Asset Administration Shell Package File Format (AASX)

The Asset Administration Shell Package (AASX) format derives from the Open Package Conventions standards, consequently inheriting its characteristics. Nevertheless, some conventions shall be defined for the AASX:

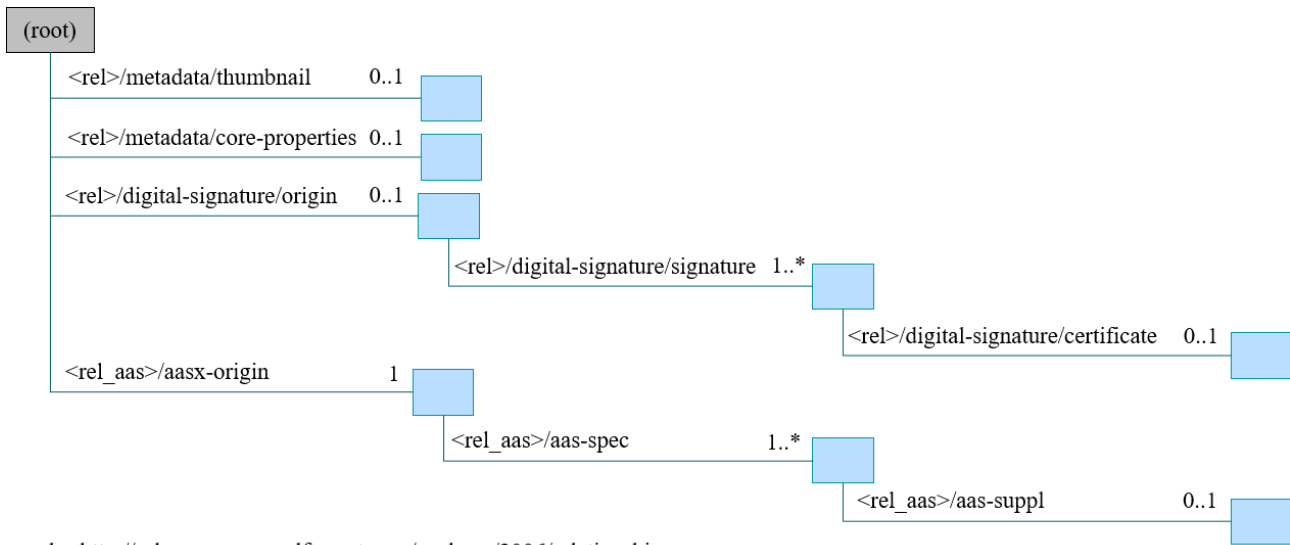
- Package format and rules according to ISO/IEC 29500-2:2012. Any derivative format from this standard (such as the AASX format) requires the definition of a logical model, physical model and a security model. Those specific conventions are described in the next subsections.
- File extension for the AASX format: **.aasx**
- MIME-type for the AASX format: `application/asset-administration-shell-package`³⁸
- **Icon** for the AASX.
- The AASX format can be identified by the file extension and content (MIME) type. Content-wise, it is possible to identify it when reading the first relationship file `/_rels/.rels` (as defined in Open Packaging Conventions) and looking for a relationship type `http://admin-shell.io/aasx/relationships/aasx-origin` (which is the entry point for the logical model of the Asset Administration Shell).
- The following paths and filenames in the package are already reserved by the Open Packaging Conventions specification and therefore shall not be used for any derivative format: `/[Content_Types].xml`; `/_rels/.rels`; `/<file_path>/_rels/<filename>.rels` (where `<filename>` is a file in the package that is source of relationships and `<file_path>` is the path to that file).
- It is not mandatory to open the AASX format in any existing Office Open XML / Open Packaging Conventions compatible office-application (e.g. Microsoft Office, LibreOffice), because the required relationships and files for the different office “models” may not be present (e.g. `http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument` for “docx” document).

8.4 ECMA-376 Relationships

As mentioned before, it is necessary to define a logical model for formats on top of Open Packaging Conventions. Figure 75 defines a set of relationship types (URIs) and the corresponding source files as a part of the logical model for the AASX format. In addition (not shown in Figure 75), a specific relationship instance has also a unique ID and a target resource (URI of a target file inside or outside the package).

³⁸ The currently MIME-type is provisory and needs to be requested officially.

Figure 75 Relationship Types for AASX Packages



rel = <http://schemas.openxmlformats.org/package/2006/relationships>
 rel_aas = <http://admin-shell.io/aasx/relationships>

The relationship types for thumbnail, core-properties, digital-signatures (origin, signature and certificate) are defined by Open Packaging Conventions, so no need to reinvent. The other relationship types were specifically defined to support the AASX package format. Here a short description on each relationship type³⁹ of Figure 75:

- **thumbnail** – Optional. Required to define a thumbnail for that package (e.g. picture of the administrated device). The thumbnail picture can be shown instead of the package’s icon based on the extension and/or content type.
- **core-properties** – Optional. There is a schema for describing the package through "core properties," which uses selected Dublin Core metadata elements in addition to some Open Packaging Conventions-specific elements. The core-properties do not describe the Administration Shell, but the package itself. Some elements of the core-properties may be similar/equal to elements of the Administration Shell. Some core-properties are: Title, Subject, Creator, Keywords, Description, LastModifiedBy, Revision, LastPrinted, Created, Modified, Category, Identifier, ContentType, Language, Version, ContentStatus.
- **digital-signature/origin, digital-signature/signature and digital-signature/certificate** – Optional. Required if you need to sign files and relationships inside the package. Their relationships basically target files that contain the data on signatures (e.g. certificate, digests, ...). See the description later in this document about digital signatures.
- **aasx-origin** – Mandatory. This relationship targets an aasx-origin file which shall be an empty file or a plain text file containing the text “Intentionally empty”. It is the entry-point for all aas specific relationships and files inside the package. The source of the aasx-origin relationship must be the package root.
- **aas-spec** – Mandatory: Targets the file (“aasenv”) that contains the structure/specification of one or more identifiable elements (such as AAS, Submodel or ConceptDescription), according to the XML or JSON format defined in this document. The source of the aasx-spec relationship must be the aasx-origin file.
- **aas-suppl** – Optional. Targets any additional file, which is referenced (not stored as blob) from within the data of an AAS via File element (see Clause 5.7.7.8). The source of any aasx-suppl relationship must be the file containing the AAS structure/specification.

³⁹ To avoid the long names of the relationship types, we will use the short name along the text.

Note: Not every File element inside the specification of an Submodel may target a file stored within the same AASX package. Only a relative URI reference (absolute-path or relative-path reference) shall be interpreted as a reference to a supplementary file within the AASX package.

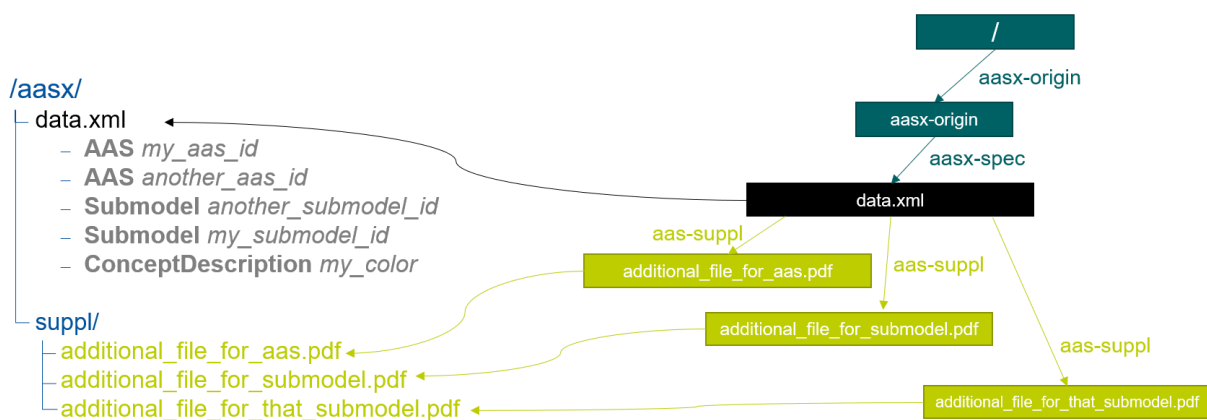
8.5 File Name Conventions

Using the ECMA-375 relationships (8.4) allows to locate files within the AASX package independently from the file name. For example, one package producer might store an aas-spec file in /aasx/device.xml, the other one in /asset-admin-shell/productX123.xml, but both use the same relationship type to target that file. To have a more consistent approach, the following conventions are defined for naming files inside the AASX package:

- **/aasx/** shall be the common prefix for all files containing AASX package specific information.
- **/aasx/aasx-origin** shall be the target of the aasx-origin relationship without content (empty file).
- **/aasx/data.<extension>** shall be the target of the aas-spec relationship, where *<extension>* is "xml" or "json", based on the type of serialization.
- It is also possible to have a serialization of the same data in both serialization formats (xml, json) stored in the same AASX package. In this case, the different serialization formats can be stored in parallel using the aforementioned extensions and appropriate ECMA-376 Content Types (MIME type). In this case, for both of these files the appropriate aas-suppl relationships, targeting the supplementary files, must be created.

An example of an AASX package is shown in Figure 76. It shows the content of the AASX package listed in a tree view using the ECMA-376 relationship types defined in Figure 75 and following the file name conventions as defined above. In this example, it is assumed that the AAS specification files are serialized into XML.

Figure 76 Example of an AASX package content - tree view (left) and ECMA-376 relationship types (right)



In addition to the AASX specific files, files common to all ECMA-376 packages - such as relationship parts (*.rels) and the Content Types stream ([Content_Types].xml) - must be contained in an AASX package in its physical representation as a .zip archive. For more information on these files, please refer to the ECMA-376 specification.

8.6 Digital Signatures

A digital signing feature is already provided by the Open Packaging Conventions specification [27]. Hence, this signing framework for packages can also be used for AASX packages. To ensure the integrity of the AAS data, all relevant files within the package (aasx-origin file, AAS structure specification file, supplementary files) and the associated relationship parts shall be signed.

8.7 Encryption

The Open Packaging Conventions specification (ISO/IEC 29500-2:2012) mentions that “ZIP-based packages shall not include encryption as described in the ZIP specification. Package implementers shall enforce this restriction [M3.9]”⁴⁰. However, an Open Packaging Conventions package may be encrypted with other means and some applications using this package format as the basis for a more specific format, may use encryption during interchange or DRM for distribution [24].

An example is the Office Document Cryptography Structure (MS-OFFCRYPTO) used by derivative office formats. Some used technologies may be covered by Patents from Microsoft and therefore it isn't recommended for the AASX format. Digital Rights Management (DRM) can also be used to encrypt content elements in a package with specific access rights granted to authorize users (see the implementation in the `system.io.packaging` namespace [31]).

Regarding encryption and confidentiality, the following rules shall be followed:

1. Decide if there is a need of including confidential content in a package. If there is no reason, then the confidential content should not be included.
2. If encryption is desired for a temporary communication act (e.g. e-mail exchange, ...) or if a AASX needs to be stored somewhere so that it can be opened later by the same entity, then encryption methods can be used for that specific mean (e.g. use BitLocker when storing the AASX in Windows-based systems that support it, use S/MIME for exchanging encrypted e-mails between entities, etc.).
3. For all other use cases⁴¹ where encryption is required for some or all of the content of the AASX:
 - Encryption methods can be used for individual files in the AASX package, as soon as the “encrypted” version replaces the original file in the package, the content type of the encryption format is known, and the content type must be listed in the `[Content-Type].xml`. The relationships as defined in this document remain the same, whenever content is encrypted or not. Note that Open Packaging Conventions related files as well as relationship files shall not be encrypted, and digital signing must be performed after encryption. One example of an encryption standard is the Secure MIME (S/MIME), where the encrypted content should be stored in `application/pkcs7-mime` format as defined in RFC 5652 and use the file extension `*.p7m`.
 - Besides encrypting the content of the package (individual files) it is possible to encrypt the full package (e.g. also using Secure MIME and saving the encrypted package in `application/pkcs7-mime` file format). In this case, the signature of the content of the package must be done before the encryption.

⁴⁰ The reason for this might be related to the transparency requirement for the package format as well as license requirements of PKWARE. For the ISO/IEC 21320-1 (Document Container File: Core) there is the following statement: “Encryption of individual files and of the central directory is prohibited. Hence this profile of ZIP_PK is more transparent than its parent format.” [30]

⁴¹ A use case could be to encrypt a submodel and only provide the access to the unencrypted data after paying a fee.

9 Mappings to Data Formats to Share I4.0-Compliant Information

9.1 General

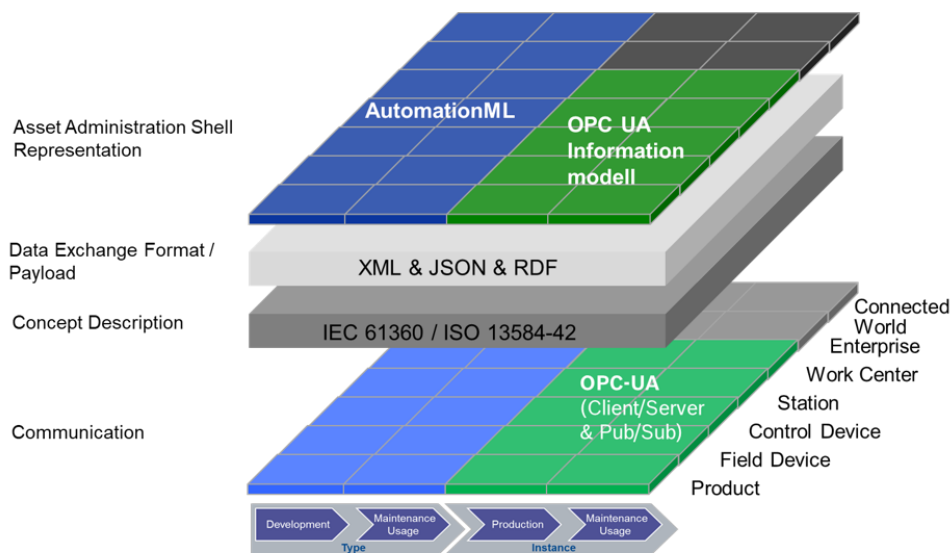
Sharing information between different systems throughout the areas covered by the entire RAMI4.0 model [1] [2] is crucial for Industrie 4.0 applications. OPC UA has been targeted as a format for information models in the domain of production operations, but there is a need for other formats for the other areas and the interrelationships between them.

This document specified the Asset Administration Shell in a technology neutral format, UML. For usage in the different life cycle phases of a product different data formats are used or recommended to be used⁴². For each of these format's serializations and mappings of the Asset Administration Shell are provided to cover the complete life cycle. Table 12 explains the main purpose of each of the formats: OPC UA information models, AutomationML, XML, JSON and RDF. The different purposes are visualized in Figure 77.

Table 12 Distinction of Different Data Formats for the AAS

Data format	Purpose / motivation
OPC UA Information models	Access to all information of the administration data and sharing of live data within production operations. Access for higher-level factory systems to this information.
AutomationML	Sharing of type and instance information about assets, particularly during engineering. Transfer of this information into the operational phase (cf. OPC UA and the corresponding mapping)
XML, JSON	Serialisation of this information for the purpose of technical communication between phases.
RDF	Mapping of this information to enable full use of the advantages of semantic technologies.

Figure 77 Graphic View on Exchange Data Formats for the Asset Administration Shell⁴³



Source: Bosch Rexroth AG. Plattform Industrie 4.0

The mapping specifications and schemata themselves are not part of the specification any longer but maintained open source. This eases usage of the specification and the different formats in open source code projects.

⁴²The abbreviated use of the word “data formats” includes the use of conceptual advantages such as information models, schemes, transmission protocols, etc.

⁴³ Only data formats considered in this document so far are mentioned in the figure.

Projects dedicated to the Asset Administration Shell are planned to be hosted under the new Top Level Project “Digital Twin” of the Eclipse Foundation [55], driven by the Industrial Digital Twin Association (IDTA).

9.2 General Rules

9.2.1 Introduction

There are some general rules that apply to all serializations or can be used in different serializations.

9.2.2 Encoding

For blobs the following encoding is required: base64 string.

9.2.3 Serialization of Values of Type “Reference”

In some mappings or serializations, the type “Reference” is converted into a single string. In this case we recommend using the following serialization:

```

<Reference> ::= [ '['<KeyType>']' ]<Key>{, <Key>}*
<KeyType> ::= GlobalRef | ModelRef
<Key> ::= (<KeyType>)<KeyValue>
<KeyType> ::= value of AAS:Key/type
<KeyIdType> ::= value of AAS:Key/.idType
<KeyValue> ::= value of AAS:Key/value

```

Note: An IRI may contain also special symbols like “(”, “,” and “[”. For being able to distinguish beginning and end of a new key a blank is added before the new key or value.

Note: KeyType is optional because from the first key in the key chain it is clear whether the reference is a global or a model reference. The examples in this document therefore do not use this prefix.

Examples:

Global References:

(GlobalReference)0173-1#02-BAA120#008

[GlobalRef](GlobalReference)0173-1#02-BAA120#008

*(Submodel)http://example.com/aas/1/1/1234859590, (SubmodelElementList)Documents,
(SubmodelElementCollection)0, (MultiLanguageProperty)Title*

Model References:

(ConceptDescription)0173-1#02-BAA120#008

[ModelRef](ConceptDescription)0173-1#02-BAA120#008

(Submodel)http://example.com/aas/1/1/1234859590, (Property)Temperature

9.2.4 Semantic Identifiers for Metamodel and Data Specifications

To enable the unique identification of concepts as used and defined in the metamodel of the Asset Administration Shell rules for creating such identifiers are defined.

The following grammar is used to create valid identifiers:

```

<Namespace> ::= ( <AAS Namespace> | <Data Specification Namespace> )
<Namespace Qualifier> ::= <AAS Namespace Qualifier> | <Data Specification Qualifier>
<AAS Namespace> ::= <Shell-Namespaces>"/aas/"<Version>

```

<Data Specification Namespace> ::=

<Shell-Namespace>"/DataSpecifications/"<idShort of Data Specification><Version>

<Shell-Namespace> ::= "https://admin-shell.io/"

<Version> ::= <Digit>+"/"<Digit>+["/"<Character>+]

<Digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<Character> ::= an unreserved character permitted by DIN SPEC 91406

? ::= zero or one

+ ::= one or more

Up to now two data specification templates are defined. For every data specification it needs to be defined which data specification namespace to use.

<AAS Namespace Qualifier> ::= "AAS:"

<Data Specification Qualifier> ::= defined per Data Specification

A concrete unique identifier is defined as follows:

<AAS Unique Concept Identifier> ::= (<Namespace> | <Namespace Qualifier>)"/"<AAS Concept Identifier>

<AAS Concept Identifier> ::= <AAS Class Name>[(<AAS Attribute> | <AAS Enumeration>)]

<AAS Attribute> ::= "/"<AAS Attribute Name>[{"/"<AAS Attribute Name>}*]

<AAS Enumeration> ::= [{"/"<AAS Attribute Name>}*]"/"<AAS Enumeration Value>

Examples for valid unique AAS concept identifiers:

https://admin-shell.io/aas/2/0/AssetAdministrationShell/administration/version

AAS:AssetAdministrationShell/administration/version

AAS:AssetInformation/assetKind/Instance

The application of the pattern is explained in the following:

The concept identifier of a Class follows the pattern:

<AAS Class name>

This also holds for abstract classes and types including Enumerations.

Examples: *AAS:Submodel*, *AAS:Qualifier*, *AAS:Reference*, *AAS:ContentType*, *AAS:AasSubmodelElements*

Attributes of Classes are separated by "/". Also inherited attributes can be referenced like this if the concrete referable is important in the context.

Basic Pattern:

<AAS Class name>"/"<AAS Attribute Name>

Examples⁴⁴: *AAS:Referable/idShort* or *AAS:Property/idShort* or *AAS:Qualifier/semanticId*

This also holds for attributes of attributes if the cardinality of the attributes involved is not greater than 1:

<AAS Class Name>"/"<AAS Attribute Name>{"/"<AAS Attribute Name>}*]

Examples: *AAS:Identifiable/administration/version*

This also holds for values of enumerations

<AAS Class Name>{"/"<AAS Attribute Name>}*]"/"<AAS Enumeration Value>]

Examples: *AAS:Key/type/Submodel* or *AAS:AasSubmodelElements/Submodel*

In case of an attribute with cardinality greater than 1 no further attributes or enumeration values can be added.

Note: Although the attribute name in UML is always singular even if the cardinality is > 1 the attribute name is annotated by the plural "s".

Examples: *AAS:Operation/InputVariables* or *AAS:AssetAdministrationShell/submodels* or *AAS:Submodel/submodelElements*

AAS:AssetAdministrationShell/submodels/administration/version or *AAS:Submodel/Property/idShort* are no valid concept identifier.

These semantic identifiers are used as values for the *RefSemantic* attribute in AutomationML Mapping of the Asset Administration Shell. These identifiers are also used in OPC UA to describe the semantics of the metamodel via the OPC UA *HasDictionaryEntry* reference type.

For specific serializations and mappings additional identifiers might be needed. For example for a set of Asset Administration Shells or a set of available concept descriptions etc. Here, the AAS metamodel and specification does not give any recommendations.

Data specification handling is special. Data Specification Templates do not belong to the metamodel of the AAS. However, only the predefined data specification templates as specified in this specification are supported in the serializations. For these the corresponding name space qualifier are defined individually.

Examples: *IEC:DataSpecificationIEC61360/preferredName* (see Clause 6.3) or *IEC:DataSpecificationIEC61360/unit* (see Clause 6.4).

For the data specification itself the AAS namespace is used: *AAS:DataSpecifciationIEC61360*

In xml and JSON data specifications are embedded into the schema itself using the attribute "embeddedDataSpecification". For these no concept identifier shall be used. I.e.

AAS:ConceptDescription/embeddedDataSpecification

is not a valid concept identifier. *AAS:DataSpecificationContent* is a valid concept identifier.

9.2.5 Embedded Data Specifications

This specification predefines data specifications that can be used within an AAS to ensure interoperability.

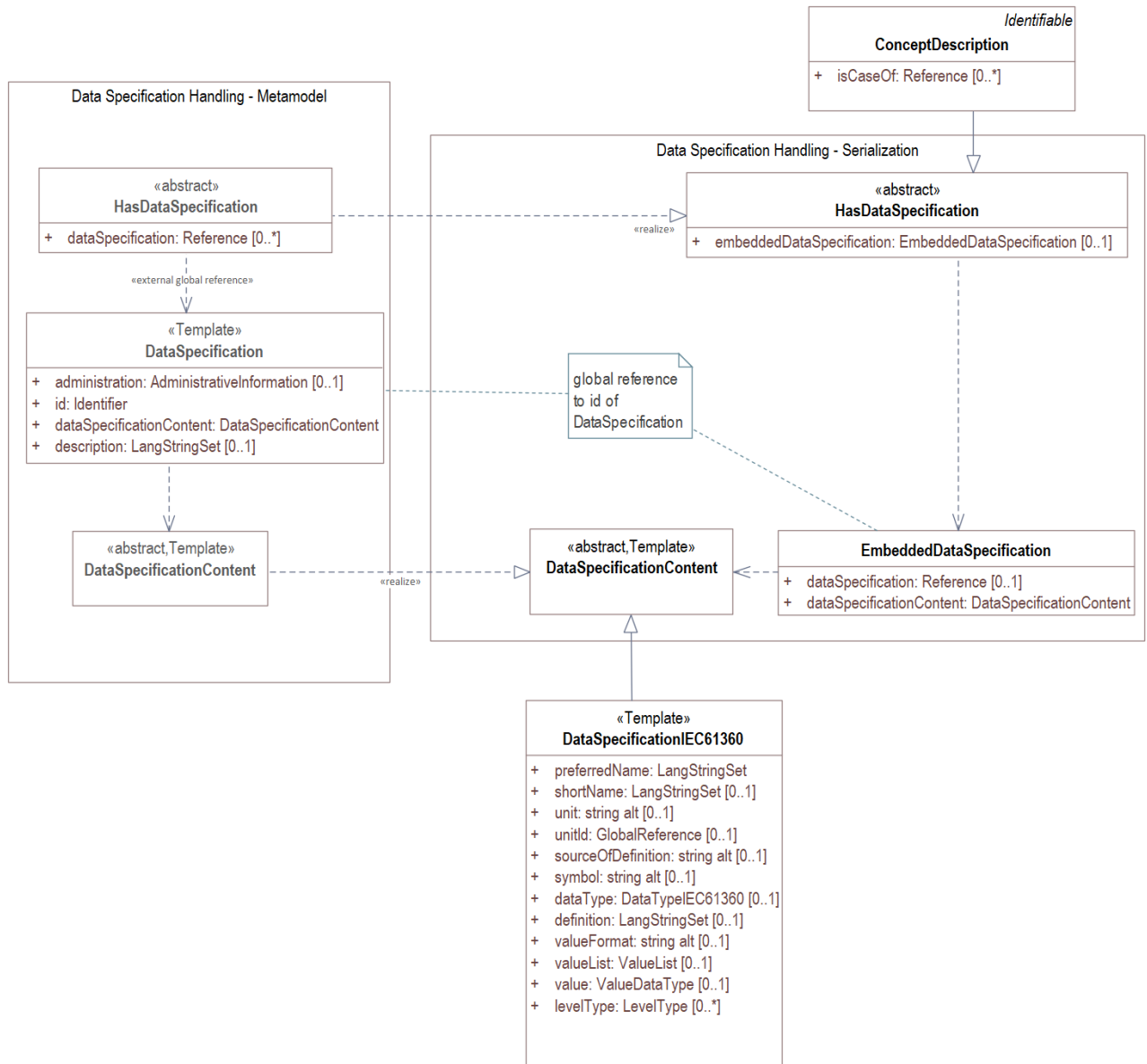
Thus, some serializations or mapping support exactly these data specifications defined in this specification and no others although the metamodel as such is more flexible and would also support proprietary data specifications.

In this case of restricted data specifications to be used the notation is that of "embedded data specifications". In Figure 78 the realization is explained: instead of a set of external global references to externally defined data specifications a set of pairs consisting of an external global reference to a data specification as well as the data specification content itself is directly "embedded". In this realization the data specification content belongs to the schema etc. whereas in the general concept the data specification including its content are not

⁴⁴ For simplicity most examples use the namespace qualifier and not the full path of the namespace.

part of the schema. This is similar to the concept of *semanticIds*: either it is an external global reference to an external concept dictionary or it is a reference to a concept description within the schema. However, for *semanticId* we only allow exactly one reference, whereas for data specifications a set of data specifications references is allowed.

Figure 78 Realization of Embedded Data Specifications



9.3 XML

For import and export scenarios the metamodel of an Asset Administration Shell needs to be serialized. A serialization format is XML.

eXtensible Markup Language (XML⁴⁵) is very well suited to deriving information from an IT system, perhaps to process it manually, and then to feed it into another IT system. It therefore meets the needs of the information sharing scenario defined in the leading picture in Clause 4. XML provides for the possibilities of scheme definitions which can be used to syntactically validate the represented information in each step.

The xml schema (.xsd files) is maintained in the repository "aas-spec" of the github project admin-shell-io [41]: <https://github.com/admin-shell-io/aas-specs/tree/master/schemas/xml>
 The mapping rules how to derive the xml schema from the technology neutral meta model as defined in this specification can be found here: <https://github.com/admin-shell-io/aas-specs/tree/master/schemas/xml#xml-mappingrules>.
 Example files can be found here: <https://github.com/admin-shell-io/aas-specs/tree/master/schemas/xml/examples>.

9.4 JSON

For import and export scenarios the metamodel of an Asset Administration Shell needs to be serialized. A serialization format is JSON⁴⁶ (JavaScript Object Notation).

Additionally, JSON format is used for describing the payload in the http/REST API for active Asset Administration Shells [49].

The JSONschema (.json files) is maintained in the repository "aas-spec" of the github project admin-shell-io [41]: <https://github.com/admin-shell-io/aas-specs/tree/master/schemas/json>
 The mapping rules how to derive the JSON schema from the technology neutral meta model as defined in this specification can be found here: <https://github.com/admin-shell-io/aas-specs/tree/master/schemas/json#json-mapping-rules>.
 Example files can be found here: <https://github.com/admin-shell-io/aas-specs/tree/master/schemas/json/examples>.

9.5 RDF

The Resource Description Framework (RDF) [44] is recommended standard of the W3C to unambiguously model and present semantic data. RDF documents are structured in the form of triples, consisting of subjects, relations and objects. The resulting model is often interpreted as a graph, with the subject and object elements as the nodes and the relations as the graph edges.

RDF is closely related to Web standards, illustrated by the fact that all elements are encoded using (HTTP-)URIs. As a common practice, the provision of additional information at the referenced location of an RDF entity directly allows the interlinking of entities⁴⁷ based on the Web. This process, the following of links in order to discover related information, is called dereferencing a resource and is supported by any browser or web client. Connecting distributed data sources through the Web in the described manner is referenced by the term Linked Data. Connecting the available resources and capabilities of Linked Data with the expressiveness of the Asset Shell is one motivation for the RDF serialization.

In addition, RDF is the basis of a wide range of logical inference and reasoning techniques. Vocabularies like RDF Schema (RDFS) and the Web Ontology Language (OWL) combine the graph-based syntax of RDF with formal definitions and axioms. This allows automated reasoners to understand the relation between entities to some extent and draw conclusions.

Combining both features, the RDF mapping of the Asset Administration Shell can provide the basis for complex queries and requests. SPARQL, the standard query language for the Semantic Web, can combine reasoning

⁴⁵ see: <https://www.w3.org/TR/2008/REC-xml-20081126/>

⁴⁶ see: <https://tools.ietf.org/html/rfc8259> or <https://www.ecma-international.org/publications/standards/Ecma-404.htm>

⁴⁷ Note: entity as a generic term and entity as a specific submodel element subtype need to be distinguished.

features with the integration of external data sources. In order to benefit of these abilities, the AAS requires a clear scheme of its RDF representation.

The RDF scheme/OWL files (.ttl files) are maintained in the repository "aas-spec" of the github project admin-shell-io [41]: <https://github.com/admin-shell-io/aas-specs/tree/master/schemas/rdf>

The mapping rules how to derive the RDF schema from the technology neutral meta model as defined in this specification can be found here: <https://github.com/admin-shell-io/aas-specs/tree/master/schemas/json#ison-mapping-rules>.

Example files can be found here: <https://github.com/admin-shell-io/aas-specs/tree/master/schemas/rdf/examples>.

9.6 AutomationML

For import and export scenarios the metamodel of an Asset Administration Shell needs to be serialized. As a serialization format, AutomationML (IEC 62714) is especially suitable for the engineering phase.

In general the serialization approach is to map each object of the Asset Administration Shell metamodel to an AutomationML Role Class or to an AutomationML Role Class accompanied by an AutomationML Interface Class. This Role Class and (if applied) Interface Class then also define the required attributes in AutomationML.

Asset administration shells itself shall be modelled as AutomationML System Unit Classes or as Internal Elements within an Instance Hierarchy depending of the kind information of type and instance.

For the Role Classes and Interface Classes that are required for the serialization an AutomationML Role Class Library resp. an Interface Class Library are defined and provided to the public.

One of the goals is to ensure that the AutomationML model of the Asset Administration Shell can be used as a standalone AutomationML model as well as in combination with existing AutomationML models such as the upcoming AutomationML Component Description. Therefore, the definition of the serialization approach defined in this Clause is interleaved with the AutomationML definitions and applies the AutomationML technology definitions widely on <https://www.automationml.org/o.red.c/dateien.html>

[37] is the AutomationML application recommendation for the Asset Administration Shell (AR AAS). This annex is just for information.

The works of the mapping of the Asset Administration Shell to AutomationML is are carried out in a joint working group between AutomationML e.V. and Plattform Industrie 4.0.

The resulting application recommendation (AR 004E) "Asset Administration Shell (AAS) Representation" [37] can be found here: <https://www.automationml.org/download-archive/>, together with .aml files

9.7 OPC UA

OPC UA is the suitable for the operating phase of Asset Administration Shells and especially applicable in case of machine to machine communication. The information model [57] is the basis for the definition of so-called OPC UA Information Models, or OPC UA Companion Specifications [58].

The works of the mapping to the OPC Unified Architecture are carried out in a joint working group⁴⁸ "I4AAS" between OPC Foundation, ZVEI and VDMA (<https://opcfoundation.org/markets-collaboration/I4AAS/>) [39].

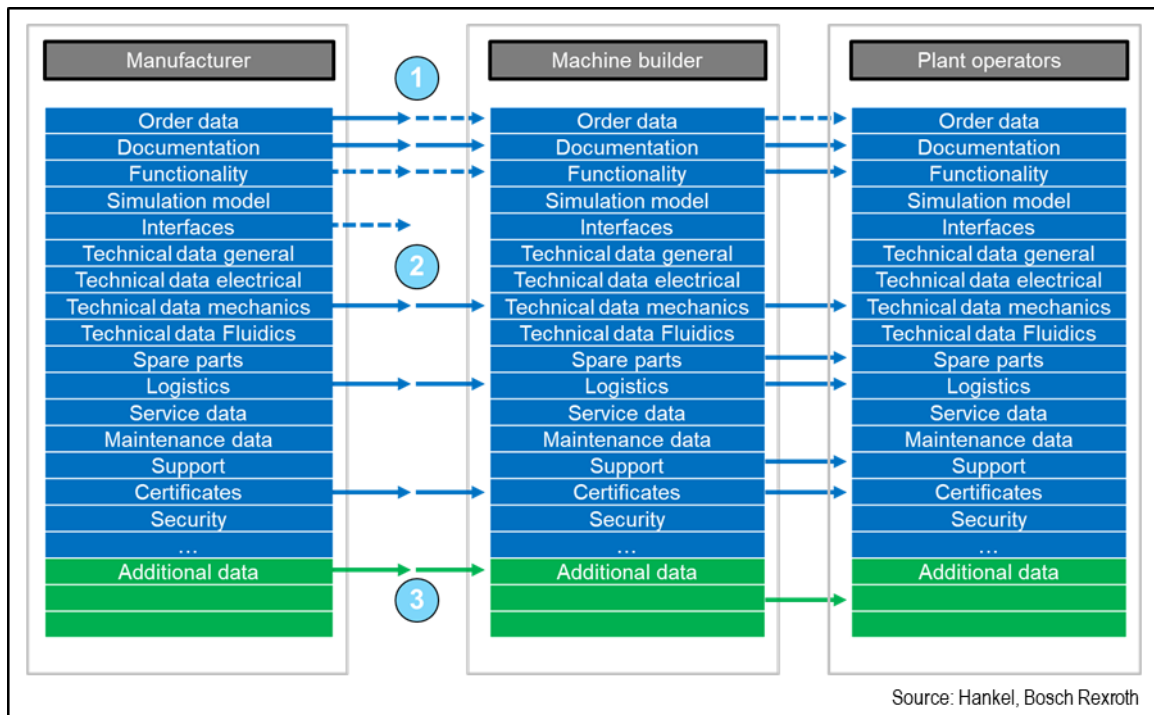
The different versions of the OPC UA Companion Specification for I4 Asset Administration Shell can be found here: <https://reference.opcfoundation.org/<version>/I4AAS/<version>/docs/>, e.g. <https://reference.opcfoundation.org/v104/I4AAS/v100/docs/> for release 1.00 [56].

⁴⁸ see: <https://opcfoundation.org/collaboration/i4aas/>

When exchanging information from partner A to partner B there are two use cases:

- The producer of information does not want to submit the complete information but only parts of it. The information submitted might vary depending on the specific consumer the information is submitted to. I.e. a filtering mechanism is needed that allows to individually shape the information for the specific consumer.
- The consumer of information does not want to include all information provided by the producer of information in his own process, i.e. he wants to filter only the relevant information.

Figure 79 Example Filtering for Export and Import

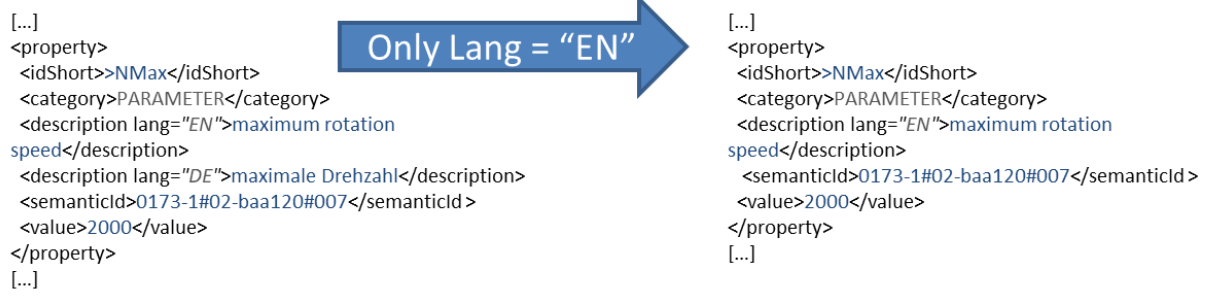


As an example, assume that the producer is submitting the complete order data. However, the consumer (in this case the machine builders) is filtering the information (1) and is only importing the information relevant to him. For the functionality both are filtering: the producer is filtering what he submits to the consumer (2) and the consumer again is not using all functionality but is filtering again which functionality shall be used in his environment. The same is possible between machine builders and operator.

Note: In the use case considered in this document, the exchange of information via sharing of xml files etc. the information that is not intended to be submitted needs to be extracted from the corresponding xml files before delivery or before import, respectively. Role or attributes access control do not fit here. The corresponding access policies might help filtering the corresponding information, but they cannot be submitted as part of the corresponding file exchanged.

Table 13 shows an example when using the defined xml format as defined in this document. In the example the German translation shall not be submitted, only English language is provided for partner B.

Table 13 Example Filtering of Information in XML⁴⁹



⁴⁹ Note: xml serialization may be simplified and not conformant to standardized xml serialization.

11 Tools for the Asset Administration Shell

11.1 Open Source Tools

This clause gives some hints with respect to available open source tools supporting the creation and operating of an Asset Administration Shell.

The top level project “Digital Twin” of the Eclipse Foundation is the home for many projects featuring the Asset Administration Shell. In the following we just mention a few. Besides, there are also other open source projects hosted in other domains. So it is not possible to give a complete overview.

The AASX Package Explorer is an open source browser and editor for creating Asset Administration Shells as .aasx packages [40]. The AASX Package Explorer supports the xml and JSON serialization of the Asset Administration Shell. Additionally, export formats for AutomationML or server generation for OPC UA are provided. But also, additional export formats like BMEcat etc. are supported. Since it is an open source implementation new features are added continuously. On [41] the specifications are hosted and some other open source code projects not yet transferred to Eclipse Foundation.

BaSyx, a software platform, is another open source implementation for the Asset Administration Shell and provides software development kits for C++, C# and Java [42].

12 Summary and Outlook

In this document a metamodel for the structural viewpoint of the Asset Administration Shell is defined using UML. It covers security aspects as well as features for handling composite I4.0 Components. Data specification templates for defining concept descriptions for properties and physical units are provided.

Additionally, an exchange format is specified, the AASX package file format.

Several serializations and mappings are offered:

- XML and JSON for Exchange between partners via exchange format *.aasx*
- RDF for reasoning
- AutomationML for the engineering phase
- OPC UA for the operation phase

Additional parts of the document series cover (see [49]):

- Interfaces and APIs for accessing the information of Asset Administration Shells (access, modify, query and execute information and active functionality). The payload of these APIs is based on the definitions of the information model in this document, part 1.
- The infrastructure, which hosts and interconnects multiple Asset Administration Shells together. It implements registry, discovery services, endpoint handling and more.

Annex

ANNEX A. CONCEPTS OF THE ADMINISTRATION SHELL

i. GENERAL

In this clause, a general information is given about sources of information and relevant concepts for the Asset Administration Shell. Some of these concepts are explained in a general manner. Some concepts are update in order to reflect actual design decisions. No new concepts are introduced. Thus, the clause can be taken as a fully informative (annex) to the specification of the Administration Shell.

ii. RELEVANT SOURCES AND DOCUMENTS

The following documents were used to identify requirements and concepts for the Administration Shell:

- Implementation strategy of Plattform Industrie 4.0 [1][2]
- Aspects of the research roadmap in application scenarios [7]
- Continuation of the application scenarios [8]
- Structure of the Administration Shell [4] [18]
- Examples for the Administration Shell of the Industrie 4.0 Components [6]
- Technical Overview "Secure identities" [9]
- Security of the Administration Shell [14]
- Relationships between I4.0 components – Composite components and smart production [12]

Note 1: The global Plattform Industrie 4.0 glossary can be found at: <https://www.plattform-i40.de/PI40/Navigation/EN/Industrie40/Glossary/glossary.html>

Note 2: The online library of the Plattform Industrie 4.0 can be found at: <https://www.plattform-i40.de/PI40/Navigation/EN/Downloads-News/downloads-news.html>

Note 3: The online library of the Industrial Digital Twin Association can be found at: <https://industrialdigitaltwin.org/en/content-hub/downloads>

iii. BASIC CONCEPTS FOR INDUSTRIE 4.0

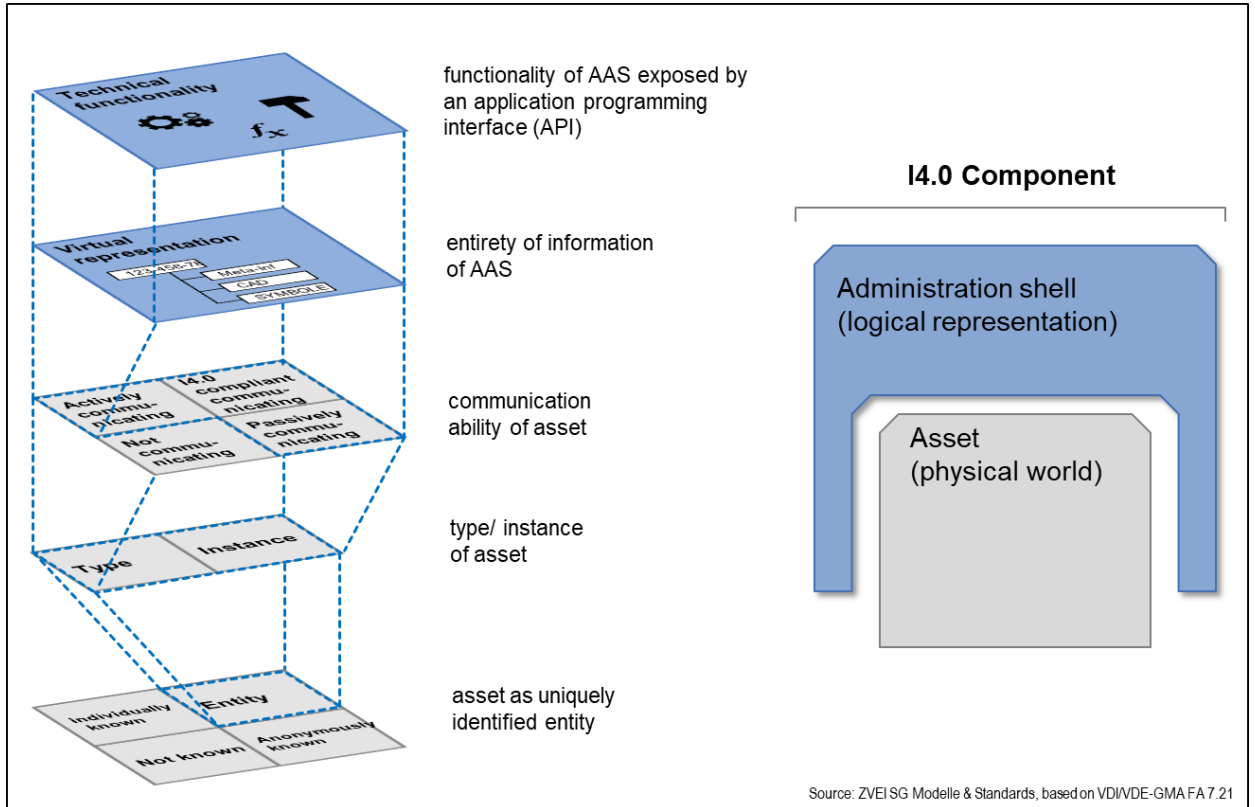
Industrie 4.0 describes concepts and definitions for the domain of smart manufacturing. For Industrie 4.0, the term asset, being any "object which has a value for an organization", is of central importance [2] [23]. Thus, assets in Industrie 4.0 can take almost any form, for example be a production system, a product, a software installation, intellectual properties or even human resources.

According [23], the "reference architecture model Industry 4.0 (RAMI4.0) provides a structured view of the main elements of an asset using a level model consisting of three axes [...]. Complex interrelationships can thus be broken down into smaller, more manageable sections by combining all three axes at each point in the asset's life to represent each relevant aspect."

Assets shall have a logical representation in the "information world", for example shall be managed by IT-systems. Thus, an asset has to be precisely identified as an entity, shall have a "specific state within its life (at least a type or instance)", shall have communication capabilities, shall be represented by means of information and shall be able to provide technical functionality [23]. This logical representation of an asset is called Administration Shell [4]. The combination of asset and Administration Shell forms the so-called I4.0 Component. In international papers [18], the term smart manufacturing replaces the term Industrie 4.0.

For the large variety of assets in Industrie 4.0, the Administration Shell allows handling of these assets in the information world in always the same manner. This reduces complexity and allows for scalability. Additional motivation can be found in [2] [4] [7] [8].

Figure 80 Important concepts of Industrie 4.0 attached to the asset [2] [23]. I4.0 Component to be formed by Administration Shell and Asset.



iv. THE CONCEPT OF PROPERTIES

According [20], the "IEC 61360 series provides a framework and an information model for product dictionaries. The concept of product type is represented by 'classes' and the product characteristics are represented by 'properties'".

Such properties are standardized data elements. The definitions of such properties can be found in a range of repositories, such as IEC CDD (common data dictionary) or ECLASS. The definition of a property (aka standardized data element type, property type) associates a worldwide unique identifier with a definition, which is a set of well-defined attributes. Relevant attributes for the Administration Shell are, amongst other, the preferred name, the symbol, the unit of measure and a human-readable textual definition of the property.

Figure 81 Exemplary definition of a property in the IEC CDD

Code:	0112/2///62683#ACE424
Version:	001
Revision:	01
IRDI:	0112/2///62683#ACE424#001
Preferred name:	rated current
Synonymous name:	
Symbol:	In
Synonymous symbol:	
Short name:	
Definition:	maximum uninterrupted current equal to the conventional free-air thermal current (I _{th})
Note:	
Remark:	
Primary unit:	A
Alternative units:	
Level:	
Data type:	LEVEL(MAX) OF REAL_MEASURE_TYPE

The instantiation of such definition (just 'property', property instance) typically associates a value to the property. By this mechanism, semantically well-defined information can be conveyed by the Administration Shell.

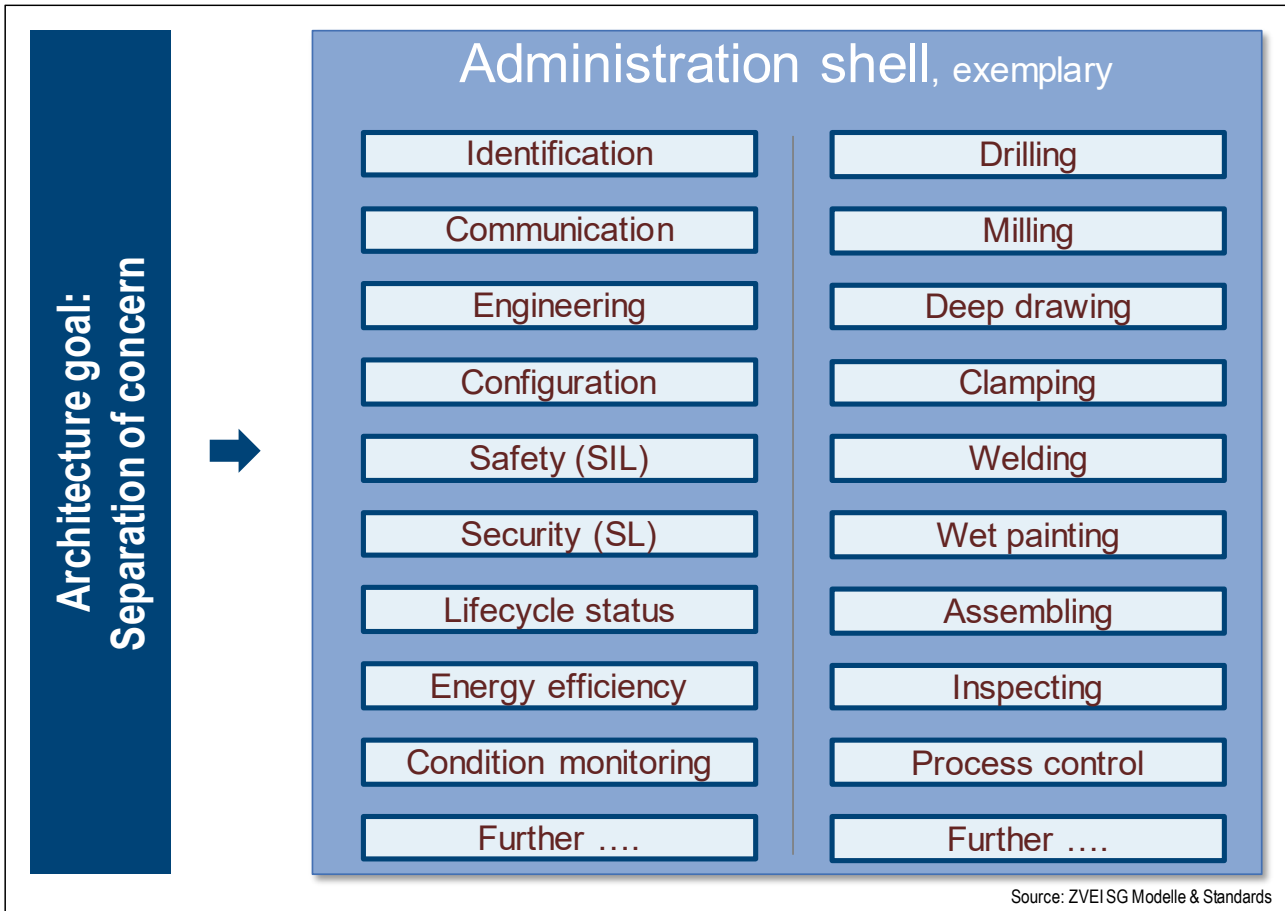
Note: Industrie 4.0 and smart manufacturing in general will require many properties which are beyond the current scope of IEC CDD, ECLASS or other repositories. It is expected that these sets of properties will be introduced, as more and more domains are modelled and standardized (next clause).

V. THE CONCEPT OF SUBMODELS

"The Administration Shell is the standardized digital representation of the asset, corner stone of the interoperability between the applications managing the manufacturing systems" [18]. Thus, it needs to provide a minimal but sufficient description according to the different application scenarios in Industrie 4.0 [7] [8]. Many different (international) standards, consortium specifications and manufacturer specifications can already contribute to this description [18].

As the figure shows, information from different many different technical domains could be associated with a respective asset and thus, many different properties are required to be represented in Administration Shells of future I4.0 Components. In order to manage these complex set of information, submodels provide a separation of concern.

Figure 82 Examples of different domains providing properties for submodels of the Administration Shell



The Administration Shell is thus made up of a series of submodels [4]. These represent different aspects of the asset concerned; for example, they may contain a description relating to safety or security [14] but could also outline various process capabilities such as drilling or installation [6].

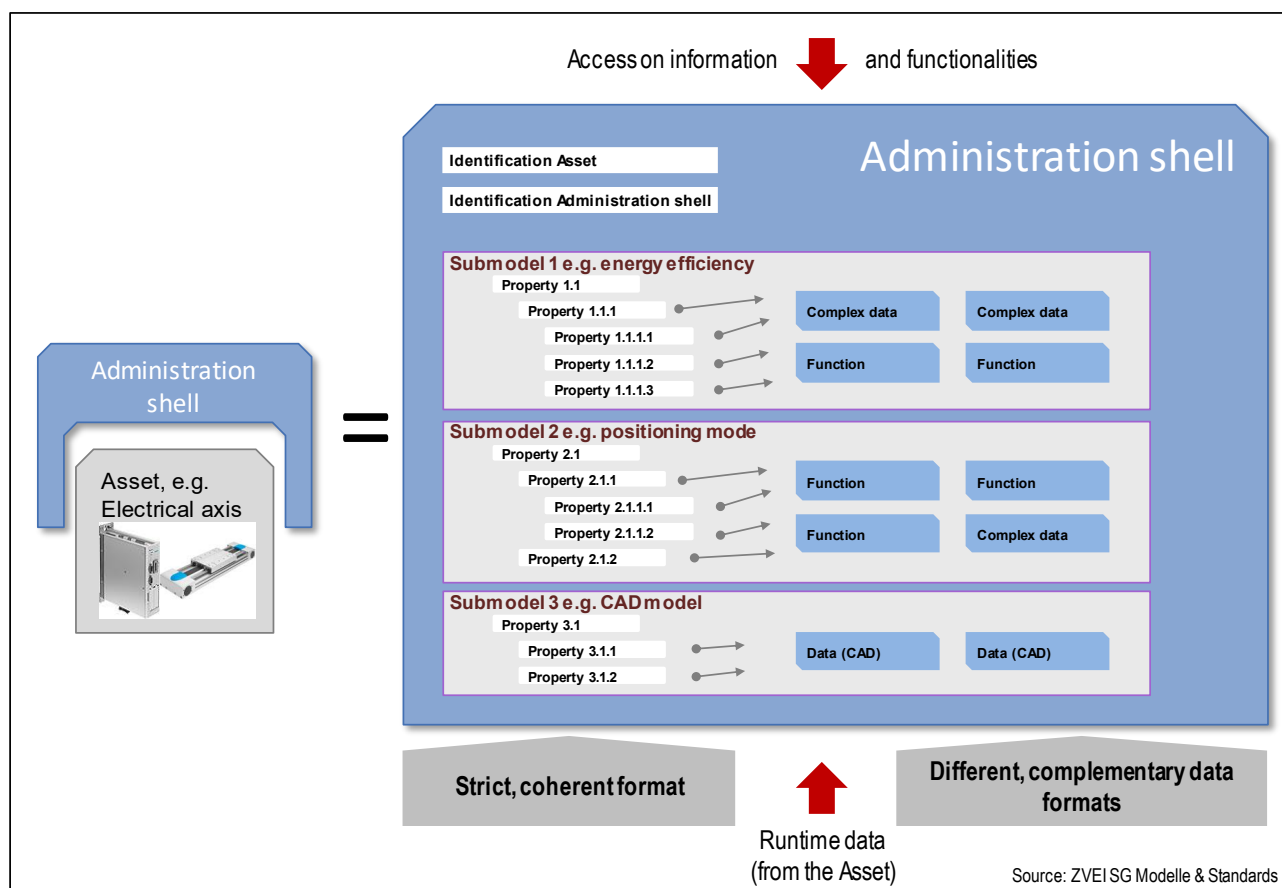
From the perspective of interoperability, the aim is to standardise only a single submodel for each aspect / technical domain. For example, it will thus be possible to find a drilling machine by searching for an Administration Shell containing a submodel "Drilling" with appropriate properties. For communication between different I4.0 components, certain properties can then be assumed to exist. In an example like this, a second submodel, "energy efficiency", could then ensure that the drilling machine is able to cut its electricity consumption when it is not in operation.

Note: side benefit of the Administration Shell will be to simplify the update of properties from product design (and in particular system design) tools, update of properties from real data collected in the instances of assets, improve traceability of assets along life cycle and help certify assets from data.

vi. BASIC STRUCTURE OF THE ASSET ADMINISTRATION SHELL

The document on the Structure of the Asset Administration Shell [4] [18] presented a rough, logical view of the Asset Administration Shell's structure. The Asset Administration Shell – shown in blue ■ in the following figure – comprises different sets of information. Both, the asset and the Administration Shell are identified by a globally unique identifier. It comprises a number of submodels for a characterisation of the Asset Administration Shell.

Figure 83 Basic structure of the Asset Administration Shell



Properties, data and functions will also contain information which not every partner within a value-added network or even within an organisational unit should be able to access or whose integrity and availability should be guaranteed. Therefore, the structure of the Administration Shell shall be able to handle aspects such as access protection, visibility, identity and rights management, confidentiality and integrity. Information security needs to be respected and has to be aligned with an overall security concept. Implementation of security must go together with the implementation of other components of an overall system.

Each submodel contains a structured quantity of properties that can refer to data and functions. A standardized format based on IEC 61360-1/ ISO 13584-42 is envisaged for the properties. Thus, property value definition shall follow the same principles as also ISO 29002-10 and IEC 62832-2. Data and functions may be available in various, complementary formats.

The properties of all the submodels therefore result in a constantly readable directory of the key information of the Administration Shell and thus of the I4.0 component. To enable binding semantics, Administration Shells, assets, submodels and properties must all be clearly identified. Permitted global identifiers are IRDI (e.g. in ISO TS 29002-5, ECLASS and IEC Common Data Dictionaries) and URIs (Unique Resource Identifiers, e.g. for ontologies).

It should be possible to filter elements of the Administration Shell or submodels according to different given views (→ Example C.4 in [18]). This facilitates different perspectives or use-cases for the application of Administration Shell's information.

vii. REQUIREMENTS

This section collects the requirements from various documents that have impact on the specific structure of the Administration Shell. These requirements serve as input for the specific description of the structures of the Administration Shell.

The following requirements are taken from the document "Implementation strategy of Plattform Industrie 4.0" [2]. They are marked "STRAT". The "Tracking" column validates the requirements by linking to features of the UML metamodel or this document in general.

ID	Requirement	Tracking
STRAT#1	A network of Industrie 4.0 components must be structured in such a way that connections between any end point (Industrie 4.0 components) are possible. The Industrie 4.0 components and their contents are to follow a common semantic model.	Network possible but not scope of this part of the document series. Common semantic model realized by domain specific submodels (<i>HasSemantics/ConceptDescription</i> and by <i>Relations</i>)
STRAT#2	It must be possible to define the concept of an Industrie 4.0 component in such a way that it can meet requirements with different focal areas, i.e. "office floor" or "shop floor".	Content-wise, many different submodels possible.
STRAT#3	Industrie 4.0 compliant communication must be performed in such a way that the data of a virtual representation of an Industrie 4.0 component can be kept either in the object itself or in a (higher level) IT system.	Metamodel and information representation independent of any deployment scenario.
STRAT#4	In the case of a virtual representation of an I4.0 component in a higher-level system, an integrity association must be ensured between the asset and its representation.	Integrity part of security approach.
STRAT#5	A suitable reference model must be established to describe how a higher-level IT system can make the Administration Shell available in an Industrie 4.0 compliant manner (SOA approach, delegation principle).	Scope of upcoming part of the document series; not scope of this part.
STRAT#6	A description is required of how the Administration Shell can be "transported" from the originator (e.g. component manufacturer or electrical designer) to the higher-level IT system (e.g. as an attachment to an email).	Hierarchical representation by XML/ JSON and package file format allow for different transport scenarios.
STRAT#7	Depending on the nature of the higher-level systems, it may be necessary for the administration objects to allow for deployment in more than one higher level IT system.	Metamodel and information representation independent of any deployment scenario.
STRAT#8	The Industrie 4.0 component, and in particular the Administration Shell, its inherent functionality and the protocols concerned are to be "encapsulation-capable" or "separable" from any field busses in use.	Metamodel and information representation independent of any communication scenario.
STRAT#9	The aim of the Industrie 4.0 component is to detect non-Industrie 4.0 compliant communication relationships leading to or from the object's Administration Shell and to make them accessible to end-to-end engineering.	Non-Industrie 4.0 compliant communication relationships could be modelled by submodels and therefore made available.
STRAT#10	It should be possible to logically assign other Industrie 4.0 components to one Industrie 4.0 component (e.g. an entire machine) in such a way that there is (temporary) nesting.	<i>References</i> and preparations for <i>Composite components</i> [12]

STRAT#11	Higher level systems should be able to access all Industrie 4.0 components in a purpose-driven and restrictable manner, even when these are (temporarily) logically assigned.	Scope of upcoming part of the document series; not scope of this part.
STRAT#12	Characteristics (1) Identifiability	Given by <i>Identifiable</i>
STRAT#13	Characteristics (2) I4.0-compliant communication	Not scope of part 1
STRAT#14	Characteristics (3) I4.0-compliant services and multiple status	Standardisation of submodels
STRAT#15	Characteristics (4) Virtual description	Available by digital representation (<i>Submodel</i> and <i>SubmodelElement</i>)
STRAT#16	Characteristics (5) I4.0-compliant semantics	<i>HasSemantics</i>
STRAT#17	Characteristics (6) Security and safety	Security by Attribute Based & Role Based Access. Safety not scope of part 1
STRAT#18	Characteristics (7) Quality of services	Metamodel and information representation independent of any communication scenario.
STRAT#19	Characteristics (8) Status	Standardisation of <i>submodels</i>
STRAT#20	Characteristics (9) Nestability	Supported by <i>Submodels</i> representing a <i>Bill of Material</i> of an Asset, <i>Entities</i> and <i>RelationshipElements</i>
STRAT#21	The minimum infrastructure must satisfy the principles of Security by Design (SbD).	Security by Attribute Based & Role Based Access.

The following requirements are taken from the document “The Structure of the Administration Shell:

Trilateral perspectives from France, Italy and Germany” [18]. They are marked “tAAS”.

Note: The term “property” was used in a very broad sense in previous publications of the Plattform Industrie 4.0. The metamodel in this document distinguishes between properties in a more classical sense as data element like “maximum temperature” and other submodel elements like operations, events etc.

Source	Requirement	Tracking
tAAS-#1	The Administration Shell shall accept properties from different technical domains in mutually distinct submodels that can be version-controlled and maintained independently of each other.	<p><i>Identifiable</i></p> <p><i>AdministrativeInformation</i></p> <p><i>Submodel</i></p> <p>Requirements tAAS-#1 implicitly contains the requirements of versioning. Versioning is supported for all elements inheriting from <i>Identifiable</i>.</p> <p>Requirement tAAS-#1 is fulfilled because several submodels per AAS are possible. Every submodel is identifiable and an <i>Identifiable</i> may contain administrative information (<i>AdministrativeInformation</i>) for versioning.</p> <p>The reason for submodels to be identifiable is that they may be maintained independently of other submodels (Requirement tAAS-#1) and that they can be reused within different AAS. However, since submodel elements may refer to elements from other AAS dependencies have to be considered in parallel development and before reuse.</p>

tAAS-#2	The Administration Shell should be capable of including properties from a wide range of technical domains and of [sic!] identify which domain they derive from.	<p><i>HasSemantics</i></p> <p>Via semantic references property definitions from different dictionaries and thus different domains can be used within submodels.</p> <p>The only thing required is that the domain a property is derived from has a unique ID (<i>semanticId</i>).</p>
tAAS-#3	For finding definitions within each relevant technical domain, different procedural models should be allowed that respectively meet the requirements of standards, consortium specifications and manufacturer specifications sets.	<p><i>HasSemantics/semanticId</i> (see tAAS-#2)</p> <p><i>ConceptDescription</i></p> <p>Proprietary manufacturer specific property – or more general – concept descriptions or copies from external dictionaries are supported by defining <i>ConceptDescriptions</i>. They are referenced in <i>semanticId</i> via their global ID.</p> <p>Up to now there is only a predefined data specification template for <i>Property</i> elements (<i>DataSpecificationIEC61360</i>).</p> <p>Usage of proprietary concept descriptions is not recommended because then interoperability cannot be ensured.</p>
tAAS-#4	<p>Different Administration Shells in respect of an asset must be capable of referencing each other.</p> <p>In particular, elements of an Administration Shell should be able to play the role of a “copy” of the corresponding components from another Administration Shell.</p>	<p><i>AssetAdministrationShell/derivedFrom</i></p> <p>The <i>derivedFrom</i> relationship is especially designed for supporting the relationship between an Asset Administration Shell representing an asset type and the Asset Administration Shells representing the asset instances of this asset type.</p> <p>See also tAAS-#16</p>
tAAS-#5	Individual Administration Shells should, while retaining their structure, be combined into an overall Administration Shell.	<p><i>AssetAdministrationShell/assetInformation</i></p> <p><i>RelationshipElement</i></p> <p>Via the submodel element “<i>RelationshipElement</i>” relations between entities can be defined.</p>
tAAS-#6	Identification of assets, Administration Shells, properties and relationships shall be achieved using a limited set of identifiers (IRDI, URI and GUID), providing as far as possible offer global uniqueness.	<p><i>Identifiable</i></p> <p>Requirement tAAS-#6 is fulfilled for all elements inheriting from <i>Identifiable</i>. For example, this is the case for <i>Asset</i>, <i>AssetAdministrationShell</i> and for concept descriptions. However, properties (like any other submodel element) are only referable. However, unique referencing is possible via the unique submodel ID and the <i>Reference</i> via <i>Keys</i> concept.</p> <p>The supported ID types include IRDI, URI (since URI are a subset of <i>IRI</i>), <i>IRI</i> and GUID (via <i>Custom</i>) as requested.</p>
tAAS-#7	The Administration Shell should allow retrieval of alternative identifiers such as a GS1 and	<p><i>AssetInformation/specificAssetIds</i></p> <p><i>AssetInformation/globalAssetId</i></p>

	GTIN identifier in return to asset ID (referencing).	Every asset has a globally unique identifier (<i>globalAssetId</i>). Besides this global identifier additional external identifiers can be specified (<i>specificAssetIds</i>).
tAAS-#8	The Administration Shell consists of header and body.	<p><i>AssetAdministrationShell</i></p> <p><i>AssetAdministrationShell/id</i></p> <p><i>AssetAdministrationShell/administration</i></p> <p><i>AssetAdministrationShell/assetInformation</i></p> <p>The Asset Administration Shell does not explicitly distinguish between Header and Body. However, the Asset Administration Shell has attributes defined that belong to itself like the global unique ID (<i>identification</i>), version information (<i>administration</i>), a mandatory reference to the asset identifier information (<i>assetInformation</i>) it represents etc.</p>
tAAS-#9	The header contains information about the identification.	<p><i>AssetAdministrationShell/assetInformation</i></p> <p>The Asset Administrative Shell is representing an asset with a unique ID.</p> <p>See also tAAS-#7</p> <p>See also tAAS-#13</p>
tAAS-#10	The body contains information about the respective asset(s).	<p><i>AssetAdministrationShell/submodels</i></p> <p>All submodels give information with respect to or related to the asset presented by the AAS.</p> <p><u>Note:</u> An Asset Administration Shell is representing exactly one asset. In case of a composite Asset Administration Shell it is implicitly representing several assets (see also tAAS-#5).</p>
tAAS-#11	The information and functionality in the Administration Shell is accessible by means of a standardized application programming interface (API).	is covered in part 2 of this document series [49].
tAAS-#12	The Administration Shell has a unique ID.	<p><i>AssetAdministrationShell/id</i></p> <p>Since <i>AssetAdministrationShell</i> inherits from <i>Identifiable</i> Requirement tAAS-#12 is fulfilled.</p>
tAAS-#13	The asset has a unique ID.	<p><i>AssetInformation/globalAssetId</i></p> <p>The unique ID of the asset is the value of the <i>globalAssetId</i>. See also Requirement tAAS-#7.</p>
tAAS-#14	An industrial facility is also an asset, it has an Administration Shell and is accessible by means of ID.	<p><i>AssetInformation/globalAssetId</i></p> <p>The only assumption is that the industrial facility also has a globally unique ID that can be used as value of the <i>globalAssetId</i>.</p>

		<p><u>Note:</u> See also composite Asset Administration Shell (see tAAS-#5) that allows the modelling of complex assets consisting of other assets that are represented by an AAS each by themselves.</p>
tAAS-#15	Types and instances must be identified as such.	<p><i>AssetInformation/assetKind</i> (values: <i>Type</i> or <i>Instance</i>)</p> <p><i>AssetAdministrationShell/derivedFrom</i></p> <p>With attribute kind of Asset Requirement tAAS-#15 is fulfilled and asset types can be distinguished from asset instances.</p> <p>Additionally, a <i>derivedFrom</i> relationship can be established between the AAS for an asset instance and the AAS for the asset type.</p>
tAAS-#16	The Administration Shell can include references to other Administration Shells or Smart Manufacturing information.	<p><i>ReferenceElement</i></p> <p><i>File</i></p> <p><i>Blob</i></p> <p><i>AssetAdministrationShell/derivedFrom</i></p> <p>The <i>derivedFrom</i> relationship between two AAS is special and is for example used to establish a relationship between asset instances and the asset type.</p> <p>For composite AAS (see tAAS-#5) there also is the relationship to AAS the composite AAS is composed of.</p> <p>The <i>ReferenceElement</i> is very generic and can reference another AAS as well as information within another AAS or even some information that is completely outside any AAS (as long as it has a global unique ID).</p> <p>Files and BLOB can be used as submodel elements to include very generic manufacturing information that is not or cannot be modelled via properties or the other submodel elements defined for the Asset Administration Shell.</p>
tAAS-#17	Additional properties, e. g. manufacturer specific, must be possible.	<p><i>HasDataSpecification</i></p> <p><i>ConceptDescription</i></p> <p><i>HasExtensions</i></p> <p>Via data specification templates additional attributes for assets, properties and other submodel elements, submodels, views and even the Asset Administration Shell itself can be defined and checked by tools.</p> <p>New proprietary concept descriptions (<i>ConceptDescription</i>) can be added and used for semantic definition of properties or other submodel elements.</p> <p>Via extensions proprietary information can be added to any referable. Extensions are not subject to standardization and thus can be ignored for interoperability use cases.</p>

		Via API (see tAAS-#11) new properties, other submodel elements and submodels can be added – assumed the corresponding access permissions are given.
tAAS-#18	A reliable minimum number of properties must be defined for each Administration Shell.	<p><i>HasKind</i> for <i>Submodel</i> and <i>SubmodelElements</i></p> <p>A reliable minimum number of properties is defined by the metamodel itself. They are called (class) attributes.</p> <p><i>HasKind</i> (with <i>kind=Template</i>) for <i>Submodel</i> and <i>submodel elements</i> enables the definition of submodel (element) templates. These templates are referenced via <i>semanticId</i>.</p> <p><u>Note:</u> the term property within the metamodel has a special semantics and shall not be mixed with the implicitly available attributes of the different classes. Although these attributes as well might be based on existing standards they are no properties in the sense that a semantic reference can be added that defines the semantics externally: The semantics is defined for the metamodel itself in the class tables within this document.</p>
tAAS-#19	The properties and other elements of information in the Administration Shell must be suitable for types and instances.	<p><i>HasKind</i> (with <i>kind=Template</i> or <i>kind=Instance</i>) for <i>Submodel</i> and <i>SubmodelElement</i></p> <p>All elements inheriting from <i>HasKind</i> can distinguish between types and instances. This is especially true for <i>SubmodelElement</i> and <i>Submodel</i>.</p> <p><u>Note:</u> Submodels or properties of <i>kind=Template</i> do not describe an asset of <i>kind=Type</i>. This is done via properties of <i>kind=Instance</i>.</p>
tAAS-#20	There must be a capability of hierarchical and countable structuring of the properties.	<p><i>SubmodelElementList</i></p> <p><i>SubmodelElementCollection</i></p> <p>Requirement tAAS-#20 is fulfilled by lists and structs of data elements. Lists and structs are built recursively and thus contain other submodel elements of the same AAS. For referencing properties or other submodel elements of other AAS a reference (<i>ReferenceElement</i>) or relationship element (<i>RelationshipElement</i>) needs to be included in the list or as element of the collection.</p>
tAAS-#21	Properties shall be able to reference other properties, even in other Administration Shells.	<p><i>SubmodelElementList</i></p> <p><i>SubmodelElementCollection</i></p> <p><i>ReferenceElement</i></p> <p><i>RelationshipElement</i></p> <p><i>OperationVariable</i> in <i>Operation</i></p> <p>A reference element can either reference any other element that is referable (i.e. inheriting from <i>Referable</i>) within the</p>

		<p>same or another AAS. Or it can reference entities completely outside any AAS via its global ID.</p> <p><u>Note:</u> For referencing elements within the same AAS it is not always necessary to use a reference property. Depending on the context also submodel element collections, relations etc. might be more suitable.</p> <p>Within <i>operations</i> also other elements are referenced or used as input or output argument via <i>OperationVariable</i></p>
tAAS-#22	Properties must be able to reference information and functions of the Administration Shell.	<p><i>Operation</i></p> <p>See also tAAS-#21</p> <p>Functions in the sense of executable entities are represented as <i>operations</i>.</p>

The following requirements have been derived from the document "Security of the Administrative Shell" [14]. They are marked as "SecAAS"

ID	Requirement	Tracking
SecAAS-#1	Identification and authentication: It must be ensured that the correct entities (Administration Shell and users) interact with each other. This applies both in a local communication context (within a machine or plant) and in a global context (across companies). The clear identification (by authentication) of the communication partners is a basic requirement for the interaction with a management shell. Without them, further security features (confidentiality, integrity, etc.) cannot be guaranteed.	<p>Certificates are considered to be part of the infrastructure.</p>
SecAAS-#2	User and rights management: An Asset Administration Shell can have different interaction partners. To control the possibilities of interaction with the Administration Shell, a user and rights management is necessary.	<p><i>Security/accessControlPolicyPoints</i></p> <p><i>AccessControlPolicyPoints/policyAdministrationPoints</i></p> <p><i>AccessControl</i></p> <p><i>AccessControl/accessPermissionRules</i></p> <p>There is no explicit subject management in the AAS: It is assumed that the identity of the subject requesting access with a given role (via the API - see tAAS-#11) is authenticated outside the AAS. The AAS can check the authorization via the endpoint to the subject attributes' provider.</p> <p>For every object in the Asset Administration Shell access permission rules can be defined.</p>

SecAAS-#3	Secure Communication: Communication with the Administrative Shell may include sensitive information. Likewise, a change in the communication between the Administration Shell and its communication partners can cause serious and dangerous disruptions in a machine or plant. It is therefore mandatory that adequate measures be taken to ensure communication security. This must be done by using appropriate security protocols.	Not applicable
SecAAS-#4	Event logging: The traceability of interaction with the Administration Shell plays a crucial role in the detection of security incidents. This traceability is achieved through logging / event logging and auditing. The management shell must therefore provide methods that log accesses and changes in state of the management shell without modification. It is also important to be able to centrally collect and evaluate this event information.	History handling will be detailed in future parts or versions of the document (series).

ANNEX B. AASX PACKAGE FILE FORMAT – BACKGROUND INFORMATION

i. SELECTION OF THE REFERENCE FORMAT FOR THE ASSET ADMINISTRATION SHELL PACKAGE FORMAT

The *Führungskreis Industrie 4.0 – UAG Verwaltungsschale* has decided to use the Open Packaging Conventions (OPC)⁵⁰ format as the reference for the Asset Administration Shell package format definition, due to the following reasons:

- Open Packaging Conventions is an international standard specified in ISO/IEC 29500-2:2012 and ECMA-376.
- Open Packaging Conventions is based on ZIP (as a package container) and XML (for the description of some internal files and definitions). Those two technologies are the most widely used in their respective domains and are also addressed for long-term archiving.
- Open Packaging Conventions can be used as package for non-office applications too (there are many examples available, such as NuGet, FDI packages, etc.). It provides a logical model that is independent from how the files are stored in the package. This logical model can be expanded to any sort of application.
- Open Packaging Conventions is also used in the scope of Industry (e.g. FDI packages) and currently in discussion as possible container format for some FDT® and ODVA Project xDS™ use cases.
- Open Packaging Conventions (and Open Document Format packages too) supports digital signing. It can be done for individual files inside the package. Encryption isn't specified in Open Packaging Conventions (it only mentions what shall not be done). Anyway, encryption is still possible (see later)
- There are some APIs to handle Open Packaging Conventions packages (Windows API, .NET, Java, ...) without the need of much knowledge on the technical specification
- Chunking in Open Packaging Conventions is encouraged, i.e. split files into small chunks. This is better for reducing the effect of file corruption and better for data access.
- There are some international organizations that recommend using Open Document Format (ISO/IEC 26300-3) instead (e.g. EU, NATO, ...), but this recommendation is related to the formats used specifically in office applications.
- The Office Open XML and Open Packaging Conventions specifications originated from Microsoft Corporation and later standardized as ISO/IEC 29500 and ECMA-376. Current and future versions of ISO/IEC 29500 and ECMA-376 are covered by Microsoft's Open Specification Promise, whereby Microsoft "irrevocably promises" not to assert any claims against those making, using, and selling conforming implementations of any specification covered by the promise (so long as those accepting the promise refrain from suing Microsoft for patent infringement in relation to Microsoft's implementation of the covered specification). [24]
- Office Open XML (including the Open Packaging Conventions format) and Open Document Format are politically conflicting formats (see details in [25] and [26]). Choosing Open Packaging Conventions as the option for storing the Asset Administration Shell information was solely a technical decision based on the arguments mentioned here.
- Open Packaging Conventions was chosen in favour of iIRDS (v1.0). The scope of iIRDS might not be aligned with the requirements of the Asset Administration Shell, i.e. iIRDS is mostly a format for storing technical documentation of industry devices based on concepts of ontology.

⁵⁰ Not to be confused with OPC (Open Platform Communication) of the OPC Foundation. Therefore, we will use the full term of "Open Packaging Conventions" instead of the abbreviation "OPC".

ANNEX C. TEMPLATES FOR UML TABLES

i. GENERAL

In this annex, the templates used for element specification are explained. For details for the semantics see Annex Legend for UML Modelling.

ii. TEMPLATE FOR CLASSES

Template for Classes:

Class:			
Explanation:			
Inherits from:	--		
Attribute	Explanation	Type	Card.

The following kinds of *Types* are distinguished:

- *Primitive*: Type is no object type (class) but a data type, it is just a value
- *Class*: Type is an object type (class), it realized as composite aggregation (composition) (does not exist independent of its parent)
- *ModelReference*<{Referable}> is a Reference with *Reference/type=ModelReference*. Such a reference is called model reference. The {Referable} is to be substituted by any referable element (including *Referable* itself for the most generic case): The element that is referred to is denoted in the *Key/type*=<{Referable}> for the last *Key* in the model reference. For the graphical representation see Annex Legend for UML Modelling, Figure 103. For more information on referencing see Clause 5.7.9.

Card. is the cardinality (or multiplicity) defining the lower and upper bound of the number of instances of the member element. "*" denotes an arbitrary infinite number of elements of the corresponding Type. "0..1" means optional. "0..*" or "0..3" etc. means that the list may be either not available (null object) or empty.

Note: Attributes having a default value are always considered to be optional. The reason is that there always is a value for the attribute because the default value is used for initialization in this case.

Examples for valid model references

If Class type equal to "ModelReference<Submodel>" then the following reference would be a valid reference (using the text serialization as defined in Clause 9.2.3):

(Submodel)http://example.com/aas/1/1/1234859590

If Class type equal to "ModelReference<Referable>" then the following references would be a valid references (using the text serialization as defined in Clause 9.2.3):

(Submodel)http://example.com/aas/1/1/1234859590

(Submodel)http://example.com/aas/1/1/1234859590, (Property)temperature

(Submodel)http://example.com/aas/1/1/1234859590, (File)myDocument

This would be an invalid reference for "ModelReference<Referable>":

(Submodel)http://example.com/aas/1/1/1234859590, (File)myDocument (FragmentReference)Hints

This would be an invalid reference for "ModelReference<Submodel>"

(Submodel)http://example.com/aas/1/1/1234859590, (Property)temperature

iii. TEMPLATE FOR ENUMERATIONS

Template for Enumerations:

Enumeration:	
Explanation:	
Set of:	--
Literal	Explanation
enumValue1	Value of enumeration
enumValue2	Value of enumeration, also included in one of the enumerations listed in "Set of:"

"Set Of" lists enumerations that are contained in the enumeration. This is just for validation that all elements are considered that are relevant for the enumeration. The elements "inherited" are greyed.

Enumeration values use Camel Case notation.

iv. TEMPLATE FOR PRIMITIVES

Template for Primitive:

Primitive	Explanation	Value Examples

v. HANDLING OF CONSTRAINTS

Constraints are prefixed with **AASd-** followed by a three-digit number. The "d" in AAS was motivated by "in Detail". The numbering of constraints is unique within namespace AASd, a number of a constraint that was removed will not be used again.

Security constraints are prefixed with **AASs-** followed by a three-digit number. The "s" in AAS was motivated by "Security". The numbering of constraints is unique within namespace AASs, a number of a constraint that was removed will not be used again.

Constraints specific for data specifications IEC61360 are prefixed with **AASc-** followed by a three-digit number. The "c" in AAS was motivated by "Concept Description". The numbering of constraints is unique within namespace AASc, a number of a constraint that was removed will not be used again.

ANNEX D. LEGEND FOR UML MODELLING

i. OMG UML GENERAL

In the following the used UML elements used in this specification are explained. For more information please refer to the comprehensive literature available for UML. The formal specification can be found in [47].

Figure 84 shows a class with name “Class1” and an attribute with name “attr” of type *Class2*. Attributes are owned by the class. Some of these attributes may represents the end of binary associations, see also Figure 91. In this case the instance of *Class2* is navigable via the instance of the owning class *Class1*.⁵¹

Figure 84 Class

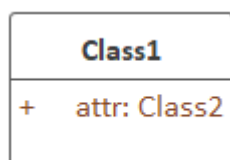


Figure 85 shows that *Class4* is inheriting all member elements from *Class3*. Or in other word, *Class3* is a generalization of *Class4*, *Class4* is a specialization of *Class3*. This means that each instance of *Class4* is also an instance of *Class3*. An instance of the *Class4* has the attributes *attr1* and *attr2* whereas instances of *Class3* only have the attribute *attr1*.

Figure 85 Inheritance/Generalization

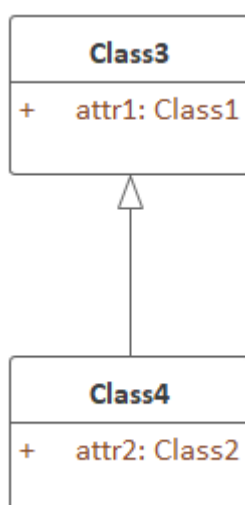


Figure 86 defines the required and allowed multiplicity/cardinality within an association between instances of *Class1* and *Class2*. In this example an instance of *Class2* is always related to exactly one instance of *Class1*. An instance of *Class1* is either related to none, one or more (unlimited, i.e. no constraint on the upper bound) instances of *Class2*. The relationship can change over time.

Multiplicity constraints can also be added to attributes and aggregations.

The notation of multiplicity is as follows:

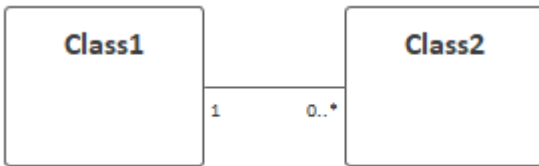
<lower-bound>.. <upper-bound>

⁵¹ „Navigability notation was often used in the past according to an informal convention, whereby non-navigable ends were assumed to be owned by the Association whereas navigable ends were assumed to be owned by the Classifier at the opposite end. This convention is now deprecated. Aggregation type, navigability, and end ownership are separate concepts, each with their own explicit notation. Association ends owned by classes are always navigable, while those owned by associations may be navigable or not. [47]”

Where <lower-bound> is a value specification of type Integer - i.e. 0, 1, 2, ... - and <upper-bound> is a value specification of type UnlimitedNatural. The star character (*) is used to denote an unlimited upper bound.

The default is 1 for lower-bound and upper-bound.

Figure 86 Multiplicity



A multiplicity element represents a collection of values. The default is a set, i.e. it is not ordered and the elements within the collection are unique, i.e. contain no duplicates. In Figure 87 an ordered collection is shown: the instances of *Class2* related to an instance of *Class1* are ordered. The stereotype <<ordered>> is used to denote that the relationship is ordered.

Figure 87 Ordered Multiplicity



Figure 88 shows that the member ends of an association can be named as well. I.e. an instance of *Class1* can be in relationship “relation” to an instance of *Class2*. Vice versa the instance of *Class2* is in relationship “reverseRelation” to the instance of *Class1*.

Figure 88 Association

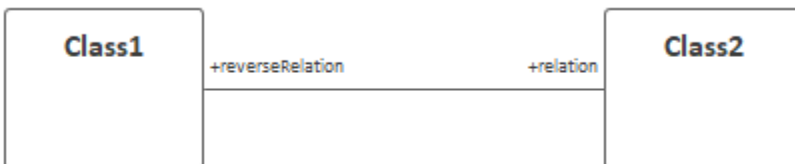


Figure 89 shows a composition, also called a composite aggregation. A composition is a binary association. It groups a set of instances. The individuals in the set are typed as specified by *Class2*. The multiplicity of instances of *Class2* to *Class1* is always 1 (i.e. upper-bound and lower-bound have value “1”). One instance of *Class2* belongs to exactly one instance of *Class1*. There is no instance of *Class2* without a relationship to an instance of *Class1*. In Figure 90 the composition is shown using an association relationship with a filled diamond as composition adornment.

Figure 89 Composition (composite aggregation)

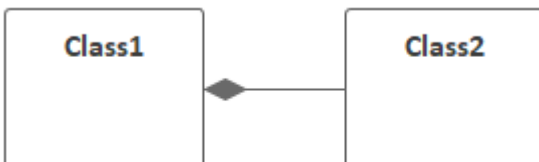


Figure 90 show an aggregation. An aggregation is a binary association. In contrast to a composition an instance of *Class2* can be shared by several instances of *Class1*. In Figure 90 the shared aggregation is shown using an association relationship with a hollow diamond as aggregation adornment.

Figure 90 Aggregation

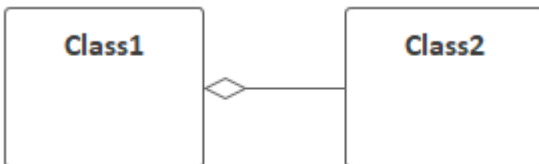


Figure 91 shows that the attribute notation can be used for an association end owned by a class. In this example the attribute name is “attr” and the elements of this attribute are typed with *Class2*. The multiplicity, here “0..*”, is added in square brackets. If the aggregation is ordered then this is added in curly brackets like in this example.

Figure 91 Navigable Attribute Notation for Associations

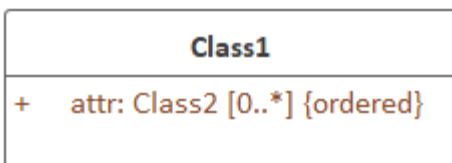


Figure 92 shows a class with three attributes with primitive types and default values. When a property with a default value is instantiated, in the absence of some specific setting for the property, the default value is evaluated to provide the initial values of the property.

Figure 92 Default Value

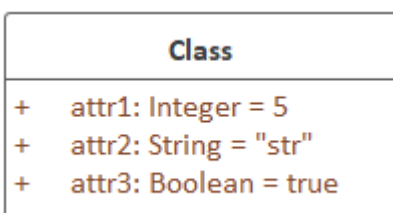


Figure 93 shows that there is a dependency relationship between *Class1* and *Class2*. In this case the dependency means that *Class1* depends on *Class2*. Why is this: because the type of attribute *attr* depends on the specification of class *Class2*. A dependency is shown as dashed arrow between two model elements.

Figure 93 Dependency

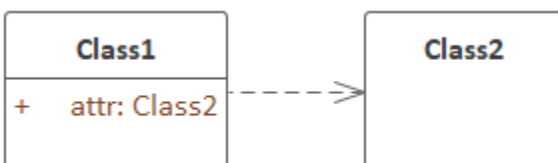


Figure 94 shows an abstract class. It uses the stereotype <<abstract>>. There are no instances of abstract classes. They are typically used to specific member elements that are then inherited by non-abstract classes.

Figure 94 Abstract Class

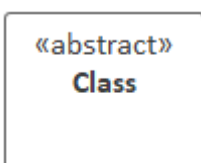


Figure 95 shows a package with name “Package2”. A package is a namespace for its members. In this example the member belonging to *Package2* is class *Class2*.

Figure 95 Package

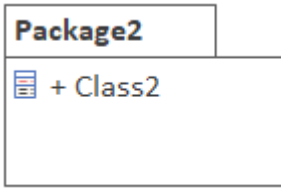
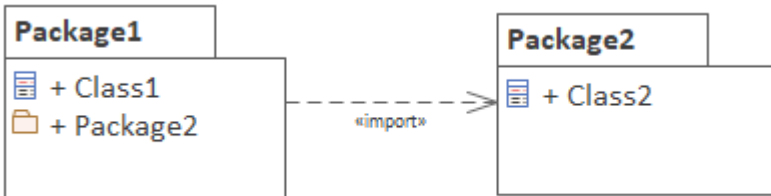


Figure 96 shows that all elements in *Package2* are imported into the namespace defined by *Package1*. This is a special dependency relationship between the two packages with stereotype <<import>>.

Figure 96 Imported Package



An enumeration is a data type whose values are enumerated as literals. Figure 97 shows an enumeration with name “Enumeration1”. It contains two literal values, “a” and “b”. It is a class with stereotype <<enumeration>>. The literals owned by the enumeration are ordered.

Figure 97 Enumeration⁵²

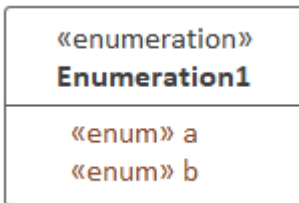


Figure 98 show the definition of the data type with name “DataType1”. A data type is a type whose instances are identified only by their value. It is a class with stereotype <<dataType>>.

Figure 98 Data Type



Figure 99 shows a primitive data type with name “int”. Primitive data types are predefined data types, without any substructure. The primitive data types are defined outside UML.

Figure 99 Primitive Data Type

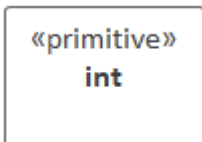


Figure 100 shows how a note can be attached to an element, in this example to class “Class1”.

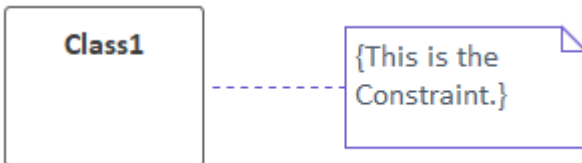
⁵² In Enterprise Architect the single enumeration values also have a stereotype <<enum>>, each.

Figure 100 Note



Figure 101 shows how a constraint is attached to an element, in this example to class “Class1”.

Figure 101 Constraint



ii. NOTES TO GRAPHICAL REPRESENTATION

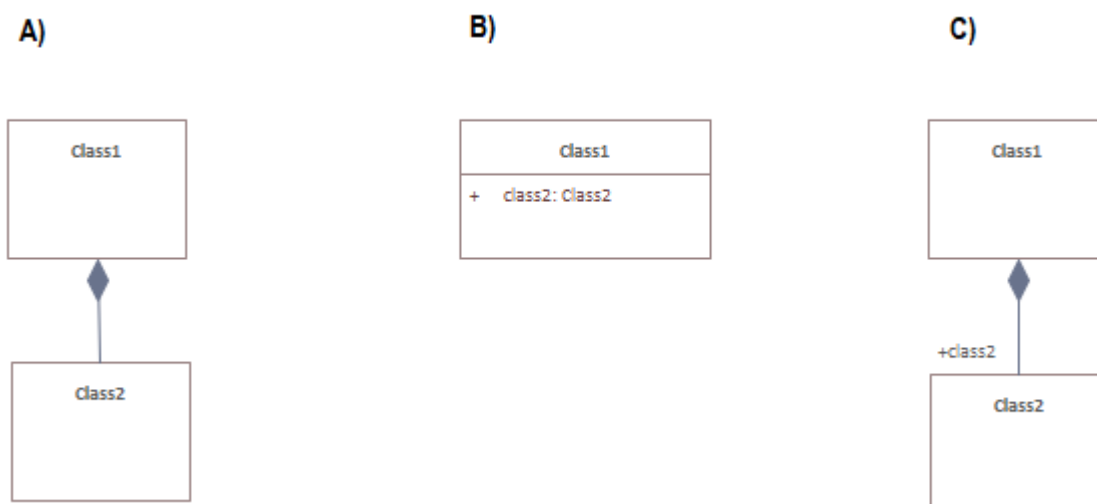
In the following specific graphical modelling rules used in this specification are explained that are not included in this form in [47].

Figure 102 shows different graphical representations of a composition (composite aggregation). In Variant A) a relationship with a filled aggregation diamond is used. In Variant B) an attribute with the same semantics is defined. And in Variant C) the implicitly assumed default name of the attribute in Variant A) is explicitly stated as such. In this document notation B) is used.

As a default it is assumed that only the end member of the association is navigable, i.e. it is possible to navigate from an instance of *Class1* to the owned instance of *Class2* but not vice versa. If there is no name for the end member of the association given then it is assumed that the name is identical to the class name but starting with a small letter – compare to Variant C).

Class2 instance does only exist if parent object of type *Class1* exists.

Figure 102 Graphical Representations of Composite Aggregation/Composition



In Figure 103 different representations of a shared aggregation are shown. In a shared aggregation a *Class2* instance can exist independent of the existence of an *Class1* instance. It is just referencing the instances of *Class2*. In Variant B) an attribute with the same semantics is defined. The reference is denoted by a star added after the type of the attribute.

As a default it is assumed that only the end member of the aggregation association is navigable, i.e. it is possible to navigate from an instance of *Class1* to the owned instance of *Class2* but not vice versa. Otherwise Variant B) would not be identical to Variant A).

A speciality in Figure 103 is that the aggregated instances are referables in the sense of the Asset Administration Shell metamodel (i.e. they inherit from the predefined abstract class “Referable”). This is why Variant B) is identical to Variant A). This would not be the case for non-referable elements in the metamodel. The structure of a reference to a model element of the Asset Administration Shell is explicitly defined. A model reference consists of an ordered list of keys. The last key in the key chain shall reference an instance of type *Class2* (i.e. Reference/type equal to “Class2”).

Figure 103 Graphical Representation of Shared Aggregation

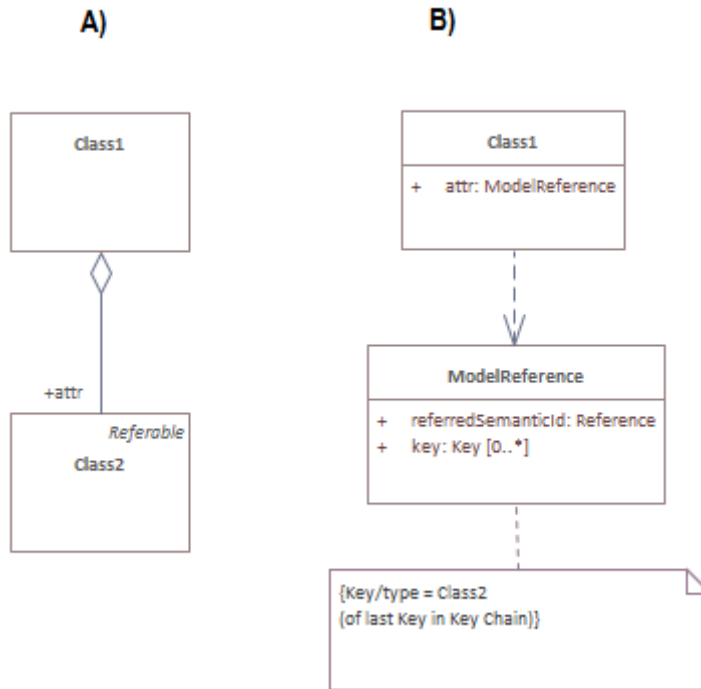
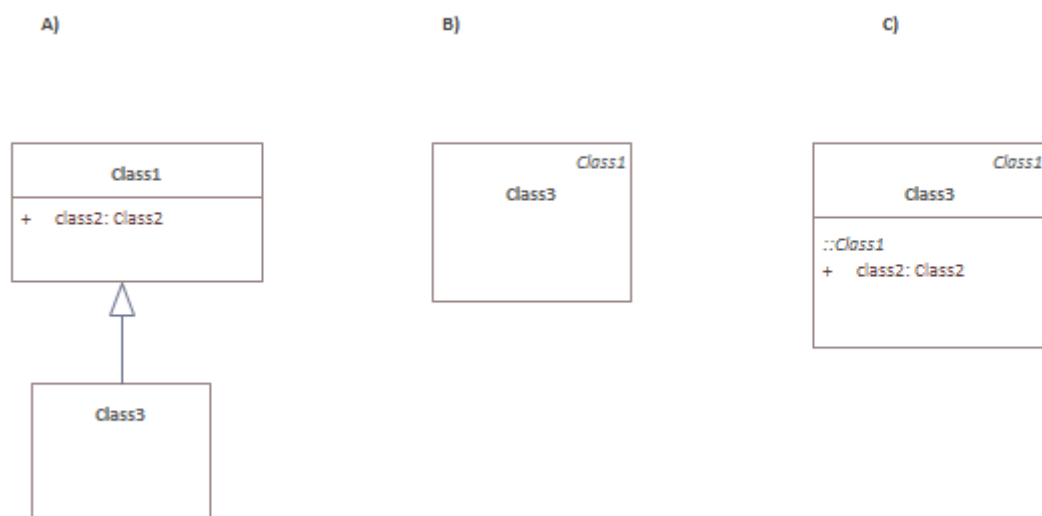


Figure 104 show different graphical representations of generalization. Variant A) is the classical graphical representation as defined in [47]. Variant B) is a short form if *Class1* is not on the same diagram. To see from which class *Class3* is inheriting the name of the class is depicted in the upper right corner.

Variant C) is not only showing from which class *Class3* instances are inheriting but also what they are inheriting. This is depicted by the class name it is inheriting from followed by “::” and then the list of all inherited elements – here attribute *class2*. Typically, the inherited elements are not shown.

Figure 104 Graphical Representation of Generalization/Inheritance



In Figure 105 different graphical notations for enumerations in combination with inheritance are shown. In Variant A) enumeration “Enumeration1” additionally contains the literals as defined by “Enumeration2”. Note: the direction of inheritance is opposite to the one for class inheritance. This can be seen in Variant C) that defines the same enumerations but without inheritance. In Variant B) another graphical notation is shown that makes it visible which literals are inherited by which enumeration. The literals within an enumeration are ordered so the order of classes it is inheriting from is important.

Figure 105 Graphical Representation for Enumeration with Inheritance

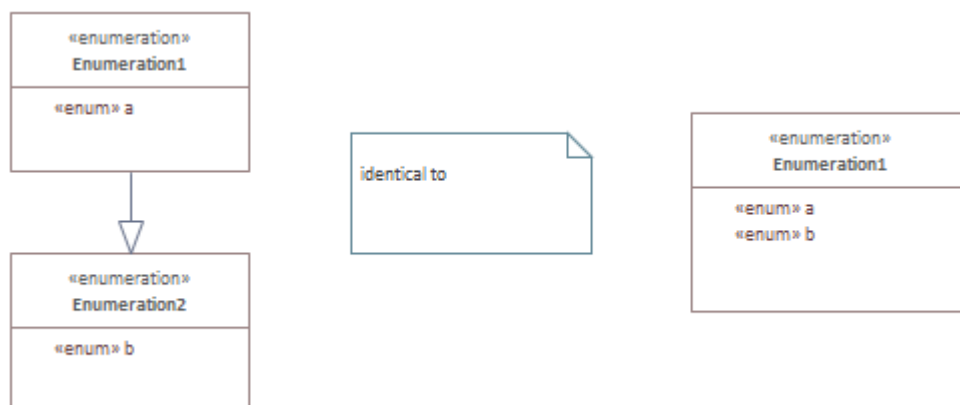
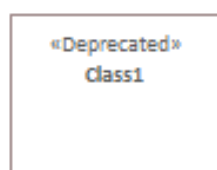


Figure 106 Graphical Representation for deprecated classes



In Figure 106 a class being deprecated is shown. The class is marked by the stereotype „Deprecated“.

ANNEX E. METAMODEL UML WITH INHERITED ATTRIBUTES

In this annex some UML diagrams are shown together with all attributes inherited for better overview.

Note: The abstract classes are numbered h0_, h1_ etc. but Aliases are defined for them without this prefix. The reason for this naming is that in the tooling used for UML modelling (Enterprise Architect) no order for inherited classes can be defined, they are ordered in an alphabetical way.

Figure 107 Core Model with Inherited Attributes

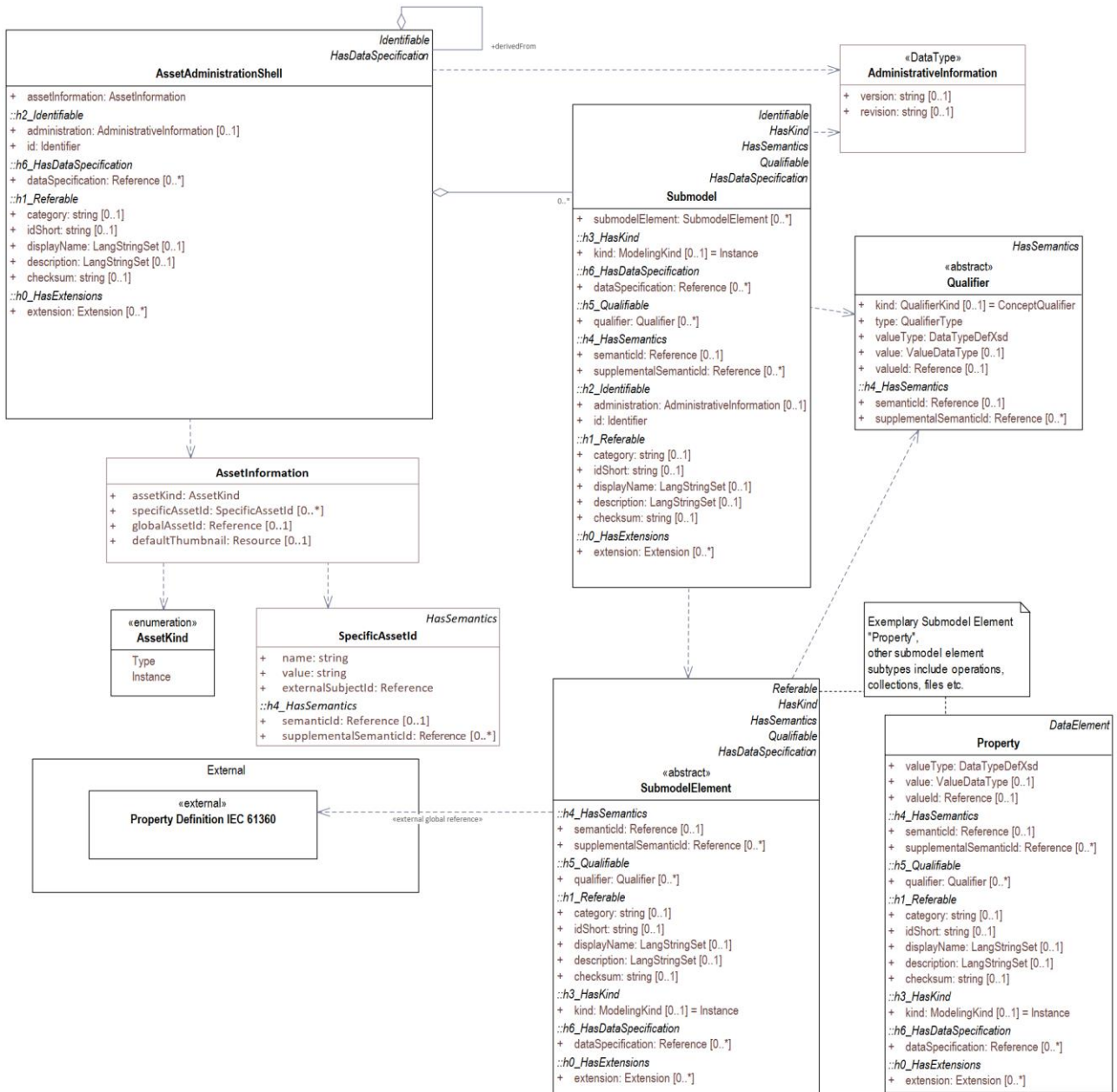
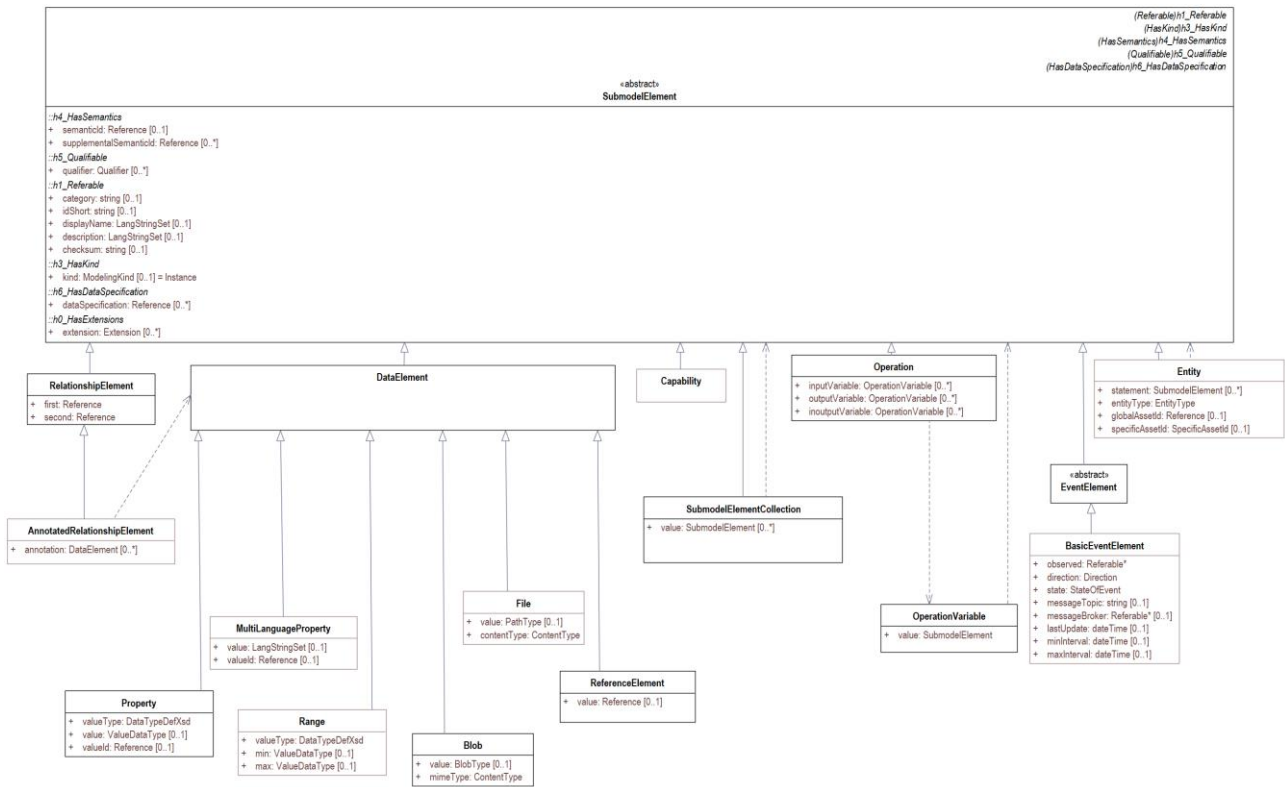


Figure 108 Model for Submodel Elements with Inherited Inheritance



ANNEX F. METAMODEL CHANGES

i. GENERAL

In this annex the changes from version to version of the metamodel are listed together with major changes in overall document. Non-backward compatible changes (nc) are marked as such.

nc="x" means not backward compatible, if no value is added in table then the change is backward compatible.

nc="(x)" means that change made was implicitly contained or stated in document before but now being formalized. Therefore, the change is considered to be backward compatible.

Changes for the security part of the metamodel are listed in separate tables.

ii. CHANGES V3.0RC02 VS. V2.0.1

A. METAMODEL CHANGES V3.0RC02 VS. V2.0.1 W/O SECURITY PART

Note to reader: if you already implemented the changes done in V3.0RC01 then refer to Annex iii. This Annex is for reader familiar with V2.0.x only.

Major changes:

- CHANGED: Split of SubmodelElementCollection into SubmodelElementList (with orderRelevant) and SubmodelElementCollection
- CHANGED: Adding reference type and referredSemanticId to Reference; Local and Parent attributes removed from Reference. Logical enumeration concept updated. Some renaming and some new enumerations. Adding constraint for references.
- CHANGED: Reference/type now as optional part of string serialization of reference
- CHANGED: idType from identifier removed, ID now string
- CHANGED: idShort of Referable now optional + Constraints added with respect to ID and idShort, includes that idShort of Submodels etc. do not need to be unique in the context of an AssetAdministrationShell any longer
- CHANGED: semanticId not mandatory any longer for SubmodelElement
- CHANGED: Revised concept on handling of Asset and assetIdentificationModel (assetInformation), Asset removed, no Asset/billOfMaterial any longer. Specific asset IDs added.
- REMOVED: ConceptDictionaries removed, not supported any longer
- REMOVED: Views removed, not supported any longer
- NEW: Event and BasicEvent updated and renamed to EventElement and BasicEventElement
- NEW: Checksum introduced for Referables
- REMOVED: security attribute removed from Asset Administration Shell but Access Control still part of the specification
- ENHANCED: DataTypeIEC61360 extended with values for IRI, IRDI, BLOB, FILE + corresponding new constraints added
- ENHANCED: Removed and splitted into DataTypeDefXsd and DataTypeDefRdf. Some types excluded and not supported
- CHANGED: Extracted and not part of this specification any longer: mapping rules for different serializations + Schemata + Example in different serializations
- EDITORIAL: Text updated, no kind column any longer in class tables, instead notation of ModelReference<{Referable}>. New table for Primitives/Data Types
- EDITORIAL: New Clause "Introduction"
- EDITORIAL: New Clause "Matching strategies for semantic identifiers"
- Constraints implicitly contained in text were formalized and numbered (normative)
- NEW: Environment explicitly part of UML (was part of serializations from the beginning)

- NEW: supplemental Semantic IDs
- NEW: Qualifier/kind (TemplateQualifier, ConceptQualifier, ValueQualifier)

Bugfixes:

- bugfix annotation AnnotatedRelationship is of type aggr and not ref* (diagram was correct)
- bugfix specification of ValueList and ValueReferencePairType, no data types, normal classes
- bugfix table specifications w.r.t. kind of attribute (from aggr to attr – column kind was removed, see above)
- bugfix data type specification LangStringSet (no diagram and table any longer)
- bugfix enumeration ReferableElements, no ConceptDictionary any longer + adding new elements like new submodel elements SubmodelElementList. Note: ReferableElements was substituted by AasSubmodelElements and Aas Identifiables.
- Entity/globalAssetId diagram (table was correct): **Type change from reference of Reference* to Reference**

Table 14 Changes w/o Security

nc	V3.0RC02 Change w.r.t. V2.0.1	Comment
	AdministrativeInformation	Bugfix: Added Stereotype "DataType"
	AnnotatedRelationship/annotation	Bugfix: Type changed from ModelReference<DataElement> to DataElement
	anySimpleTypeDef	Type removed, was not used in any class definition any longer, was mentioned in Text only.
x	Asset	Removed, asset referenced via AssetInformation/globalAssetId only
x	AssetAdministrationShell/asset	Removed, substituted by AssetAdministrationShell/assetInformation (but no reference any longer but an aggregation)
x	AssetAdministrationShell/conceptDictionaries	Removed
x	AssetAdministrationShell/security	Removed Note: Security is still part of the Asset Administration Shell, but the Asset Administration Shell and its elements are referenced from Security.
	AssetAdministrationShell/view	Removed, Views not longer supported
x	BasicEvent	Renamed to BasicEventElement
x	ConceptDictionary	Removed
x	Constraint	Abstract class removed. Formula now used in Security part only
(x)	DataTypeDef	Removed and splitted into DataTypeDefXsd and DataTypeDefRdf. Some types excluded and not supported (see notes in corresponding clause) Before: just string allowing any xsd simple type as string + added prefix xs: or rdf:, resp., to every value in enumeration

nc	V3.0RC02 Change w.r.t. V2.0.1	Comment
x	Entity/asset	Removed, substituted by Entity/globalAssetId and Entity/specificAssetId
x	Event	Renamed to EventElement
x	Extension/refersTo	Type changed from Reference to ModelReference
x	Extension/valueType	Type changed from DataTypeDef to DataTypeDefXsd
x	File/mimeType	Renamed to contentType + Type changed from MimeType to ContentType
x	Formula	Now abstract class Formula now used in Security part only
x	Formula/dependsOn	Removed since formula language not yet defined
x	Identifiable/identification	Removed Substituted by Identifiable/id
x	IdentifiableElements	Renamed to AasIdentifiables
x	Identifier	Type changed Before struct class with two attributes: id and idType. Now string data type only.
	IdentifierType	Enumeration removed because no idType any longer
x	Key/idType	removed
x	Key/local	Local attribute removed.
(x)	KeyElements	Renamed to KeyTypes The elements remain except for new SubmodelElementList, and renamed submodel elements Event and BasicEvent to EventElement and BasicEventElement
	KeyType	Enumeration removed because no Key/idType any longer
	LocalKeyType	Enumeration removed because no Key/idType any longer
x	MimeType	Type name changed to ContentType
	Property/valueType	Type changed from DataTypeDef to DataTypeDefXsd
x	Qualifiable/qualifier	Type changed from Constraint to Qualifier
	Qualifier	Does not inherit from abstract class "Constraint" any longer
	Qualifier/valueType	Type changed from DataTypeDef to DataTypeDefXsd

nc	V3.0RC02 Change w.r.t. V2.0.1	Comment
	Range/valueType	Type changed from DataTypeDef to DataTypeDefXsd
	Referable/idShort	Now optional, was mandatory
x	Referable/parent	Parent attribute removed.
x	ReferableElements	Substituted with enumeration AasSubmodelElements and AasIdentifiables
x	ReferableElements/AccessPermissionRule	Removed from Enumeration, AccessPermissionRule is not referable any longer Not part of new AasReferableNonIdentifiables
x	ReferableElement/BasicEvent	Renamed to BasicEventElement Now part of AasSubmodelElements
(x)	ReferablesElements/ConceptDictionary	Bugfix: ConceptDictionary removed from enumeration since ConceptDictionary not part of specification any longer Not part of new KeyTypes
x	ReferableElements/Event	Renamed to EventElement Now part of AasSubmodelElements
	RelationshipElement/first	Type changes from model reference Referable to Reference (global or model reference)
	RelationshipElement/second	Type changes from model reference Referable to Reference (global or model reference)
	ValueDataType	Before as specified via DataTypeDef, now any xsd atomic type as specified via DataTypeDefXsd
x	View	Removed

Table 15 New Elements in Metamodel w/o Security

nc	V3.0RC02 vs. V2.0.1 New Elements	Comment
	AasSubmodelElements	New enumeration used for References Before ReferableElements
	AasIdentifiables	New enumeration used for References, includes abstract Identifiable Before Identifiables
	AasReferableNonIdentifiables	New enumeration used for References
	AasReferables	New enumeration used for References, includes abstract Referable
x	AssetAdministrationShell/assetInformation	substitute for AssetAdministrationShell/asset but

nc	V3.0RC02 vs. V2.0.1 New Elements	Comment
		no reference any longer but an aggregation
	AssetInformation	with attributes/functionality from former class Asset because not specific to Asset but AAS
	AssetInformation/assetKind	Former Asset/assetKind
	AssetInformation/globalAssetId	Former Asset/identification/id
	AssetInformation/specificAssetId	Former Asset/assetIdentificationModel
	AssetInformation/thumbnail	Optional Attribute of new class AssetInformation that was not available in Asset class before
	BasicEventElement	Former name: BasicEvent Was part of non-normative part before
	BasicEventElement/direction	Former name: BasicEvent/observed Was part of non-normative part before
	BasicEventElement/lastUpdate	Was part of non-normative part before
	BasicEventElement/messageBroker	Was part of non-normative part before
	BasicEventElement/messageTopic	Was part of non-normative part before
	BasicEventElement/minInterval	Was part of non-normative part before
	BasicEventElement/maxInterval	Was part of non-normative part before
	BasicEventElement/observed	Was part of non-normative part before
	BasicEventElement/state	Was part of non-normative part before
	ContentType	Former name: MimeType
	dateTimeStamp	New data type for metamodel as used in EventPayload
	DataTypeDefRdf	Enumeration for types of Rdf + added prefix rdf: to every value in enumeration
	DataTypeDefXsd	Enumeration consisting of enumerations decimalBuildInTypes, durationBuildInTypes, PrimitiveTypes that correspond to anySimpleTypes of xsd. + added prefix xs: to every value in enumeration
	Direction	New Enumeration for BasicEventElement

nc	V3.0RC02 vs. V2.0.1 New Elements	Comment
	Environment	New class for entry point for Asset Administration Shells, submodels and concept descriptions.
	EventElement	Former name: Event
	EventPayload	New class for event payload
	EventPayload/observableSemanticId	Was part of non-normative part before
	EventPayload/payload	Was part of non-normative part before
	EventPayload/source	Was part of non-normative part before
	EventPayload/sourceSemanticId	Was part of non-normative part before
	EventPayload/subjectId	Was part of non-normative part before
	EventPayload/timestamp	Was part of non-normative part before
	Extension	New class, part of new abstract class HasExtensions
	FragmentKeys	New enumeration used for References
	GenericFragmentKeys	New enumeration used for References
	GenericGloballyIdentifiers	New enumeration used for References
	GloballyIdentifiables	New enumeration used for References
	HasExtensions	New abstract class, inherited by Referable
	HasSemantics/supplementalSemanticId	New attribute
	Identifiable/id	Substitute for Identifiable/identification
	IdentifierKeyValuePair	New class for AssetInformation/specificAssetId
	KeyTypes	Before: KeyElements New submodel element SubmodelElementList added, renamed submodel elements Event and BasicEvent to EventElement and BasicEventElement
	Qualifier/kind	New attribute for Qualifier
	QualifierKind	New enumeration for Qualifier/kind
	PrimitiveTypes	Enumeration for DataTypeDefXsd
	Referable/checksum	New optional attribute for all referables
	Referable/displayName	New optional attribute for all referables
	Reference/referredSemanticId	New optional attribute for Reference

nc	V3.0RC02 vs. V2.0.1 New Elements	Comment
x	Reference/type	New mandatory attribute for Reference
	ReferenceTypes	New enumeration for Reference/type
	StateOfEvent	New enumeration for BasicEventElement
	SpecificAssetId	New type for AssetInformation/specificAssetId
	SpecificAssetId/name	New type for AssetInformation/specificAssetId
	SpecificAssetId/value	New type for AssetInformation/specificAssetId
	SpecificAssetId/externalSubjectId	New type for AssetInformation/specificAssetId See Attribute Based Access Control (ABAC) for subject concept
	SubmodelElementElements	Enumeration for submodel elements (split of ReferableElements)
	SubmodelElementList	Before SubmodelElementCollection was used for lists and structs
	SubmodelElementList/orderRelevant	Similar to SubmodelElementCollection/ordered
	SubmodelElementList/value	Similar to SubmodelElementCollection/value but ordered and with all elements having the same semanticId
	SubmodelElementList/semanticIdListElement	Attribute of new class SubmodelElementList
	SubmodelElementList/typeValueListElement	Attribute of new class SubmodelElementList
	SubmodelElementList/valueTypeListElement	Attribute of new class SubmodelElementList

Table 16 New, Changed or Removed Constraints w/o Security

Nc	V3.0RC02 vs. V2.0.1	New, Update, Removed, Reformulated	Comment
	AASd-001	Removed	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. For namespace part see AASd-022

Nc	V3.0RC02 vs. V2.0.1	New, Update, Removed, Reformulated	Comment
53	AASd-003	Update	idShort is case-sensitive and not case-insensitive Constraint AASd-003: <i>idShort</i> of <i>Referables</i> shall be matched case-sensitive.
	AASd-005	Reformulated	Constraint AASd-005: If AdministrativeInformation/version is not specified than also AdministrativeInformation/revision shall be unspecified. This means, a revision requires a version. if there is no version there is no revision neither. Revision is optional.
	AASd-008	Removed	Constraint AASd-008: The submodel element value of an operation variable shall be of kind=Template.
	AASd-010	Renamed	Renamed and reformulated to AASs-010 (see NEW)
	AASd-011	Renamed	Renamed and reformulated to AASs-011 (see NEW)
	AASd-012	Reformulated	Constraint AASd-012: If both, the MultiLanguageProperty/value and the MultiLanguageProperty/valueId are present then for each string in a specific language the meaning must be the same as specified in MultiLanguageProperty/valueId
	AASd-014	Reformulated	Entity was changed Constraint AASd-014: Either the attribute globalAssetId or specificAssetId of an <i>Entity</i> must be set if <i>Entity/entityType</i> is set to " <i>SelfManagedEntity</i> ". They are not existing otherwise.
(x)	AASd-020	New	Constraint AASd-020: The value of Property/value shall be consistent to the data type as defined in Property/valueType.
(x)	AASd-021	New	Constraint AASd-021: Every qualifiable can only have one qualifier with the same <i>Qualifier/type</i> .
	AASd-023	Removed	No Asset any longer that can be referenced as alternative to global reference Constraint AASd-023: AssetInformation/globalAssetId either is a reference to an Asset object or a global reference.
x	AASd-027	New	Constraint AASd-027: <i>idShort</i> of <i>Referables</i> shall have a maximum length of 128 characters.
	AASd-050	New	Version information in data specification ID updated to /3/0/RC02. hasDataSpecification corrected to HasDataSpecification Constraint AASd-050: If the DataSpecificationContent DataSpecificationIEC61360 is used for an element then the value of hasDataSpecification/dataSpecification shall contain the global reference to the IRI of the corresponding data specification template https://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/3/0/RC02 .
(x)	AASd-050b	New	Constraint AASd-050b: If the DataSpecificationContent DataSpecificationPhysicalUnit is used for an element then the value of HasDataSpecification/dataSpecification shall contain the global reference to the IRI of the corresponding data specification template https://admin-shell.io/DataSpecificationTemplates/DataSpecificationPhysicalUnit0/3/0/RC02 .

⁵³ Every model valid for V3.0RC02 is still valid in V3.0RC01, however there might be implementations that need to be changed if they assumed that the user can type case-insensitive names and get all elements that match the name in an case-insensitive way.

Nc	V3.0RC02 vs. V2.0.1	New, Update, Removed, Reformulated	Comment
x	AASd-076	Removed	Substituted by AASc-002. Simplified, no reference to concept description
	AASd-077	New	Constraint AASd-077: The name of an extension within HasExtensions needs to be unique.
	AASd-090	Update	Exception: File and Blob data elements removed. Reformulated. Constraint AASd-090: For data elements category (inherited by Referable) shall be one of the following values: CONSTANT, PARAMETER or VARIABLE. Default: VARIABLE
x	AASd-076	Removed	Substituted by AASc-002. Simplified, no reference to concept description
	AASd-077	New	Constraint AASd-077: The name of an extension within HasExtensions needs to be unique.
	AASd-090	Update	Exception: File and Blob data elements removed. Reformulated. Constraint AASd-090: For data elements category (inherited by Referable) shall be one of the following values: CONSTANT, PARAMETER or VARIABLE. Default: VARIABLE
	AASd-100	New	Constraint AASd-100: An attribute with data type "string" is not allowed to be empty.
	AASd-107	New	Constraint AASd-107: If a first level child element in a SubmodelElementList has a semanticId it shall be identical to SubmodelElementList/semanticIdListElement.
	AASd-108	New	Constraint AASd-108: All first level child elements in a SubmodelElementList shall have the same submodel element type as specified in SubmodelElementList/typeValueListElement.
	AASd-109	New	Constraint AASd-109: If SubmodelElementList/typeValueListElement equal to Property or Range SubmodelElementList/valueTypeListElement shall be set and all first level child elements in the SubmodelElementList shall have the the value type as specified in SubmodelElementList/valueTypeListElement.
	AASd-114	New	Constraint AASd-114: If two first level child elements in a SubmodelElementList have a semanticId then they shall be identical.
	AASd-115	New	Constraint AASd-115: If a first level child element in a SubmodelElementList does not specify a semanticId then the value is assumed to be identical to SubmodelElementList/semanticIdListElement.
	AASd-116	New	Constraint AASd-116: "globalAssetId" (case-insensitive) is a reserved key. If used as value for SpecificAssetId/name IdentifierKeyValuePair/value shall be identical to AssetInformation/globalAssetId.
	AASd-117	New	Needed because Referable/idShort now optional Constraint AASd-117: idShort of non-identifiable Referables not equal to SubmodelElementList shall be specified (i.e. idShort is mandatory for all Referables except for SubmodelElementLists and all Identifiables).
	AASd-118	New	Because of new attribute supplementalSemanticId for HasSemantics Constraint AASd-118: If there is a supplemental semantic ID (HasSemantics/supplementalSemanticId) defined then there shall be also a main semantic ID (HasSemantics/semanticId).

Nc	V3.0RC02 vs. V2.0.1	New, Update, Removed, Reformulated	Comment
	AASd-119	New	New Qualifier/kind attribute Constraint AASd-119: If any Qualifier/kind value of a Qualifiable/qualifier is equal to TemplateQualifier and the qualified element inherits from "hasKind" then the qualified element shall be of kind Template (HasKind/kind = "Template").
	AASd-120	New	For new submodel element SubmodelElementList Constraint AASD-120: idShort of submodel elements within a SubmodelElementList shall not be specified.
	AASd-121	New	Constraint AASd-121: For References the type of the first key of Reference/keys shall be one of GloballyIdentifiables.
	AASd-122	New	Constraint AASd-122: For global references, i.e. References with Reference/type = GlobalReference, the type of the first key of Reference/keys shall be one of GenericGloballyIdentifiables.
	AASd-123	New	Constraint AASd-123: For model references, i.e. References with Reference/type = ModelReference, the type of the first key of Reference/keys shall be one of AasIdentifiables.
	AASd-124	New	Constraint AASd-124: For global references, i.e. References with Reference/type = GlobalReference, the last key of Reference/keys shall be either one of GenericGloballyIdentifiables or one of GenericFragmentKeys.
	AASd-125	New	Constraint AASd-125: For model references, i.e. References with Reference/type = ModelReference, with more than one key in Reference/keys the type of the keys following the first key of Reference/keys shall be one of FragmentKeys.
	AASd-126	New	Constraint AASd-126: For model references, i.e. References with Reference/type = ModelReference, with more than one key in Reference/keys the type of the last Key in the reference key chain may be one of GenericFragmentKeys or no key at all shall have a value out of GenericFragmentKey.
	AASd-127	New	Constraint AASd-127: For model references, i.e. References with Reference/type = ModelReference, with more than one key in Reference/keys a key with type FragmentReference shall be preceded by a key with type File or Blob. All other AAS fragments, i.e. type values out of AasSubmodelElements, do not support fragments.
	AAS-128	New	Constraint AASd-128: For model references, i.e. References with Reference/type = ModelReference, the Key/value of a Key preceded by a Key with Key/type=SubmodelElementList is an integer number denoting the position in the array of the submodel element list.

B. METAMODEL CHANGES V3.0RC02 VS. V2.0.1 – DATA SPECIFICATION IEC61360

Table 17 Changes w.r.t. Data Specification IEC61360

nc	V3.0RC02 Change w.r.t. V2.0.1	Comment
	DataSpecification	Stereotype <<Template>> added + does not inherit from Identifiable any longer because Data Specification are handled in a different way Some attributes are added to DataSpecification as new attributes like id, administration and description.
	DataSpecification/category	Removed, was inherited before by Identifiable
	DataSpecification/displayName	Removed, was inherited before by Identifiable
	DataSpecification/idShort	Removed, was inherited before by Identifiable
x	DataSpecificationIEC61360/value	Type changed from ValueDataType to string
	DataSpecificationIEC61360/valueId	Removed, the valueId is identical to the ID of the concept description
	DataSpecificationContent	Stereotype <<Template>> added
x	DataTypeIEC61360	Some new values were added: BLOB, FILE, HTML, IRDI. URL renamed to IRI. See separate entries for individual changes.
x	DataTypeIEC61360/URL	Renamed to IRI
	ValueList/valueReferencePairs	Bugfix, was ValueList/valueReferencePairTypes before
x	ValueReferencePair/value	Type changed from ValueDataType to string

Table 18 New Elements in Metamodel DataSpecification IEC61360

nc	V3.0RC02 vs. V2.0.1	Comment
	DataSpecification/administration	Was inherited before by Identifiable
	DataSpecification/id	Was inherited before by Identifiable
	DataSpecification/description	Was inherited before by Identifiable
	DataTypeIEC61360/BLOB	New value
	DataTypeIEC61360/FILE	New value
	DataTypeIEC61360/HTML	New value
	DataTypeIEC61360/IRDI	New value
	DataTypeIEC61360/IRI	Converted Iri to CamelCase and renamed to Iri from URL

Table 19 New, Changed or Removed Constraints Data Specification IEC61360

nc	V3.0RC02 vs. V2.0.1	New, Update, Removed, Reformulated	Comment
	AASc-002	New	Updated version of AASd-076, renamed to AASC-002 because applicable to data specification IEC61360 Constraint AASc-002: DataSpecificationIEC61360/preferredName shall be provided at least in English
(x)	AASc-003	New	Constraint AASc-003: For a ConceptDescription with category VALUE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) DataSpecificationIEC61360/value shall be set.
(x)	AASc-004	New	Constraint AASc-004: For a ConceptDescription with category PROPERTY or VALUE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/dataType is mandatory and shall be defined.
(x)	AASc-005	New	Constraint AASc-005: For a ConceptDescription with category REFERENCE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/dataType is STRING by default.
(x)	AASc-006	New	Constraint AASc-006: For a ConceptDescription with category DOCUMENT using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/dataType shall be one of the following values: STRING or URL.
(x)	AASc-007	New	Constraint AASc-007: For a ConceptDescription with category QUALIFIER_TYPE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/dataType is mandatory and shall be defined.
(x)	AASc-008	New	Constraint AASc-008: For a ConceptDescriptions except for a ConceptDescription of category VALUE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/definition is mandatory and shall be defined at least in English.
(x)	AASc-009	New	Constraint AASc-009: If DataSpecificationIEC61360/dataType one of: INTEGER_MEASURE, REAL_MEASURE, RATIONAL_MEASURE, INTEGER_CURRENCY, REAL_CURRENCY, then DataSpecificationIEC61360/unit or DataSpecificationIEC61360/unitId shall be defined.
(x)	AASc-010	New	Constraint AASc-010: If DataSpecificationIEC61360/value is not empty then DataSpecificationIEC61360/valueList shall be empty and vice versa

C. METAMODEL CHANGES V3.0RC02 VS. V2.0.1 – SECURITY PART

Changes:

- Removed: Deprecated: policy decision point, policy enforcement point and policy information points are not part of information model but of server infrastructure hosting the Asset Administration Shells
- Removed: Certificate Handling not part of information model but of server infrastructure hosting the Asset Administration Shells

Table 20 Changes w.r.t. Security

nc	V3.0RC02 Change w.r.t. V2.0.1	Comment
x	AccessControlPolicyPoints/policyAdministrationpoint	Type changed from PolicyAdministrationPoint to AccessControl
x	AccessControlPolicyPoints/policyDecisionPoint	Removed
x	AccessControlPolicyPoints/policyEnforcementPoint	Removed
x	AccessControlPolicyPoints/policyInformationPoint	Removed
x	AccessPermissionRule	Does not inherit from Referable any longer Does not inherit from Qualifiable any longer
x	BlobCertificate	Removed
x	Certificate	Removed
x	Formula	Now abstract class, only used in security part now (not used in Qualifiables any longer)
x	Formula/dependsOn	Removed attribute
x	PolicyAdministrationPoint	Removed
x	policyDecisionPoint	Removed
x	policyEnforcementPoint	Removed
x	policyInformationPoints	Removed
x	Security/certificate	Removed
x	Security/requiredCertificateExtension	Removed

Table 21 New Elements in Metamodel Security

nc	V3.0RC02 vs. V2.0.1	Comment
	AccessPermissionRule/constraint	Substitute for inherited attributes from Qualifiable

Table 22 New, Changed or Removed Constraints Security

nc	V3.0RC02 vs. V2.0.1	New, Update, Removed, Reformulated	Comment
	AASd-015	Removed	Renamed to AASs-015 (see NEW)
	AASs-009	Removed	Removed since class PolicyAdministrationPoint was removed

nc	V3.0RC02 vs. V2.0.1	New, Update, Removed, Reformulated	Comment
			Constraint AASs-009: Either there is an external policy administration point endpoint defined (PolicyAdministrationPoint/externalPolicyDecisionPoints=true) or the AAS has its own access control
	AASs-010	NEW	Reformulation of AASd-010 Constraint AASs-010: The property referenced in Permission/permission shall have the category "CONSTANT".
	AASs-011	NEW	Reformulation of AASd-011 Constraint AASs-011: The property referenced in Permission/permission shall be part of the submodel that is referenced within the "selectablePermissions" attribute of "AccessControl".
	AASs-015	NEW	Constraint AASs-015: Every data element in SubjectAttributes/subjectAttributes shall be part of the submodel that is referenced within the "selectableSubjectAttributes" attribute of "AccessControl".

iii. CHANGES V3.0RC02 VS. V3.0RC01

A. METAMODEL CHANGES V3.0RC02 VS. V3.0RC01 W/O SECURITY PART

Major changes:

- CHANGED: Split of SubmodelElementCollection into SubmodelElementList (with orderRelevant) and SubmodelElementCollection
- CHANGED: Adding reference type and referredSemanticId to Reference; Local and Parent attributes removed from Reference. Logical enumeration concept updated. Some renaming. Adding constraint for references.
- CHANGED: Reference/type now as optional part of string serialization of reference
- CHANGED: idType from identifier removed, ID now string.
- CHANGED: idShort of Referable now optional + Constraints added with respect to id and idShort
- REMOVED: AssetInformation/billOfMaterial removed
- REMOVED: Asset removed
- REMOVED: Views removed, not supported any longer
- NEW: Event and BasicEvent updated and renamed to EventElement and BasicEventElement
- NEW: Checksum introduced for Referables
- REMOVED: security attribute removed from Asset Administration Shell but Access Control still part of the specification
- DataTypeIEC61360 extended with values for IRI, IRDI, BLOB, FILE + corresponding new constraints added
- ENHANCED: Removed and splitted into DataTypeDefXsd and DataTypeDefRdf. Some types excluded and not supported
- CHANGED: Extracted and not part of this specification any longer: mapping rules for different serializations + Schemata + Example in different serializations
- EDITORIAL: Text updated, no kind column any longer in class tables, instead notation of ModelReference<{Referable}>. New table for Primitives/Data Types
- EDITORIAL: New Clause “Introduction”
- EDITORIAL: New Clause “Matching strategies for semantic identifiers”
- NEW: Environment
- NEW: supplemental Semantic IDs
- NEW: Qualifier/kind
- CHANGED: Renaming of IdentifierKeyValuePair used in AssetInformation to SpecificAssetId

Bugfixes:

- bugfix annotation AnnotatedRelationship is of type aggr and not ref* (diagram was correct)
- bugfix specification of ValueList and ValueReferencePairType, no data types, normal classes
- bugfix table specifications w.r.t. kind of attribute (from aggr to attr – column kind was removed, see above)
- bugfix data type specification LangStringSet (no diagram and table any longer)
- bugfix enumeration ReferableElements, no ConceptDictionary any longer + adding new elements like new submodel elements SubmodelElementList
- Entity/globalAssetId diagram (table was correct): Type change from reference of Reference to Reference (from Reference* to Reference)

Table 23 Changes w/o Security

nc	V3.0RC02 Change w.r.t. V3.0RC01	Comment
	AdministrativeInformation	Bugfix: Added Stereotype “DataType”

nc	V3.0RC02 Change w.r.t. V3.0RC01	Comment
	AnnotatedRelationship/annotation	Type changed from ModelReference<DataElement> to DataElement
x	Asset	Removed, asset referenced via globalAssetId only
x	AssetAdministrationShell/security	Removed Note: Security is still part of the Asset Administration Shell, but the Asset Administration Shell and its elements are referenced from Security.
	AssetAdministrationShell/view	Removed, Views not longer supported
x	AssetInformation/billOfMaterial	Removed
x	AssetInformation/defaultThumbnail	Type changed from File to Resource
x	AssetInformation/specificAssetId	Type changed from IdentifierKeyValuePair to SpecificAssetId
x	BasicEvent	Renamed to BasicEventElement
x	Constraint	Abstract class removed. Formula now used in Security part only
(x)	DataTypeDef	Splitted into DataTypeDefXsd and DataTypeDefRdf. Some types excluded and not supported (see notes in corresponding clause) Before: just string allowing all anySimpleTypes of xsd and langString of rdf
	Entity/globalAssetId	Bugfix: Type change from reference of Reference to Reference (from Reference* to Reference)
x	Event	Renamed to EventElement
x	Extension/refersTo	Type changed from Reference to ModelReference
x	File/mimeType	Renamed to contentType + Type name changed from MimeType to ContentType
x	Formula	Now abstract class Formula now used in Security part only

nc	V3.0RC02 Change w.r.t. V3.0RC01	Comment
x	Formula/dependsOn	Removed since formula language not yet defined
x	Identifiable/identification	Removed Substituted by Identifiable/id
(x)	IdentifiableElements	Renamed to AasIdentifiables
x	Identifier	Type changed Before struct class with two attributes: id and idType. Now string data type only.
x	IdentifierKeyValuePair	Renamed to SpecificAssetId and change of attribute "key" to "name"
	IdentifierType	Enumeration removed because no idType any longer
x	Key/idType	removed
(x)	KeyElements	Renamed to KeyTypes The elements remain except for new SubmodelElementList, and renamed submodel elements Event and BasicEvent to EventElement and BasicEventElement
	KeyType	Enumeration removed because no Key/idType any longer
	LocalKeyType	Enumeration removed because no Key/idType any longer
x	MimeType	Type name changed to ContentType
	Property/valueType	Type changed from DataTypeDef to DataTypeDefXsd
x	Qualifiable/qualifier	Type changed from Constraint to Qualifier
	Qualifier	Does not inherit from abstract class "Constraint" any longer
	Qualifier/valueType	Type changed from DataTypeDef to DataTypeDefXsd
	Range/valueType	Type changed from DataTypeDef to DataTypeDefXsd
	Referable/idShort	Now optional, was mandatory
x	ReferableElements	Substituted with enumeration AasSubmodelElements and AasIdentifiables

nc	V3.0RC02 Change w.r.t. V3.0RC01	Comment
x	ReferableElements/AccessPermissionRule	Removed from Enumeration, AccessPermissionRule is not referable any longer Not part of new AasReferableNonIdentifiabiles
x	ReferableElement/BasicEvent	Renamed to BasicEventElement Now part of AasSubmodelElements
(x)	ReferablesElements/ConceptDictionary	Bugfix: ConceptDictionary removed from enumeration since ConceptDictionary not part of specification any longer Not part of new KeyTypes
x	ReferableElements/Event	Renamed to EventElement Now part of AasSubmodelElements
	RelationshipElement/first	Type changes from model reference Referable to Reference (global or model reference)
	RelationshipElement/second	Type changes from model reference Referable to Reference (global or model reference)
	ValueDataType	Before as specified via DataTypeDef, now any xsd atomic type as specified via DataTypeDefXsd + Prefix xs: added to every value in list
x	ValueList/valueReferencePairType	Bugfix: renamed to ValueList/valueReferencePairs
x	View	removed

Table 24 New Elements in Metamodel w/o Security

nc	V3.0RC02 vs. V2.0RC01 New Elements	Comment
	AasSubmodelElements	New enumeration used for References Before ReferableElements
	AasIdentifiabiles	New enumeration used for References, includes abstract Identifiable Before Identifiabiles
	AasReferableNonIdentifiabiles	New enumeration used for References
	AasReferables	New enumeration used for References, includes abstract Referable

nc	V3.0RC02 vs. V2.0RC01 New Elements	Comment
	BasicEventElement	Former name: BasicEvent
	BasicEventElement/direction	Former name: BasicEvent/observed
	BasicEventElement/lastUpdate	
	BasicEventElement/messageBroker	
	BasicEventElement/messageTopic	
	BasicEventElement/minInterval	
	BasicEventElement/maxInterval	
	BasicEventElement/observed	
	BasicEventElement/state	
	ContentType	Former name: MimeType
	DataTypeDefRdf	Enumeration for types of Rdf + added prefix rdf: to every value in enumeration
	DataTypeDefXsd	Enumeration consisting of enumerations decimalBuildInTypes, durationBuildInTypes, PrimitiveTypes that correspond to anySimpleTypes of xsd. + added prefix xs: to every value in enumeration
	dateTimeStamp	New data type for metamodel as used in EventPayload
	decimalBuildInTypes	Enumeration for DataTypeDef
	Direction	New Enumeration for BasicEventElement
	durationBuildInTypes	Enumeration for DataTypeDef
	Environment	New class for entry point for Asset Administration Shells, submodels and concept descriptions.
	EventElement	Former name: Event
	EventPayload	New class for event payload
	EventPayload/observableReference	
	EventPayload/observableSemanticId	
	EventPayload/payload	
	EventPayload/source	
	EventPayload/sourceSemanticId	
	EventPayload/subjectId	
	EventPayload/timestamp	
	EventPayload/topic	

nc	V3.0RC02 vs. V2.0RC01 New Elements	Comment
	File/contentType	Former name: mimeType
	FragmentKeys	New enumeration used for References
	GenericFragmentKeys	New enumeration used for References
	GenericGloballyIdentifiers	New enumeration used for References
	GloballyIdentifiables	New enumeration used for References
	HasSemantics/supplementalSemanticId	New attribute
	Identifiable/id	Substitute for Identifiable/identification
	KeyTypes	Before: KeyElements New submodel element SubmodelElementList added, renamed submodel elements Event and BasicEvent to EventElement and BasicEventElement
	ModelReference	New class inheriting from Reference
x	Reference/type	New mandatory attribute of Reference
	Reference/referredSemanticId	New optional attribute of Reference
	PrimitiveTypes	Enumeration for DataTypeDefXsd
	Qualifier/kind	New attribute for Qualifier
	QualifierKind	New enumeration for Qualifier/kind
	Referable/checksum	
	SpecificAssetId	Before: IdentifierKeyValuePair, was renamed
	SpecificAssetId/name	Before: IdentifierKeyValuePair/key, was renamed
	SpecificAssetId/value	Before: IdentifierKeyValuePair/value
	SpecificAssetId/externalSubjectId	Before: IdentifierKeyValuePair/externalSubjectId
	StateOfEvent	New enumeration for BasicEventElement
	SubmodelElementElements	Enumeration for submodel elements (split of ReferableElements into SubmodelElementElements and IdentifiableElements)
	SubmodelElementList	Before SubmodelElementCollection was used for lists and structs
	SubmodelElementList/orderRelevant	Similar to SubmodelElementCollection/ordered
	SubmodelElementList/value	Similar to SubmodelElementCollection/value but ordered and with all elements having the same semanticId

nc	V3.0RC02 vs. V2.0RC01 New Elements	Comment
	SubmodelElementList/semanticIdListElement	Attribute for new class SubmodelElementList
	SubmodelElementList/typeValueListElement	Attribute for new class SubmodelElementList
	SubmodelElementList/valueTypeListElement	Attribute for new class SubmodelElementList

Table 25 New, Changed or Removed Constraints w/o Security

Nc	V3.0RC02 vs. V3.0RC01	New, Update, Removed, Reformulated	Comment
⁵⁴	AASd-003	Update	idShort is case-sensitive and not case-insensitive Constraint AASd-003: <i>idShort</i> of <i>Referables</i> shall be matched case-sensitive.
	AASd-005	Reformulated	Constraint AASd-005: If AdministrativeInformation/version is not specified than also AdministrativeInformation/revision shall be unspecified. This means, a revision requires a version. if there is no version there is no revision neither. Revision is optional.
	AASd-008	Removed	Constraint AASd-008: The submodel element value of an operation variable shall be of kind=Template.
	AASd-023	Removed	No Asset any longer that can be referenced as alternative to global reference Constraint AASd-023: AssetInformation/globalAssetId either is a reference to an Asset object or a global reference.
	AASd-026	Removed	SubmodelElementCollection was split into SubmodelElementList and SubmodelElementRecord. No attribute allowDuplicates any longer. Constraint AASd-026: If allowDuplicates==false then it is not allowed that the collection contains several elements with the same semantics (i.e. the same semanticId).
x	AASd-027	New	Constraint AASd-027: <i>idShort</i> of <i>Referables</i> shall have a maximum length of 128 characters.
	AASd-050	Update	Version information in data specification ID updated to /3/0/RC02. hasDataSpecification corrected to HasDataSpecification Constraint AASd-050: If the <i>DataSpecificationContent</i> <i>DataSpecificationIEC61360</i> is used for an element then the value of <i>HasDataSpecification/dataSpecification</i> shall contain the global reference to the IRI of the corresponding data specification template <i>https://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/3/0/RC02</i> .

⁵⁴ Every model valid for V3.0RC02 is still valid in V3.0RC01, however there might be implementations that need to be changed if they assumed that the user can type case-insensitive names and get all elements that match the name in an case-insensitive way.

Nc	V3.0RC02 vs. V3.0RC01	New, Update, Removed, Reformulated	Comment
(x)	AASd-050b	New	Constraint AASd-050b: If the DataSpecificationContent DataSpecificationPhysicalUnit is used for an element then the value of HasDataSpecification/dataSpecification shall contain the global reference to the IRI of the corresponding data specification template https://admin-shell.io/DataSpecificationTemplates/DataSpecificationPhysicalUnit0/3/0/RC02 .
	AASd-052a	Removed	removed, still recommended; would be renamed to AASc if still needed Constraint AASd-052a: If the semanticId of a Property references a ConceptDescription then the ConceptDescription/category shall be one of following values: VALUE, PROPERTY.
	AASd-052b	Removed	removed, still recommended; would be renamed to AASc if still needed Constraint AASd-052b: If the semanticId of a MultiLanguageProperty references a ConceptDescription then the ConceptDescription/category shall be one of following values: PROPERTY.
	AASd-053	Removed	removed, still recommended; would be renamed to AASc if still needed Constraint AASd-053: If the semanticId of a Range submodel element references a ConceptDescription then the ConceptDescription/category shall be one of following values: PROPERTY.
	AASd-054	Removed	removed, still recommended; would be renamed to AASc if still needed Constraint AASd-054: If the semanticId of a ReferenceElement submodel element references a ConceptDescription then the ConceptDescription/category shall be one of following values: REFERENCE.
	AASd-055	Removed	removed, still recommended; would be renamed to AASc if still needed Constraint AASd-055: If the semanticId of a RelationshipElement or an AnnotatedRelationshipElement submodel element references a ConceptDescription then the ConceptDescription/category shall be one of following values: RELATIONSHIP.
	AASd-056	Removed	removed, still recommended; would be renamed to AASc if still needed Constraint AASd-056: If the semanticId of a Entity submodel element references a ConceptDescription then the ConceptDescription/category shall be one of following values: ENTITY. The ConceptDescription describes the elements assigned to the entity via Entity/statement.
	AASd-057	Removed	removed, still recommended; would be renamed to AASc if still needed Constraint AASd-057: The semanticId of a File or Blob submodel element shall only reference a ConceptDescription with the category DOCUMENT.
	AASd-058	Removed	removed, still recommended; would be renamed to AASc if still needed

Nc	V3.0RC02 vs. V3.0RC01	New, Update, Removed, Reformulated	Comment
			Constraint AASd-058: The semanticId of a Capability submodel element shall only reference a ConceptDescription with the category CAPABILITY.
	AASd-059	Removed	<p>removed, still recommended; would be renamed to AASc if still needed</p> <p>SubmodelElementCollection was split into SubmodelElementList and SubmodelElementCollection. AASd-092 and AASd-093 contain it.</p> <p>Constraint AASd-059: If the semanticId of a SubmodelElementCollection references a ConceptDescription then the category of the ConceptDescription shall be COLLECTION or ENTITY.</p>
	AASd-060	Removed	<p>removed, still recommended; would be renamed to AASc if still needed</p> <p>Constraint AASd-060: If the semanticId of a Operation submodel element references a ConceptDescription then the category of the ConceptDescription shall be one of the following values: FUNCTION.</p>
	AASd-061	Removed	<p>removed, still recommended; would be renamed to AASc if still needed</p> <p>Constraint AASd-061: If the semanticId of a Event submodel element references a ConceptDescription then the category of the ConceptDescription shall be one of the following values: EVENT.</p>
	AASd-062	Removed	<p>removed, still recommended; would be renamed to AASc if still needed</p> <p>Constraint AASd-062: If the <i>semanticId</i> of a <i>Property</i> references a <i>ConceptDescription</i> then the <i>ConceptDescription/category</i> shall be one of following values: APPLICATION_CLASS.</p>
	AASd-063	Removed	<p>removed, still recommended; would be renamed to AASc if still needed</p> <p>Constraint AASd-063: If the semanticId of a Qualifier references a ConceptDescription then the ConceptDescription/category shall be one of following values: QUALIFIER.</p>
	AASd-064	Removed	<p>Removed because there are not VIEWS any longer</p> <p>Constraint AASd-064: If the <i>semanticId</i> of a <i>View</i> references a <i>ConceptDescription</i> then the category of the <i>ConceptDescription</i> shall be <i>VIEW_</i>.</p>
	AASd-065	Removed	<p>removed, still recommended; would be renamed to AASc if still needed</p> <p>Constraint AASd-065: If the semanticId of a Property or MultiLanguageProperty references a ConceptDescription with the category VALUE then the value of the property is identical to</p>

Nc	V3.0RC02 vs. V3.0RC01	New, Update, Removed, Reformulated	Comment
			DataSpecificationIEC61360/value and the valueId of the property is identical to DataSpecificationIEC61360/valueId.
	AASd-066	Removed	<p>removed, still recommended; would be renamed to AASc if still needed</p> <p>Update because of renaming of ValueReferencePairType into ValueReferencePair</p> <p>Constraint AASd-066: If the semanticId of a Property or MultiLanguageProperty references a ConceptDescription with the category PROPERTY and DataSpecificationIEC61360/valueList is defined the value and valueId of the property is identical to one of the value reference pair types references in the value list, i.e. ValueReferencePair/value or ValueReferencePair/valueId, resp.</p>
	AASd-067	Removed	<p>removed, still recommended; would be renamed to AASc if still needed</p> <p>Constraint AASd-067: If the semanticId of a MultiLanguageProperty references a ConceptDescription then DataSpecificationIEC61360/dataType shall be STRING_TRANSLATABLE.</p>
	AASd-068	Removed	<p>removed, still recommended; would be renamed to AASc if still needed</p> <p>Constraint AASd-068: If the semanticId of a Range submodel element references a ConceptDescription then DataSpecificationIEC61360/dataType shall be a numerical one, i.e. REAL_* or RATIONAL_*.</p>
	AASd-069	Removed	<p>removed, still recommended; would be renamed to AASc if still needed</p> <p>Constraint AASd-069: If the semanticId of a Range references a ConceptDescription then DataSpecificationIEC61360/levelType shall be identical to the set {Min, Max}.</p>
(x)	AASd-070	Renamed	Now AASc-004.
(x)	AASd-071	Renamed	Now AASc-005
(x)	AASd-072	Renamed	Now AASc-006.
(x)	AASd-073	Renamed	Now AASc-007
(x)	AASd-074	Renamed	Now AASc-008
	AASd-075	Removed	<p>Content now documented as separate constraints, see Annex 0 b</p> <p>Constraint AASd-075: For all ConceptDescriptions using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) values for the attributes not being marked as mandatory or optional in tables Table 8, Table 9, Table 10 and Table 11. depending on its category are ignored and handled as undefined.</p>
	AASd-076	Removed	Substituted by AASc-002. Simplified, no reference to concept description

Nc	V3.0RC02 vs. V3.0RC01	New, Update, Removed, Reformulated	Comment
	AASd-080	Removed	No <i>Key/type</i> GlobalReference any longer <u>Constraint AASd-080</u> : In case <i>Key/type == GlobalReference idType</i> shall not be any LocalKeyType (<i>IdShort, FragmentId</i>).
	AASd-081	Removed	No <i>Key/idType</i> any longer <u>Constraint AASd-081</u> : In case <i>Key/type==AssetAdministrationShell Key/idType</i> shall not be any LocalKeyType (<i>IdShort, FragmentId</i>).
	AASd-090	Update	Exception: File and Blob data elements removed. Reformulated. Constraint AASd-090: For data elements category (inherited by Referable) shall be one of the following values: CONSTANT, PARAMETER or VARIABLE. Default: VARIABLE
	AASd-092	Removed	removed, still recommended; would be renamed to AASc and updated if still needed SubmodelElementCollection was split into SubmodelElementList and SubmodelElementCollection (here: SubmodelElementCollection) Constraint AASd-092: If the semanticId of a SubmodelElementCollection with SubmodelElementCollection/allowDuplicates == false references a ConceptDescription then the ConceptDescription/category shall be ENTITY.
	AASd-093	Removed	removed, still recommended; would be renamed to AASc and updated if still needed SubmodelElementCollection was split into SubmodelElementList and SubmodelElementStruct (here: SubmodelElementList) Constraint AASd-093: If the semanticId of a SubmodelElementCollection with SubmodelElementCollection/allowDuplicates == true references a ConceptDescription then the ConceptDescription/category shall be COLLECTION.
	AASd-107	New	Constraint AASd-107: If a first level child element in a SubmodelElementList has a semanticId it shall be identical to SubmodelElementList/semanticIdListElement.
	AASd-108	New	Constraint AASd-108: All first level child elements in a SubmodelElementList shall have the same submodel element type as specified in SubmodelElementList/typeValueListElement.
	AASd-109	New	Constraint AASd-109: If SubmodelElementList/typeValueListElement equal to Property or Range SubmodelElementList/valueTypeListElement shall be set and all first level child elements in the SubmodelElementList shall have the the value type as specified in SubmodelElementList/valueTypeListElement.
	AASd-114	New	Constraint AASd-114: If two first level child elements in a SubmodelElementList have a semanticId then they shall be identical.
	AASd-115	New	Constraint AASd-115: If a first level child element in a SubmodelElementList does not specify a semanticId then the value is

Nc	V3.0RC02 vs. V3.0RC01	New, Update, Removed, Reformulated	Comment
			assumed to be identical to SubmodelElementList/semanticIdListElement.
	AASd-116	New	Constraint AASd-116: "globalAssetId" (case-insensitive) is a reserved key. If used as value for SpecificAssetId/name IdentifierKeyValuePair/value shall be identical to AssetInformation/globalAssetId.
	AASd-117	New	Needed because Referable/idShort now optional Constraint AASd-117: idShort of non-identifiable Referables not equal to SubmodelElementList shall be specified (i.e. idShort is mandatory for all Referables except for SubmodelElementLists and all Identifiables).
	AASd-118	New	Constraint AASd-118: If there is a supplemental semantic ID (HasSemantics/supplementalSemanticId) defined then there shall be also a main semantic ID (HasSemantics/semanticId).
	AASd-119	New	New Qualifier/kind attribute Constraint AASd-119: If any Qualifier/kind value of a Qualifiable/qualifier is equal to TemplateQualifier and the qualified element inherits from "hasKind" then the qualified element shall be of kind Template (HasKind/kind = "Template").
	AASd-120	New	For new submodel element SubmodelElementList Constraint AASD-120: idShort of submodel elements within a SubmodelElementList shall not be specified.
	AASd-121	New	Constraint AASd-121: For References the type of the first key of Reference/keys shall be one of GloballyIdentifiables.
	AASd-122	New	Constraint AASd-122: For global references, i.e. References with Reference/type = GlobalReference, the type of the first key of Reference/keys shall be one of GenericGloballyIdentifiables.
	AASd-123	New	Constraint AASd-123: For model references, i.e. References with Reference/type = ModelReference, the type of the first key of Reference/keys shall be one of AasIdentifiables.
	AASd-124	New	Constraint AASd-124: For global references, i.e. References with Reference/type = GlobalReference, the last key of Reference/keys shall be either one of GenericGloballyIdentifiables or one of GenericFragmentKeys.
	AASd-125	New	Constraint AASd-125: For model references, i.e. References with Reference/type = ModelReference, with more than one key in Reference/keys the type of the keys following the first key of Reference/keys shall be one of FragmentKeys.
	AASd-126	New	Constraint AASd-126: For model references, i.e. References with Reference/type = ModelReference, with more than one key in Reference/keys the type of the last Key in the reference key chain may be one of GenericFragmentKeys or no key at all shall have a value out of GenericFragmentKey.
	AASd-127	New	Constraint AASd-127: For model references, i.e. References with Reference/type = ModelReference, with more than one key in

Nc	V3.0RC02 vs. V3.0RC01	New, Update, Removed, Reformulated	Comment
			Reference/keys a key with type FragmentReference shall be preceded by a key with type File or Blob. All other AAS fragments, i.e. type values out of AasSubmodelElements, do not support fragments.
	AAS-128	New	Constraint AASd-128: For model references, i.e. References with Reference/type = ModelReference, the Key/value of a Key preceded by a Key with Key/type=SubmodelElementList is an integer number denoting the position in the array of the submodel element list.

B. METAMODEL CHANGES V3.0RC02 VS. V3.0RC01 – DATA SPECIFICATION IEC61360

Table 26 Changes w.r.t. Data Specification IEC61360

nc	V3.0RC02 Change w.r.t. V3.0RC01	Comment
	DataSpecification	Stereotype <<Template>> added + does not inherit from Identifiable any longer because Data Specification are handled in a different way Some attributes are added to DataSpecification as new attributes like id, administration and description.
	DataSpecification/category	Removed, was inherited before by Identifiable
	DataSpecification/displayName	Removed, was inherited before by Identifiable
	DataSpecification/idShort	Removed, was inherited before by Identifiable
	DataSpecificationIEC61360/unitId	Type changes from Reference to GlobalReference
x	DataSpecificationIEC61360/value	Type changed from ValueDataType to string
	DataSpecificationIEC61360/valueId	Removed, the valueId is identical to the ID of the concept description
	DataSpecificationContent	Stereotype <<Template>> added
x	DataTypeInfoEC61360	Some new values were added: BLOB, FILE, HTML, IRDI. URL renamed to IRI. See separate entries for individual changes.
x	DataTypeInfoEC61360/URL	Renamed to IRI
	ValueList/valueReferencePairs	Bugfix, was ValueList/valueReferencePairTypes before
x	ValueReferencePair/value	Type changed from ValueDataType to string

Table 27 New Elements in Metamodel DataSpecification IEC61360

nc	V3.0RC02	Comment
x	ValueReferencePair/valueId	Type changed from Reference to GlobalReference
	DataSpecification/administration	Was inherited before by Identifiable
	DataSpecification/id	Was inherited before by Identifiable
	DataSpecification/description	Was inherited before by Identifiable
	DataTypeInfoEC61360/BLOB	New value
	DataTypeInfoEC61360/FILE	New value
	DataTypeInfoEC61360/HTML	New value
	DataTypeInfoEC61360/IRDI	New value
	DataTypeInfoEC61360/IRI	Converted Iri to CamelCase and renamed to Iri from URL

Table 28 New, Changed or Removed Constraints Data Specification IEC61360

nc	V3.0RC02	New, Update, Removed, Reformulated	Comment
	AASc-002	New	Updated version of AASd-076, renamed to AASC-002 because applicable to data specification IEC61360 Constraint AASc-002: DataSpecificationIEC61360/preferredName shall be provided at least in English
(x)	AASc-003	New	Constraint AASc-003: For a ConceptDescription with category VALUE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) DataSpecificationIEC61360/value shall be set.
(x)	AASc-004	New	Constraint AASc-004: For a ConceptDescription with category PROPERTY or VALUE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/dataType is mandatory and shall be defined.
(x)	AASc-005	New	Constraint AASc-005: For a ConceptDescription with category REFERENCE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/dataType is STRING by default.
(x)	AASc-006	New	Constraint AASc-006: For a ConceptDescription with category DOCUMENT using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/dataType shall be one of the following values: STRING or URL.
(x)	AASc-007	New	Constraint AASc-007: For a ConceptDescription with category QUALIFIER_TYPE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0)

nc	V3.0RC02	New, Update, Removed, Reformulated	Comment
			- DataSpecificationIEC61360/dataType is mandatory and shall be defined.
(x)	AASc-008	New	Constraint AASc-008: For a ConceptDescriptions except for a ConceptDescription of category VALUE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/definition is mandatory and shall be defined at least in English.
(x)	AASc-009	New	Constraint AASc-009: If DataSpecificationIEC61360/dataType one of: INTEGER_MEASURE, REAL_MEASURE, RATIONAL_MEASURE, INTEGER_CURRENCY, REAL_CURRENCY, then DataSpecificationIEC61360/unit or DataSpecificationIEC61360/unitId shall be defined.
(x)	AASc-010	New	Constraint AASc-010: If DataSpecificationIEC61360/value is not empty then DataSpecificationIEC61360/valueList shall be empty and vice versa

C. METAMODEL CHANGES V3.0RC02 VS. V3.0RC01 – SECURITY PART

Changes:

- Removed: Deprecated: policy decision point, policy enforcement point and policy information points are not part of information model but of server infrastructure hosting the Asset Administration Shells
- Removed: Certificate Handling not part of information model but of server infrastructure hosting the Asset Administration Shells

Table 29 Changes w.r.t. Security

nc	V3.0RC02 Change w.r.t. V3.0RC01	Comment
x	AccessControlPolicyPoints/policyAdministrationpoint	Type changed from PolicyAdministrationPoint to AccessControl
x	AccessControlPolicyPoints/policyDecisionPoint	Removed
x	AccessControlPolicyPoints/policyEnforcementPoint	Removed
x	AccessControlPolicyPoints/policyInformationPoint	Removed
x	AccessPermissionRule	Does not inherit from Referable any longer Does not inherit from Qualifiable any longer
x	BlobCertificate	Removed
x	Certificate	Removed
x	Formula	Now abstract class, only used in security part now (not used in Qualifiables any longer)
x	Formula/dependsOn	Removed attribute

nc	V3.0RC02 Change w.r.t. V3.0RC01	Comment
x	PolicyAdministrationPoint	Removed
x	policyDecisionPoint	Removed
x	policyEnforcementPoint	Removed
x	policyInformationPoints	Removed
x	Security/certificate	Removed
x	Security/requiredCertificateExtension	Removed

Table 30 New Elements in Metamodel Security

nc	V3.0RC02 vs. V3.0RC01	Comment
	AccessPermissionRule/constraint	Substitute for inherited attributes from Qualifiable

Table 31 New, Changed or Removed Constraints Security

nc	V3.0RC02 vs. V3.0RC01	New, Update, Removed, Reformulated	Comment
	AASs-009	Removed	Removed since class PolicyAdministrationPoint was removed Constraint AASs-009: Either there is an external policy administration point endpoint defined (PolicyAdministrationPoint/externalPolicyDecisionPoints=true) or the AAS has its own access control
	AASs-015	Updated	Constraint AASs-015: Every data element in SubjectAttributes/subjectAttributes shall be part of the submodel that is referenced within the “selectableSubjectAttributes” attribute of “AccessControl”.

iv. CHANGES V3.0RC01 VS. V2.0.1

A. METAMODEL CHANGES V3.0RC01 W/O SECURITY PART

Major changes:

- idShort of Submodels etc. do not need to be unique in the context of an AssetAdministrationShell any longer
- Constraints implicitly contained in text were formalized and numbered
- Revised concept on handling of Asset and assetIdentificationModel (assetInformation)
- ConceptDictionaries not supported any longer
- semanticId not mandatory any longer for SubmodelElement
- More than one bill of material for assetInformation in Asset Administration Shell
- Local attribute in References removed
- Parent attribute in Referables removed

Table 32 Changes w.r.t. V2.0 w/o Security

nc	V3.0RC01 Change w.r.t. V2.0.1	Comment
	anySimpleTypeDef	Type removed, was not used in any class definition any longer, was mentioned in Text only.
x	AssetAdministrationShell/asset	Removed, substituted by AssetAdministrationShell/assetInformation (but no reference any longer but an aggregation)
x	Asset/assetKind	Attribute "assetKind" moved to AssetAdministrationShell/AssetInformation
x	Asset/assetIdentificationModel	Attribute "assetIdentificationModel " Removed, substituted by AssetInformation /IdentifierKeyValuePairs
x	Asset/billOfMaterial	Attribute "billOfMaterial" moved to AssetAdministrationShell/AssetInformation
x	AssetAdministrationShell/conceptDictionaries	Removed
	ConceptDescription/isCaseOf	Text changed, no global reference requested, just reference.
x	ConceptDictionary	Removed
x	Entity/asset	Removed, substituted by Entity/globalAssetId and Entity/specificAssetId
x	Key/local	Local attribute removed.
x	Referable/parent	Parent attribute removed.

Table 33 New Elements in Metamodel V3.0RC01 w/o Security

nc	V3.0RC01 vs. V2.0.1	Comment
x	AssetAdministrationShell/assetInformation	substitute for AssetAdministrationShell/asset but no reference any longer but an aggregation
	AssetInformation	with attributes/functionality from former class Asset because not specific to Asset but AAS
	AssetInformation/thumbnail	Optional Attribute of new class AssetInformation that was not available in Asset class before
x	Entity/globalAssetId	Substitute for Entity/asset (together with Entity/specificAssetId)
x	Entity/specificAssetId	Substitute for Entity/asset (together with Entity/globalAssetId)
	Extension	New class, part of new abstract class HasExtensions
	HasExtensions	New abstract class, inherited by Referable
	IdentifierKeyValuePair	New class for AssetInformation/specificAssetId
	Referable/displayName	New optional attribute for all referables

Table 34 New, Changed or Removed Constraints w/o Security

nc	V3.0RC01	New, Update, Removed, Reformulated	Comment
	AASd-001	Removed	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. For namespace part see AASd-022
x	AASd-002	Update	reformulated, formula added <i>idShort of Referables</i> shall only feature letters, digits, underscore ("_"); starting mandatory with a letter. I.e. [a-zA-Z][a-zA-Z0-9_]+
	AASd-010	Reformulated	Constraint AASd-010: The property has the category "CONSTANT". Reformulated to Constraint AASd-010: The property referenced in Permission/permission shall have the category "CONSTANT".
	AASd-011	Reformulated	Constraint AASd-011: The property referenced in Permission/permission shall be part of the submodel that is referenced within the "selectablePermissions" attribute of "AccessControl".
	AASd-012	Reformulated	Constraint AASd-012: If both, the MultiLanguageProperty/value and the MultiLanguageProperty/valueId are present then for each string in a specific language the meaning must be the same as specified in MultiLanguageProperty/valueId
	AASd-014	Reformulated	Entity was changed Constraint AASd-014: Either the attribute globalAssetId or specificAssetId of an <i>Entity</i> must be set if <i>Entity/entityType</i> is set to " <i>SelfManagedEntity</i> ". They are not existing otherwise.
(x)	AASd-020	New	Constraint AASd-020: The value of Property/value shall be consistent to the data type as defined in Property/valueType.
(x)	AASd-021	New	Constraint AASd-021: Every qualifiable can only have one qualifier with the same <i>Qualifier/type</i> .
(x)	AASd-022	New	Splitted part from AASd-001 Constraint AASd-022: idShort of non-identifiable referables shall be unique in its namespace.
(x)	AASd-026	New	Constraint AASd-026: If allowDuplicates==false then it is not allowed that the collection contains several elements with the same semantics (i.e. the same semanticId).
(x)	AASd-050	New	Constraint AASd-050: If the DataSpecificationContent DataSpecificationIEC61360 is used for an element then the value of hasDataSpecification/dataSpecification shall contain the global reference to the IRI of the corresponding data specification template http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0 .
(x)	AASd-051	New	Constraint AASd-051: A ConceptDescription shall have one of the following categories: VALUE, PROPERTY, REFERENCE,

nc	V3.0RC01	New, Update, Removed, Reformulated	Comment
			DOCUMENT, CAPABILITY, RELATIONSHIP, COLLECTION, FUNCTION, EVENT, ENTITY, APPLICATION_CLASS, QUALIFIER, VIEW. Default: PROPERTY.
(x)	AASd-052a	New	Constraint AASd-052a: If the semanticId of a Property references a ConceptDescription then the ConceptDescription/category shall be one of following values: VALUE, PROPERTY.
(x)	AASd-052b	New	Constraint AASd-052b: If the semanticId of a MultiLanguageProperty references a ConceptDescription then the ConceptDescription/category shall be one of following values: PROPERTY.
(x)	AASd-053	New	Constraint AASd-053: If the semanticId of a Range submodel element references a ConceptDescription then the ConceptDescription/category shall be one of following values: PROPERTY.
(x)	AASd-054	New	Constraint AASd-054: If the semanticId of a ReferenceElement submodel element references a ConceptDescription then the ConceptDescription/category shall be one of following values: REFERENCE.
(x)	AASd-055	New	Constraint AASd-055: If the semanticId of a RelationshipElement or an AnnotatedRelationshipElement submodel element references a ConceptDescription then the ConceptDescription/category shall be one of following values: RELATIONSHIP.
(x)	AASd-056	New	Constraint AASd-056: If the semanticId of a Entity submodel element references a ConceptDescription then the ConceptDescription/category shall be one of following values: ENTITY. The ConceptDescription describes the elements assigned to the entity via Entity/statement.
(x)	AASd-057	New	Constraint AASd-057: The semanticId of a File or Blob submodel element shall only reference a ConceptDescription with the category DOCUMENT.
(x)	AASd-058	New	Constraint AASd-058: The semanticId of a Capability submodel element shall only reference a ConceptDescription with the category CAPABILITY.
(x)	AASd-059	New	Constraint AASd-059: The semanticId of a SubmodelElementCollection submodel element shall only reference a ConceptDescription with the category COLLECTION or ENTITY.
(x)	AASd-060	New	Constraint AASd-060: If the semanticId of a Operation submodel element references a ConceptDescription then the category of the ConceptDescription shall be one of the following values: FUNCTION.
(x)	AASd-061	New	Constraint AASd-061: If the semanticId of a Event submodel element references a ConceptDescription then the category of the ConceptDescription shall be one of the following values: EVENT.
(x)	AASd-062	New	Constraint AASd-062: If the <i>semanticId</i> of a <i>Property</i> references a <i>ConceptDescription</i> then the <i>ConceptDescription/category</i> shall be one of following values: APPLICATION_CLASS.

nc	V3.0RC01	New, Update, Removed, Reformulated	Comment
(x)	AASd-063	New	Constraint AASd-063: If the semanticId of a Qualifier references a ConceptDescription then the ConceptDescription/category shall be one of following values: QUALIFIER.
(x)	AASd-064	New	Constraint AASd-064: If the semanticId of a View references a ConceptDescription then the category of the ConceptDescription shall be VIEW_.
(x)	AASd-065	New	Constraint AASd-065: If the semanticId of a Property or MultiLanguageProperty references a ConceptDescription with the category VALUE then the value of the property is identical to DataSpecificationIEC61360/value and the valueId of the property is identical to DataSpecificationIEC61360/valueId.
(x)	AASd-066	New	Constraint AASd-066: If the semanticId of a Property or MultiLanguageProperty references a ConceptDescription with the category PROPERTY and DataSpecificationIEC61360/valueList is defined the value and valueId of the property is identical to one of the value reference pair types references in the value list, i.e. ValueReferencePair/value or ValueReferencePair/valueId, resp.
(x)	AASd-067	New	Constraint AASd-067: If the semanticId of a MultiLanguageProperty references a ConceptDescription then DataSpecificationIEC61360/dataType shall be STRING_TRANSLATABLE.
(x)	AASd-068	New	Constraint AASd-068: If the semanticId of a Range submodel element references a ConceptDescription then DataSpecificationIEC61360/dataType shall be a numerical one, i.e. REAL_* or RATIONAL_*.
(x)	AASd-069	New	Constraint AASd-069: If the semanticId of a Range references a ConceptDescription then DataSpecificationIEC61360/levelType shall be identical to the set {Min, Max}.
(x)	AASd-070	New	Constraint AASd-070: For a ConceptDescription with category PROPERTY or VALUE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/dataType is mandatory and shall be defined.
(x)	AASd-071	New	Constraint AASd-071: For a ConceptDescription with category REFERENCE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/dataType is STRING by default.
(x)	AASd-072	New	Constraint AASd-072: For a ConceptDescription with category DOCUMENT using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/dataType shall be one of the following values: STRING or URL.
(x)	AASd-073	New	Constraint AASd-073: For a ConceptDescription with category QUALIFIER using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/dataType shall be one of the following values: QUALIFIER.

nc	V3.0RC01	New, Update, Removed, Reformulated	Comment
			shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/dataType is mandatory and shall be defined.
(x)	AASd-074	New	Constraint AASd-074: For all ConceptDescriptions except for ConceptDescriptions of category VALUE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) - DataSpecificationIEC61360/definition is mandatory and shall be defined at least in English.
(x)	AASd-075	New	Constraint AASd-075: For all ConceptDescriptions using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/2/0) values for the attributes not being marked as mandatory or optional in tables Table 6, Table 7, Table 8 and Table 9. depending on its category are ignored and handled as undefined.
	AASd-077	New	Constraint AASd-077: The name of an extension within HasExtensions needs to be unique.
(x)	AASd-080	New	Constraint AASd-080: In case <i>Key/type == GlobalReference idType</i> shall not be any <i>LocalKeyType (IdShort, FragmentId)</i> .
	AASd-081	New	Constraint AASd-081: In case <i>Key/type==AssetAdministrationShell Key/idType</i> shall not be any <i>LocalKeyType (IdShort, FragmentId)</i> .
(x)	AASd-092	New	Constraint AASd-092: If the semanticId of a SubmodelElementCollection with SubmodelElementCollection/allowDuplicates == false references a ConceptDescription then the ConceptDescription/category shall be ENTITY.
(x)	AASd-093	New	Constraint AASd-093: If the semanticId of a SubmodelElementCollection with SubmodelElementCollection/allowDuplicates == true references a ConceptDescription then the ConceptDescription/category shall be COLLECTION.
	AASd-100	New	Constraint AASd-100: An attribute with data type "string" is not allowed to be empty.

B. METAMODEL CHANGES V3.0RC01 – SECURITY PART

Table 35 Changes Metamodel w.r.t. Security

nc	V3.0RC01 w.r.t. V2.0.1 Change	Comment

Table 36 New Elements in Metamodel Security

nc	V3.0RC01 vs. V2.0.1 New Elements w.r.t V2.0.1	Comment

Table 37 New, Changed or Removed Constraints Security

nc	V3.0RC01	New, Removed, Reformulated	Update, Comment
	AASd-010	Removed	Renamed to AASs-010 (see NEW)
	AASs-010	NEW	Reformulation of AASd-010 Constraint AASs-010: The property referenced in Permission/permission shall have the category "CONSTANT".
	AASd-011	Removed	Renamed to AASs-011 (see NEW)
	AASs-011	NEW	Reformulation of AASd-011 Constraint AASs-011: The property referenced in Permission/permission shall be part of the submodel that is referenced within the "selectablePermissions" attribute of "AccessControl".
	AASd-015	Removed	Renamed to AASs-015 (see NEW)
	AASs-015	NEW	Constraint AASd-015: The data element SubjectAttributes/subjectAttribute shall be part of the submodel that is referenced within the "selectableSubjectAttributes" attribute of "AccessControl".

v. CHANGES V2.0.1 VS. V2.0

A. METAMODEL CHANGES V2.0.1 W/O SECURITY PART

Major changes:

- Only bugfixes

Table 38 Changes w.r.t. V2.0.1 w/o Security

nc	V2.0.1 Change w.r.t. V2.0	Comment
	DataTypeIEC61360/INTEGER_COUNT	Bugfix, was missing
	DataTypeIEC61360/INTEGER_MEASURE	Bugfix, was missing
	DataTypeIEC61360/INTEGER_CURRENCY	Bugfix, was missing
	hasDataSpecification	Bugfix, ist abstract class

Table 39 New Elements in Metamodel V2.0.1 w/o Security

V2.0.1 w.r.t. V2.0 New Elements	Comment

Table 40 New, Changed or Removed Constraints w/o Security

nc	V2.0.1	New, Update, Removed	Comment
	AASd-013	Removed	Constraint AASd-013: Min and Max of a Property Range can be empty, denoting a range with open upper and lower boundary

B. METAMODEL CHANGES V2.0.1 – SECURITY PART

Table 41 Changes Metamodel w.r.t. V2.0 Security

nc	V2.1 Change w.r.t. V2.0	Comment

Table 42 New Elements in Metamodel V2.1 w.r.t. V2.0 Security

V2.1	Comment

Table 43 New, Changed or Removed Constraints w/o Security

nc	V2.0.1 w.r.t. V2.0	New, Update, Removed	Comment
	AASd-001	update	idShort now mandatory Constraint AASd-001: an identifiable element this id is mandatory and used for referring to the element in its name space. → Constraint AASd-001: In case of a referable element not being an identifiable element this ID is used for referring to the element in its name space.
	AASd-013	removed	Constraint AASd-013: In case of a range with kind=Instance either the min or the max value or both need to be defined.

vi. CHANGES V2.0 VS. V1.0

A. METAMODEL CHANGES V2.0 W/O SECURITY PART

Major changes:

- Composite I4.0 Components supported via new Entity submodel element and billOfMaterial
- Event submodel element introduced
- Capability submodel element introduced
- Annotatable relationship submodel element introduced
- MultiLanguageProperty submodel element introduced
- Range submodel element introduced
- Data Specification Template IEC61360 extended for Values, ValueLists and Ranges
- Also referencing of fragments within a file etc. now supported

Table 44 Changes w.r.t. V1.0 w/o Security

nc	V2.0 Change w.r.t. V1.0	Comment
(x) ⁵⁵	anySimpleTypeDef	Type starts now with capital letter: AnySimpleTypeDef Type changed from string to values representing xsd-type anySimpleType
	Asset	Does not inherit from HasKind any longer (but attribute kind remains)
	Asset/kind	Now of type "AssetKind" instead of "Kind". Instead of value Type and Instance now value Template and Instance
	AssetAdministrationShell/security	Now optional to support passive AAS of type 1
	Code	Data type removed, not used any longer
x	DataSpecificationIEC61360/shortName	Type changed from string to LangStringSet Cardinality changed from mandatory to optional
x	DataSpecificationIEC61360/sourceOfDefinition	Type changed from langString to string
(x) ⁵⁶	DataSpecificationIEC61360/dataType	Type changed from string to Enumeration Cardinality changed from mandatory to optional
x	DataSpecificationIEC61360/code	Attribute code removed
	DataSpecificationIEC61360/definition	Cardinality changed from mandatory to optional
	HasDataSpecification	Was abstract before
	HasDataSpecification/hasDataSpecification	Renamed to HasDataSpecification/dataSpecification
x	HasKind/kind	Now of type "ModelingKind" instead of "Kind". Values changed: Type now Template; Instance remains
x	File/value	File name not without but with extension
x	Identifiable/description	Type changed from langString to LangStringSet
x	IdentifierType/URI	URI renamed to IRI
	Kind	Type Kind removed and substituted by types AssetKind and ModelingKind
x	OperationVariable	Does not inherit from SubmodelElement any longer

⁵⁵ Implicitly there was a constraint restricting the values to the values in the enumeration. This is now formalized.

⁵⁶ Implicitly there was the constraint that only IEC61360 data types are allowed to be used. This is now formalized.

nc	V2.0 Change w.r.t. V1.0	Comment
	Property/value	Type changed from anySimpleTypeDef to ValueDataType
x	Qualifier/qualifierType	Renamed to Qualifier/type
x	Qualifier/qualifierValue	Renamed to Qualifier/value Type changed from AnySimpleTypeDef to ValueDataType
x	Qualifier/qualifierValueId	Renamed to Qualifier/valueId
x	Referable/idShort	Now mandatory, was optional (but with constraints for defined elements)
x	Reference/key	Cardinality changed from 0..* to 1..*

Table 45 New Elements in Metamodel V1.0 w/o Security

V2.0	Comment
AnnotatedRelationshipElement	New submodel element, inheriting from RelationshipElement
Asset/billOfMaterial	New attribute
AssetKind	New enumeration type
BasicEvent	New submodel element, inherits from Event
Capability	New submodel element
DataSpecificationIEC61360/valueList	For value lists (string)
DataSpecificationIEC61360/value	For coded and explicit values
DataSpecificationIEC61360/valueId	For coded values
DataSpecificationIEC61360/levelType	For Ranges
DataSpecificationPhysicalUnit	New data specification template
DataTypesIEC61360	New enumeration type
Entity	New submodel Element
EntityType	New enumeration type
IdentifierType	Is a subset of KeyType Enumeration
KeyElements/FragmentReference	New value FragmentReference as part of KeyElements Enumeration
LocalKeyType	Is a subset of KeyType Enumeration
LocalKeyType/FragmentId	New value for KeyType Enumeration (via subset LocalKeyType)
LangStringSet	New type, used for example in MultiLanguageProperty
LevelType	New enumeration type
ModelingKind	New enumeration type
MultiLanguageProperty	New submodel element
Qualifier/valueType	New attribute to be consistent with valueType of Property etc.

V2.0	Comment
Range	New submodel element
ReferableElements/BasicEvent	New enumeration value
ReferableElements/Capability	New enumeration value
ReferableElements/Event	New enumeration value
ReferableElements/MultiLanguageProperty	New enumeration value
ReferableElements/Range	New enumeration value
ValueDataType	New type, used for example for Property value
ValueList	New class
ValueReferencePairType	New class

Table 46 New, Changed or Removed Constraints w/o Security

nc	V2.0	New, Update, Removed	Comment
	AASd-007	update	Reformulated Constraint AASd-007: 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.
	AASd-008	update	Reformulated Constraint AASd-008: The submodel element value of an operation variable shall be of kind=Template.
	AASd-025	removed	Redundant to AASd-015 Constraint AASd-025: The data element shall be part of the submodel that is referenced within the “selectableSubjectAttributes” attribute of “AccessControl”.

B. METAMODEL CHANGES V2.0 – SECURITY PART

Table 47 Changes Metamodel w.r.t. V1.0 Security

nc	V2.0 Change w.r.t. V1.0	Comment
x	AccessControl/selectableEnvironmentAttributes	Type changed from Submodel to Submodel*
	AccessPermissionRule/permissionsPerObject	Cardinality now consistent for figure and table: 0..*
x	AccessPermissionRule/targetSubjectAttributes	Cardinality changed from 1..* to 1
	Certificate	Was abstract, now not abstract and contains attributes (see in table New)
x	PermissionKind/allow	Now PermissionKind/Allow start with capital letter for enumeration values
x	PermissionKind/deny	Now PermissionKind/Deny start with capital letter for enumeration values
x	PermissionKind/not applicable	Now PermissionKind/NotApplicable start with capital letter for enumeration values
x	PermissionKind/Undefined	Now PermissionKind/Undefined start with capital letter for enumeration values
	PermissionsPerObject	Name now consistent for figure and table (in table PermissionPerObject, needs to be PermissionsPerObject)
x	PolicyAdministrationPoint/externalAccessControl	Type changed from Endpoint to Boolean, cardinality 1
x	PolicyInformationPoints/externalInformationPoint	Type changed from Endpoint to Boolean, cardinality 1 externalInformationPoint renamed to externalInformationPoints
x	Security/trustAnchor	Renamed to Security/certificate

Table 48 New Elements in Metamodel w.r.t. Security

V2.0	Comment
BlobCertificate	New class inheriting from Certificate
Certificate	Abstract class: was foreseen in V1.0 but not yet modelled
Security/requiredCertificateExtension	New attribute
PolicyEnforcementPoint	Was foreseen in V1.0 but not yet modelled
PolicyEnforcementPoint/externalPolicyEnforcementPoint	
PolicyDecisionPoint	Was foreseen in V1.0 but not yet modelled
PolicyDecisionPoint/externalPolicyDecisionPoint	

Table 49 New, Changed or Removed Constraints w/o Security

nc	V2.0	New, Update, Removed	Comment

ANNEX G. BIBLIOGRAPHY

- [1] “Recommendations for implementing the strategic initiative INDUSTRIE 4.0”, acatech, April 2013. [Online]. Available: <https://www.acatech.de/Publikation/recommendations-for-implementing-the-strategic-initiative-industrie-4-0-final-report-of-the-industrie-4-0-working-group/>
- [2] “Implementation Strategy Industrie 4.0: Report on the results of the Industrie 4.0 Platform”; BITKOM e.V. / VDMA e.V., /ZVEI e.V., April 2015. [Online]. Available: <https://www.bitkom.org/noindex/Publikationen/2016/Sonstiges/Implementation-Strategy-Industrie-40/2016-01-Implementation-Strategy-Industrie40.pdf>
- [3] DIN SPEC 91345:2016-04 “Referenzarchitekturmodell Industrie 4.0 (RAMI4.0) / Reference Architecture Model Industrie 4.0 (RAMI4.0) / Modèle de référence de l’architecture de l’industrie 4.0 (RAMI4.0)”, ICS 03.100.01; 25.040.01; 35.240.50, April 2016. [Online]. Available: <https://www.beuth.de/en/technical-rule/din-spec-91345-en/250940128>
- [4] “Structure of the Administration Shell, continuation of the development of the reference model for the Industrie 4.0 component”, Plattform Industrie 4.0, Working Paper, April 2016. [Online]. Available: <https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/structure-of-the-administration-shell.html>
- [5] “Which criteria do Industrie 4.0 products need to fulfil? Guideline 2020”, Federal Ministry for Economic Affairs and Energy (BMWi), July 2020. [Online]. Available: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/criteria-industrie-40-products_2020.html
- [6] “Beispiele zur Verwaltungsschale der Industrie 4.0-Komponente – Basisteil (German)”; ZVEI e.V., Whitepaper, November 2016. [Online]. Available: <https://www.zvei.org/presse-medien/publikationen/beispiele-zur-verwaltungsschale-der-industrie-40-komponente-basisteil/>
- [7] “Aspects of the research roadmap in application scenarios”, Plattform Industrie 4.0, working paper, April 2016. [Online]. Available: <http://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/aspects-of-the-research-roadmap.html>
- [8] “Fortschreibung der Anwendungsszenarien der Plattform Industrie 4.0 (German)”; Plattform Industrie 4.0, Ergebnispapier, October 2016. [Online]. Available: <https://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/fortschreibung-anwendungsszenarien.html>
- [9] “Security in RAMI4.0”, Plattform Industrie 4.0, Berlin, technical overview, April 2016. [Online]. Available: <http://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/security-rami40-en.html>
- [10] “Die Deutsche Normungs-Roadmap Industrie 4.0 / The German standardization roadmap Industrie 4.0”, DKE Deutsche Kommission Elektrotechnik, Elektronik Informationstechnik im DIN und VDE, Version 2.0, 2015. [Online]. Available: <http://www.din.de/de/forschung-und-innovation/industrie4-0/roadmap-industrie40-62178>
- [10a] “Weiterentwicklung des Interaktionsmodells für Industrie 4.0-Komponenten“, Plattform Industrie 4.0, discussion paper, November 2016. [Online]. Available: <https://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/interaktionsmodell-i40-komponenten-it-gipfel.html>
- [11] “Definition of terms relating to Industrie 4.0”, Fraunhofer IOSB and VDI/VDE-GMA Fachausschuss 7.21. Accessed: 2020-11-14. [Online]. Available: <http://i40.iosb.fraunhofer.de/search?patterns=FA7.21%20Begriffe>
- [12] “Relationships between I4.0 Components – Composite Components and Smart Production”, Plattform Industrie 4.0, Berlin, working paper, June 2017. [Online]. Available: <https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/hm-2018-relationship.html>
- [13] “Industrie 4.0 Plug-and-Produce for Adaptable Factories”; Plattform Industrie 4.0, Berlin, working paper, June 2017. [Online]. Available: <http://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/Industrie-40-%20Plug-and-Produce.html>

- [14] “Security der Verwaltungsschale / Security of the Administration Shell”, Plattform Industrie 4.0, Berlin, working paper, April 2017. [Online]. Available: <http://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/security-der-verwaltungsschale.html>
- [15] DIN SPEC 92000:2019-09 “Data Exchange on the Base of Property Value Statements (PVSX)”, 2019 September.
- [16] “Verwaltungsschale in der Praxis. Wie definiere ich Teilmodelle, beispielhafte Teilmodelle und Interaktion zwischen Verwaltungsschalen (in German)”, Version 1.0, April 2019, Plattform Industrie 4.0 in Kooperation mit VDI/VDE-GMA Fachausschuss 7.20, Federal Ministry for Economic Affairs and Energy (BMWi), Available: <https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/2019-verwaltungsschale-in-der-praxis.html>
- [17] “I4.0-Sprache. Vokabular, Nachrichtenstruktur und semantische Interaktionsprotokolle der I4.0-Sprache (German)”, Plattform Industrie 4.0 in Kooperation mit VDI/VDE-GMA Fachausschuss 7.20, April 2018. [Online]. Available: <https://www.plattform-i40.de/I40/Redaktion/DE/Downloads/Publikation/hm-2018-sprache.html>
- [18] “The Structure of the Administration Shell: TRILATERAL PERSPECTIVES from France, Italy and Germany”, March 2018, [Online]. Available: <https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/hm-2018-trilaterale-coop.html>
- [19] „Access control for Industrie 4.0 components for application by manufacturers, operators and integrators“, Plattform Industrie 4.0, Discussion Paper, Sept. 2019 (German Version Nov. 2018), [Online]. Available: <https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Access%20control%20for%20Industrie%204.0%20components.html>
- [20] “Industrial automation systems and integration — Exchange of characteristic data — Part 10: Characteristic data exchange format”, Technical Specification ISO/TS 29002-10:2009(E), 2009
- [21] “Reference Architecture Model for the Industrial Data Space”, Fraunhofer in cooperation with Industrial Data Space Association, 2017. [Online]. Available: https://www.fit.fraunhofer.de/content/dam/fit/en/documents/Industrial-Data-Space_Reference-Architecture-Model-2017.pdf
- [22] Vincent Hu, David Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller and Karen Scarfone, “Guide to Attribute Based Access Control (ABAC) Definition and Considerations”, NIST Special Publication 800-162, Jan. 2014. [Online]. Available: <http://dx.doi.org/10.6028/NIST.SP.800-162>
- [23] “Smart Manufacturing - Reference Architecture Model Industry 4.0 (RAMI4.0)”, IEC PAS 63088, International Electrotechnical Commission (IEC), 2017
- [24] “Sustainability of Digital Formats: Planning for Library of Congress Collections. Open Packaging Conventions (Office Open XML)”, ISO 29500-2:2008-2012, 2012. [Online]. Available: <https://www.loc.gov/preservation/digital/formats/fdd/fdd000363.shtml>
- [25] “Standardization of Office Open XML”, Wikipedia. Accessed: 2019-01-26 [Online]. Available: https://en.wikipedia.org/wiki/Standardization_of_Office_Open_XML
- [26] “OpenDocument standardization”, Wikipedia. Accessed: 2019-01-26 [Online]. Available: https://en.wikipedia.org/wiki/OpenDocument_standardization
- [27] “The Digital Signing Framework of the Open Packaging Conventions”. Accessed: 2019-01-26. [Online]. Available: <https://msdn.microsoft.com/en-us/library/aa905326.aspx>
- [28] “Open Packaging Conventions Fundamentals”. Accessed: 2019-01-26 [Online]. Available: [https://msdn.microsoft.com/en-us/library/windows/desktop/dd742818\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd742818(v=vs.85).aspx)
- [29] “What is a digital signature? Fundamental principles”. Accessed: 2019-01-26. [Online]. Available: <http://securityaffairs.co/wordpress/5223/digital-id/what-is-a-digital-signature-fundamental-principles.html>

- [30] "Sustainability of Digital Formats: Planning for Library of Congress Collections. Document Container File: Core (based on ZIP 6.3.3)". Accessed: 2019-01-26. [Online]. Available: <https://www.loc.gov/preservation/digital/formats/fdd/fdd000361.shtml>
- [31] "System.IO.Packaging Namespace", MSDN, Accessed: 2019-01-26 [Online]. Available: [https://msdn.microsoft.com/en-us/library/system.io.packaging\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.io.packaging(v=vs.110).aspx)
- [32] DIN SPEC 16593-1 "Reference Model for Industrie 4.0 Service Architectures – Part 1: Basic Concepts of an Interaction-based Architecture", Beuth-Verlag: Berlin, Germany, 2018. [Online]. Available: <https://www.beuth.de/en/technical-rule/din-spec-16593-1/287632675>
- [33] ISO 13854-42 "Standard data element types with associated classification scheme – Part 1: Definitions – Principles and methods" Edition 4.0, 2017-07
- [34] IEC 61360-1 "Standard data element types with associated classification scheme – Part 1: Definitions – Principles and methods", Edition 4.0, 2017-07
- [35] ISO/TS 29002-10:2009(E) "Industrial automation systems and integration — Exchange of characteristic data — Part 10: Characteristic data exchange format", First edition 2009-12-01
- [36] A. Bayha, J. Bock, B. Boss, C. Diedrich, S. Malakuti "Describing Capabilities of Industrie 4.0 Components". Nov. 2020. Plattform Industrie 4.0. [Online] Available: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Capabilities_Industrie40_Components.html
- [37] AutomationML Association: "Application Recommendations: Asset Administration Shell Representation (AR 004E)", Version 1.0.0, 20.11.2019, [Online]. Available: <https://www.automationml.org/o.red.c/dateien.html>
- [38] H. Knublauch, D. Knotokostas "Shapes Constraint Language (SHACL)" W3C Recommendation, 2017, [Online]. Available: <https://www.w3.org/TR/shacl/>
- [39] "I4AAS – Industrie 4.09 Asset Administration Shell". June 2021. [Online] Available: <https://opcfoundation.org/markets-collaboration/I4AAS/>
- [40] "AASX Package Explorer. Software" Download: <https://github.com/admin-shell-io/aasx-package-explorer>
- [41] "AAS Repository. Repository for Information and Code for the Asset Administration Shell". <https://github.com/admin-shell-io>
- [42] "Eclipse BaSyx". [Online]. Available: <https://www.eclipse.org/basyx/>
- [43] DIN SPEC 91406:2019 "Automatische Identifikation von physischen Objekten und Informationen zum physischen Objekt in IT-Systemen, insbesondere IoT-Systemen/Automatic identification of physical objects and information on physical objects in IT systems, particularly IoT systems". December 2019
- [44] F. Manola, E. Miller "RDF 1.1 Primer" W3C Recommendation, 2014, [Online]. Available: <https://www.w3.org/TR/rdf11-primer/>
- [45] T. R. Gruber "A translation approach to portable ontology specifications." Knowledge acquisition 5.2 (1993): 199-220. [Online]. Available: <https://tomgruber.org/writing/ontolingua-kaj-1993.htm>
- [46] "The Industrial Internet of Things Vocabulary". Technical Report. Version 2.3. October 10, 2020. Industrial Internet Consortium. IIC:IIVOC:V2.3:20201025 [Online] Available: <https://www.iiconsortium.org/vocab/>
- [47] "OMG Unified Modeling Language (OMG UML)". Formal/2017-12-05. Version 2.5.1. December 2018. [Online] Available: <https://www.omg.org/spec/UML/>
- [48] T. Preston-Werner "Semantic Versioning". Version 2.0.0. Accessed: 2020-11-13. [Online] Available: <https://semver.org/spec/v2.0.0.html>
- [49] "Details of the Asset Administration Shell – Interoperability at Runtime – Exchanging Information via Application Programming Interfaces". Version 1.0RC01. November 2020. Plattform Industrie 4.0 [Online] Available: <https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Details-of-the-Asset-Administration-Shell-Part2/1/0.html>

- [50] “Asset Administration Shell. Reading Guide”. Plattform Industrie 4.0 in cooperation with IDTA. November 2020. [Online] Available: https://industrialdigitaltwin.org/wp-content/uploads/2022/02/AAS-ReadingGuide_202201.pdf
- [51] “Submodel Templates of the Asset Administration Shell - Generic Frame for Technical Data for Industrial Equipment in Manufacturing”, Version 1.1, Nov. 2020, Plattform Industrie 4.0 [Online] Available: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Submodel_templates-Asset_Administration_Shell-Technical_Data.html
- [52] “Submodel Templates of the Asset Administration Shell - ZVEI Digital Nameplate for industrial equipment”, Version 1.0, Nov. 2020, Plattform Industrie 4.0 [Online] Available: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Submodel_templates-Asset_Administration_Shell-digital_nameplate.html
- [53] “Secure Download Service”, Discussion Paper. Oct. 2020, Plattform Industrie 4.0 [Online] Available: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/secure_downloadservice.html
- [54] “eXtensible Access Control Markup Language (XACML)”, OASIS Standard, Version 3.0, 22. Jan. 2013, [Online]. Available: <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf>
- [55] Top Level Project “Eclipse Digital Twin” Available: <https://projects.eclipse.org/projects/dt>
- [56] OPC 30270: OPC UA for Asset Administration Shell (AAS). 2021-06-04. [Online]. Available: <https://reference.opcfoundation.org/v104/I4AAS/v100/docs/>
- [57] OPC Unified Architecture Specification. Part 5 Information Model. [Online]. Available: <https://opcfoundation.org/developer-tools/specifications-unified-architecture>
- [58] OPC UA Information Models. [Online]. Available: <https://opcfoundation.org/developer-tools/specifications-opc-ua-information-models>
- [59] IEC 63278-1 “Asset Administration Shell for industrial applications – Part 1: Asset Administration Shell structure”. 95/925/CDV
- [60] “Registered AAS Submodel Templates”. Industrial Digital Twin Association. Available: <https://industrialdigitaltwin.org/en/content-hub/submodels>

AUTHORS of V1.0, V2.x and/or V3.x

Note: Not all authors contributed to all versions or to this version. The company name reflects the company at the point of time of the publication and might differ from the company name in previous versions.

Sebastian Bader, SAP SE

Erich Barnstedt, Microsoft Deutschland GmbH

Dr. Heinz Bedenbender, VDI/VDE-Gesellschaft für Mess- und Automatisierungstechnik (GMA)

Bernd Berres, MPDV Mikrolab GmbH

Meik Billmann, IDTA e.V.

Dr. Birgit Boss, Robert Bosch GmbH

Nico Braunsch, TU Dresden

Dr. André Braunmandl, BSI

Erich Clauer, SAP SE

Professor Dr. Christian Diedrich, ifak - Institut f. Automation und Kommunikation e.V. Magdeburg

Björn Flubacher, BSI

Wolfgang Fritsche, IABG mbH

Kai Garrels, ABB STOTZ-KONTAKT GmbH

Dr. Andreas Graf Gatterburg, Hilscher Gesellschaft für Systemautomation mbH

Martin Hankel, Bosch Rexroth AG

Sebastian Heppner, RWTH Aachen

Oliver Hillermeier, SAP SE

Dr. Michael Hoffmeister, Festo AG & Co. KG

Dr. Lutz Jänicke, PHOENIX CONTACT GmbH & Co. KG

Michael Jochem, Robert Bosch GmbH

Tobias Klausmann, Lenze SE SE

Alexander Köpke, Microsoft Deutschland GmbH

Yevgen Kogan, KUKA Deutschland GmbH

Dr. Heiko Koziol, ABB AG

Florian Krebs, Deutsches Zentrum für Luft- und Raumfahrt e.V.

Walter Kuhlbusch, Festo SE & Co. KG

Dr. Christoph Legat, Hekuma GmbH

Professor Dr. Arndt Lüder, Otto-von-Guericke Universität

Dr. Wolfgang Mahnke, ascolab GmbH

Dr. Somayeh Malakuti, ABB AG

→ Continued next page

AUTHORS of Version 1.0, 2.x and/or 3.x - continued

Dr. Marco Mendes, Schneider Electric Automation GmbH

Dr. Torben Miny, RWTH Aachen

Dr. Jörg Neidig, SIEMENS AG

Andreas Neubacher, Deutsche Telekom

Andreas Orzelski, PHOENIX CONTACT GmbH & Co. KG

Florian Pethig, Fraunhofer IOSB-INA

Stefan Pollmeier, ESR Pollmeier GmbH Servo-Antriebstechnik

Magnus Redeker, Fraunhofer IOSB-INA

Marko Ristin, ZHAW Zürich

Manuel Sauer, SAP SE

Volker Schaber, SICK AG

Daniel Schel, Fraunhofer IPA

Otto Schell, Deutschsprachige SAP Anwender Gruppe e.V. (DSAG)

Marc Schier, Microsoft Deutschland GmbH

Dr. Miriam Schleipen, EKS InTec GmbH

Dr. Michael Schmitt, SAP SE

Tizian Schröder, Otto-von-Guericke-Universität Magdeburg

Guido Stephan, SIEMENS AG

Dr. Ljilijana Stojanovic, Fraunhofer IOSB

Andreas Teuscher, SICK AG

André Uhl, Schneider Electric Automation GmbH

Dr. Thomas Usländer, Fraunhofer IOSB

Jens Vialkowitsch, Robert Bosch GmbH

Friedrich Vollmar

Thomas Walloschke, Industrie KI GmbH

Bernd Waser, Murrelektronik GmbH

Jörg Wende, IBM Deutschland GmbH

Mathias Wiegand, Festo AG & Co. KG

Nico Wilhelm, ZF Friedrichshafen AG

Constantin Ziesche, Bosch Rexroth AG

This working paper has been elaborated in the Joint Working Subgroup “Asset Administration Shell” of the Working Group on “Reference Architectures, Standards and Norms “ of Plattform Industrie 4.0, the Working Group “Open Technology” of the Industrial Digital Twin Association (IDTA) and the working group “Models and Standards” of the ZVEI in cooperation with the Working Groups “Security of networked Systems” (Plattform Industrie 4.0) and “Security” (ZVEI). The OPC UA Mapping has been elaborated in the joint working group “I4AAS” of the OPC Foundation, ZVEI, VDMA and the Plattform Industrie 4.0. The AutomationML Mapping has been elaborated in the joint working group of AutomationML e.V. and the Plattform Industrie 4.0.

