April 2023

Specification of the Asset Administration Shell

Part 3a: Data Specification – IEC 61360

SPECIFICATION

IDTA Number: 01003-a-3-0

Imprint

Publisher
Industrial Digital Twin Association
Lyoner Strasse 18
60528 Frankfurt am Main
Germany
https://www.industrialdigitaltwin.org/

Status
March 2023

Illustrations
Publik. Agentur für Kommunikation GmbH, designed by Publik. Agentur für Kommunikation GmbH

Contents

Table of Tables

Table of Figures

1 Preamble

1.1 Editorial Notes

The document "Details of the Asset Administration Shell – Part 1 – The exchange of information between partners in the value chain of Industie 4.0, V3.0RC02" was split into several parts. One of them is this document, which represents Part 3a and describes a data specification that is defined to be used with the core model as specified in Part 1. This is also why versioning now starts with V3.0: it is only valid in combination with V3.0 of Part 1.

This document, version 3.0, was produced from June 2022 to November 2022 by the sub working group "Asset Administration Shell" of the joint working group of the Plattform Industrie 4.0 working group "Reference Architectures, Standards and Norms" and the "Open Technology" working group of the Industrial Digital Twin Association (IDTA). It is the first release published by the IDTA.

For a complete history, please refer to Part 1a of the document series " Details of the Asset Administration Shell".

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

This specification is versioned using Semantic Versioning 2.0.0 (semver) and follows the semver specification [13].

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]:

1. MUST   This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
2. MUST NOT   This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
3. SHOULD   This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
4. SHOULD NOT   This phrase, or the phrase "NOT RECOMMENDED", mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
5. MAY   This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein, an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

[11] https://www.ietf.org/rfc/rfc2119.txt

1.2 Scope of This Document

The Asset Administration Shell (see Part 1a of the document series) allows to define data specification templates. Data specification templates aim to enable interoperability between the partners that agree on the template. A template defines a set of attributes, with each attribute having a clear semantics. This set of attributes corresponds to a (sub-)schema.

This document specifies data specification templates conformant to IEC 61360. IEC 61360 specifies how to define the semantics of single properties or values. The value range of a property can be defined as a value list – an enumeration -, while each of the (coded) values of the value list are treated as single concepts. They are thus suited to be used as data specifications for concept descriptions.

This document assumes familiarity with the concept and specification of the Asset Administration Shell as defined in Part 1.

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 organizations and further collaborations.

Please consult the continuously updated reading guide [15] for an overview of documents on the Asset Administration Shell. The reading guide gives advice on which documents should be read depending on the role of the reader.

1.3 Normative References

[AAS-Part1] "The Exchange of information between partners in the value chain of Industrie 4.0", part of document series "Details of the Asset Administration Shell". V3.0. Jan. 2023. Industrial Digital Twin Association.

[IEC61360-1] Standard data element types with associated classification scheme –
Part 1: Definitions – Principles and methods. Edition 4.0 2017-07

[[IEC61360-2]] Standard data element types with associated classification scheme for electronic components. Part 2: EXPRESS dictionary schema. Edition 2012.

[ISO 13584-42] ISO 13584-42:2010, *Industrial automation systems and integration – Parts library – Part 42: Description methodology: Methodology for structuring part families*

1.4 Structure of the Document

All clauses that are normative have "(normative)" as a suffix in the heading of the clause.

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 of Asset Administration Shell types and how this document is related to them.

Clause 4 explains the purpose of the data specification template specified in this document by giving examples of existing data dictionaries.

Clause 5 shows how the data specification template is related to Part 1 and its elements.

Clause 6 is the main normative part of the document. It specifies the data specification templates supporting IEC 61360.

Clause 7 explains how data types of IEC 61360 are mapped to data types of values as introduced in Part 1.

Clause 8 introduces categories for concept descriptions and how they are used in combination with the data specification template IEC61360. The constraints as defined in Clause 0 also mainly refer to the rules on how these categories should be applied.

Note: since categories are deprecated since V3.0, Clause 8 can also be skipped.

Clause 10 specifies the data types used in the data specification.

Clause 0 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.

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

The Annex contains additional background information on the Asset Administration Shell (Annex A). It also provides information about UML (Annex D) and the tables used to specify UML classes as used in this specification (Annex C). Annex B introduces the Backus-Naur-Form used in the document series.

Metamodel changes compared to previous versions are described in Annex E.

The bibliography can be found in 0.

1.5 Working Principles

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

The partners represented in the Industrial Digital Twin Association (IDTA), as well as in the Plattform Industrie 4.0 and associations such as ZVEI, VDMA, VDI/ VDE and Bitkom, ensure that there is broad sectoral coverage of 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 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 and Definitions

Please note:

the definitions of terms are only valid in a certain context. This glossary applies only within the context of this document. For a more extensive list, please refer to Part 1 of the document series.

If available, definitions were taken from IEC 63278-1 DRAFT, July 2022, and from IEC 61360.

application

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

Note 1 to entry: an application can be distributed among resources and may communicate with other applications.

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

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]

Asset Administration Shell (AAS)

standardized digital representation of an asset

Note 1 to entry: Asset Administration Shell and Administration Shell are used synonymously.

→ [SOURCE: IEC 63278-1, note added]

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]

concept

unit of knowledge created by a unique combination of characteristics

→ [SOURCE: EC 63278-1; IEC 61360-1:2016, 3.1.8; ISO 22274:2013, 3.7]

enumeration

list of named constants called enumerators, each numerator name in the enumeration being unambiguous

→ [SOURCE:IEC 61360-1_2017]

identifier (ID)

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

Note 1 to entry: there are specific identifiers, e.g. UUID Universal unique identifier, IEC 15418 (GS1).

→ [SOURCE: Glossary Industrie 4.0]

minimum value

lower bound of a range of values in which the said value is meaningful

EXAMPLE 1: lowest value specified of a quantity, established for a specified set of operating conditions at which a component, device, equipment, or system can operate and perform according to specified requirements.
Note 1 to entry: additional information about the nature of the value can be obtained from the definition of the *Property* information object to which the value belongs.

→ [SOURCE:IEC 61360-1_2017]

maximum value

upper bound of a range of values in which the said value is meaningful

EXAMPLE 1: highest value specified of a quantity, established for a specified set of operating conditions at which a component, device, equipment, or system can operate and perform according to specified requirements.
Note 1 to entry: additional information about the nature of the value can be obtained from the definition of the *Property* information object to which the value belongs.

→ [SOURCE:IEC 61360-1_2017]

nominal value

value of a quantity used to designate or identify an item with its value, and not necessarily corresponding to the real value of the property

Note 1 to entry: additional information about the nature of the value can be obtained from the definition of the *Property* information object to which the value belongs.

→ [SOURCE:IEC 61360-1_2017]

non-quantitative property

property that identifies or describes an object by means of codes, abbreviations, names, references or descriptions

EXAMPLE 1: typical information content of non-quantitative properties is items such as codes, abbreviations, names, references, or descriptions.

→ [SOURCE: IEC 61360-2:201 7– based on IEC 61360-2:2012, 3.28, modified – "data element type" is replaced by "property" in the term and definition.]

property

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

Note 1 to entry: the concept of type and instance applies to properties.
Note 2 to entry: this definition applies to properties as described in IEC 61360/ ISO 13584-42.
Note 3 to entry: 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 to entry: the property instances have a value and are provided by the manufacturers. A property instance is also called property-value pair in certain standards.
Note 5 to entry: properties include nominal value, actual value, runtime variables, measurement values, etc.
Note 6 to entry: a property describes one characteristic of a given object.
Note 7 to entry: a property can have attributes such as code, version, and revision.
Note 8 to entry: the specification of a property can include predefined choices of values.

→ [SOURCE: according to 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 1 to entry: qualifier can have associated values.

- → [SOURCE: according to IEC 62569-1]

quantitative property

property with a numerical value representing a physical quantity, a quantity of information or a count of objects

- → [SOURCE: IEC 61360-1_2017 – based on IEC 61360-2:2012, 3.40, modified – "data element type" is replaced by "property"]

Submodel

container of SubmodelElements defining a hierarchical structure consisting of SubmodelElements

- → [SOURCE: IEC 63278-1]

SubmodelElement

elements in a Submodel

- → [SOURCE: IEC 63278-1]

2.2 Abbreviations Used in Document

Abbreviation	Description
AAS	Asset Administration Shell
AASX	Package file format for the Asset Administration Shell
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)
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 Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTC	Universal Time Coordinated
VDE	Verband der Elektrotechnik, Elektronik und Informationstechnik e.V.
VDI	Verein Deutscher Ingenieure e.V.
VDMA	Verband Deutscher Maschinen- und Anlagenbau e.V.
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
ZIP	archive file format that supports lossless data compression

Abbreviation	Description
ZVEI	Zentralverband Elektrotechnik- und Elektronikindustrie e. V.

3 Introduction

This document is part of the series "Details of the Asset Administration Shell" that provide the specifications for interoperable usage of the Asset Administration Shell.

This part of the series extends Part 1 and defines a technology-neutral specification of data specification templates, enabling the description of concept descriptions conformant to IEC 61360 in UML. This UML meta model serves as the basis for deriving several different formats for exchanging Asset Administration Shells, e.g. for XML, JSON, RDF, AutomationML, and OPC UA information models.

Figure 1 shows the different ways of exchanging information via Asset Administration Shells. This part of the "Asset Administration Shell in Detail" series is the basis for all of these types of information exchange.

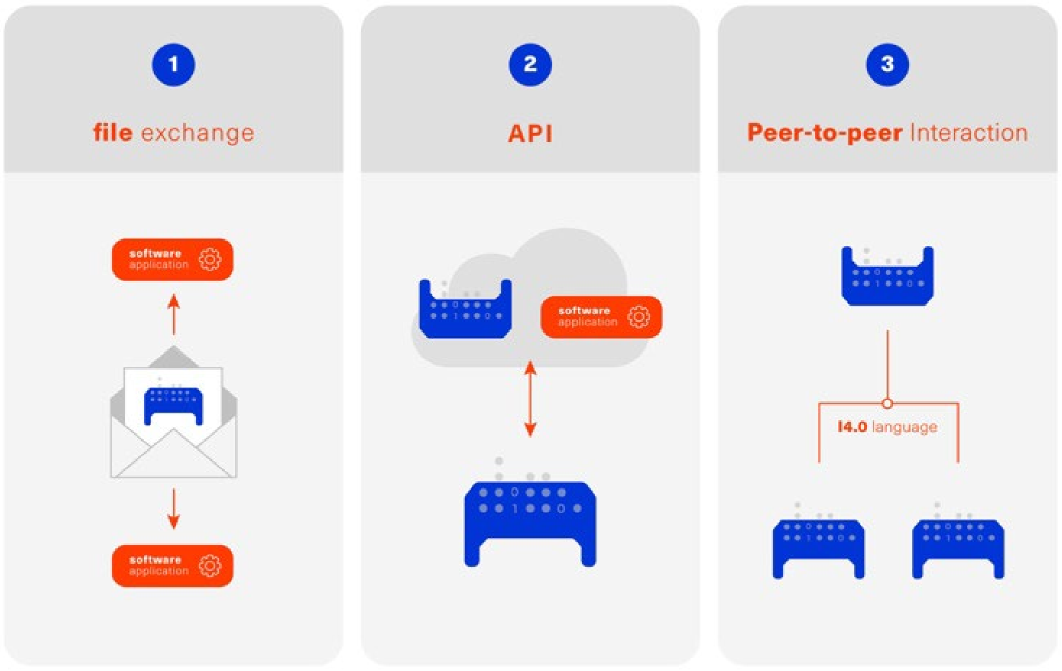


Figure 1 Types of Information Exchange via Asset Administration Shells

File exchange (1) is described in Part 5 of this document series.

The API (2) is specified in Part 2 of the document series "Details of the Asset Administration Shell" [14]. It also includes access to concept descriptions using the data specifications as specified in this document.

The I4.0 language (3) is based on the information metamodel specified in Part 1 and 3 [23].

Part 3 is not a single document. Instead, it is an own series of documents, each featuring a specific use case that is supported by the specified data specification templates.

4 General

4.1 Introduction

IEC 61360 is a standard that describes how to define the semantics of properties in a data dictionary. The known data dictionaries ECLASS and IEC CDD are based on this standard. The data specification templates specified in this document make it possible to directly use concept descriptions as standardized in these data dictionaries. Additionally, concept descriptions, which do not (yet) exist in these data dictionaries, can be defined using the same schema.

Concept descriptions, whether defined externally in existing data dictionaries or internally as part of the Asset Administration Shell environment, are the foundation for defining submodel templates [24] [16].

IEC 61360-1:2017 is largely compliant to IEC 61360-2:2012 and ISO 3584-42:2010.

Note: for details on how to use the data specifications and for further explanations, please refer directly to IEC 61360.

The following subclauses show some examples from these existing data dictionaries to ease understanding of the data specification templates.

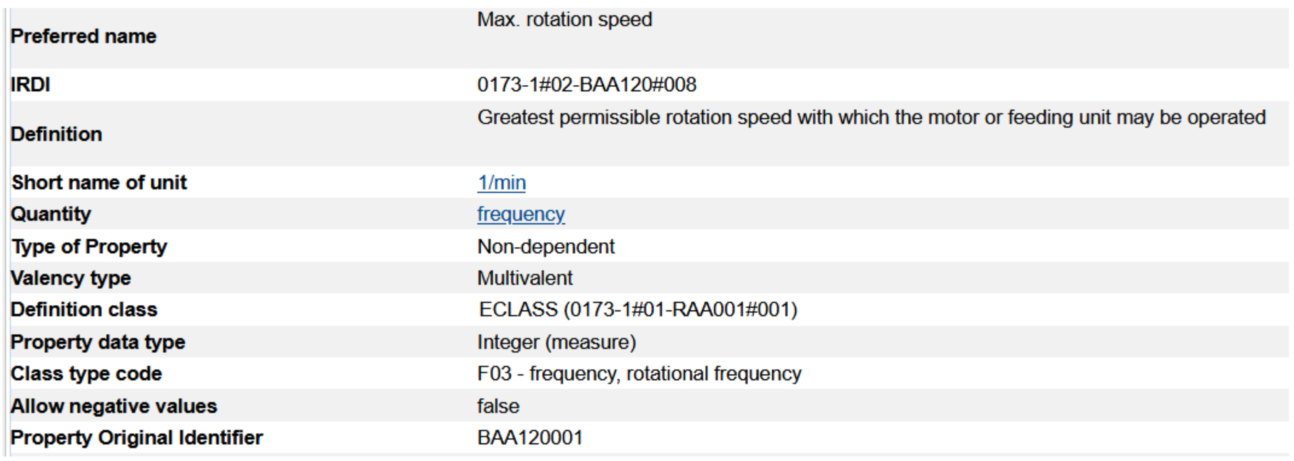
4.2 Concept Descriptions for Properties and Values

The data specification template IEC 61360 introduces additional attributes to define the semantics – i.e. a concept description – of a property or a value based on IEC 61360.

IEC 61360 requests to use IRDIs for the identification of a concept. The Asset Administration Shell allows to use other identifiers besides IRDI. The IRDI, the unique identifier of an IEC 61360 property or value, maps to ConceptDescription/id.

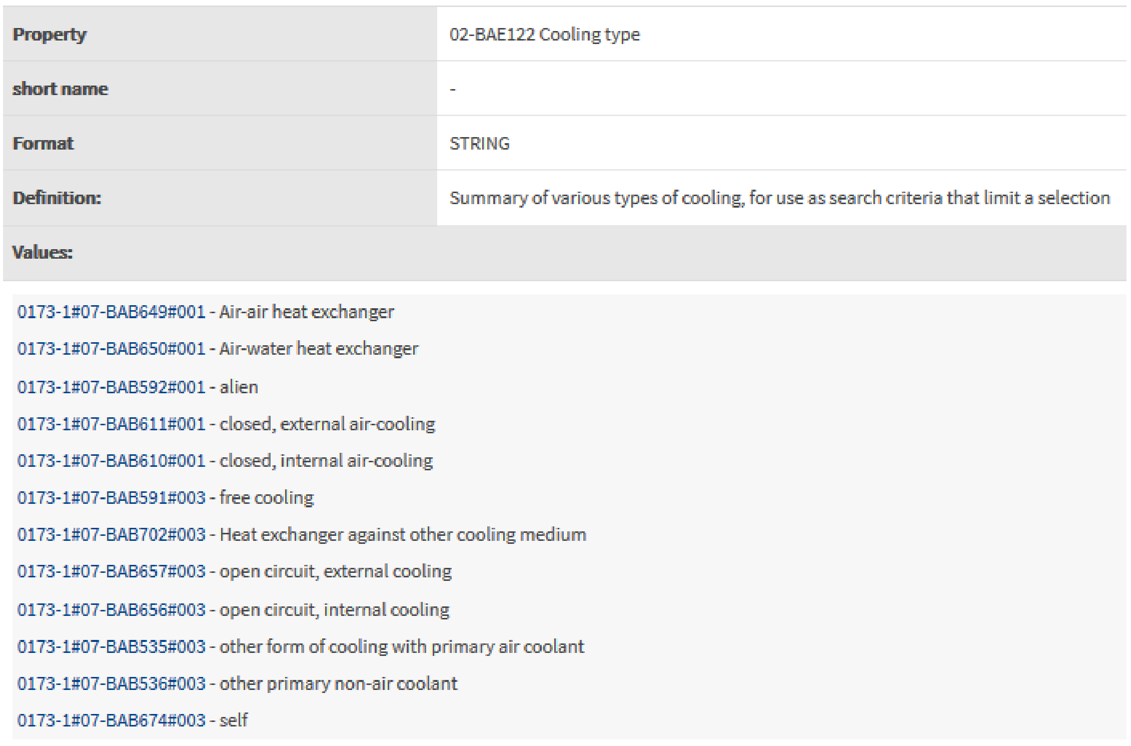
Figure 2 to Figure 5 show examples from ECLASS
Figure 3 shows a property with enumeration type. One of the values in this enumeration is shown in
Figure 4, each value has its own unique ID. The unique identifier of a value (
Figure 4) is also used for *Property/valueId.*

Figure 6 Example for Property with Level Type from IEC CDD shows an example from IEC CDD for a concept description of a *Property* with usage of Level Type (in this example level type MIN, MAX and NOM, see data type). This is a short form of defining a collection of three properties with the same data type and semantics except for the level.



Preferred name	Max. rotation speed
IRDI	0173-1#02-BAA120#008
Definition	Greatest permissible rotation speed with which the motor or feeding unit may be operated
Short name of unit	1/min
Quantity	frequency
Type of Property	Non-dependent
Valency type	Multivalent
Definition class	ECLASS (0173-1#01-RAA001#001)
Property data type	Integer (measure)
Class type code	F03 - frequency, rotational frequency
Allow negative values	false
Property Original Identifier	BAA120001

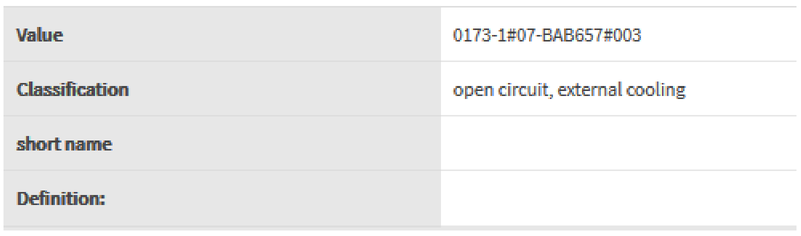
Figure 2 Example Property From ECLASS



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:	

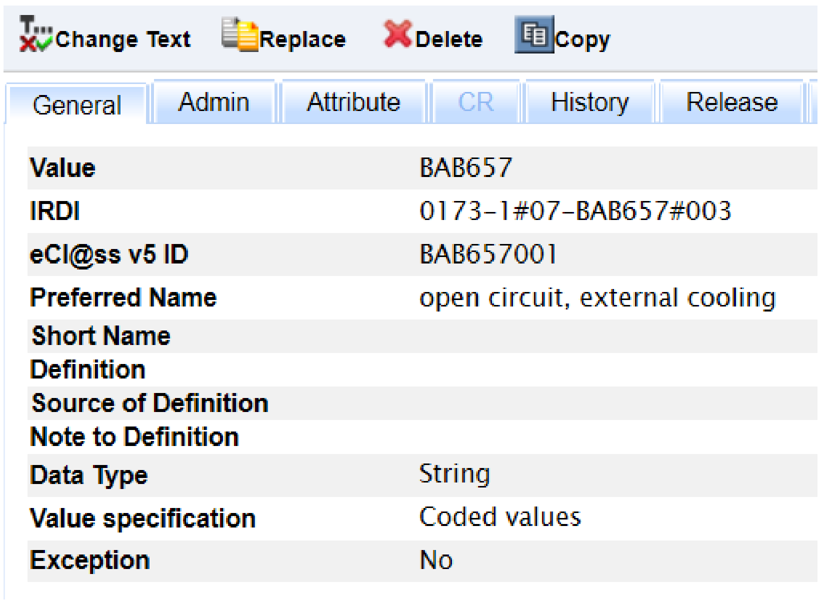
0173-1#07-BAB649#001 - Air-air heat exchanger
0173-1#07-BAB650#001 - Air-water heat exchanger
0173-1#07-BAB592#001 - alien
0173-1#07-BAB611#001 - closed, external air-cooling
0173-1#07-BAB610#001 - closed, internal air-cooling
0173-1#07-BAB591#003 - free cooling
0173-1#07-BAB702#003 - Heat exchanger 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 3 Example Property Description with Value List from ECLASS



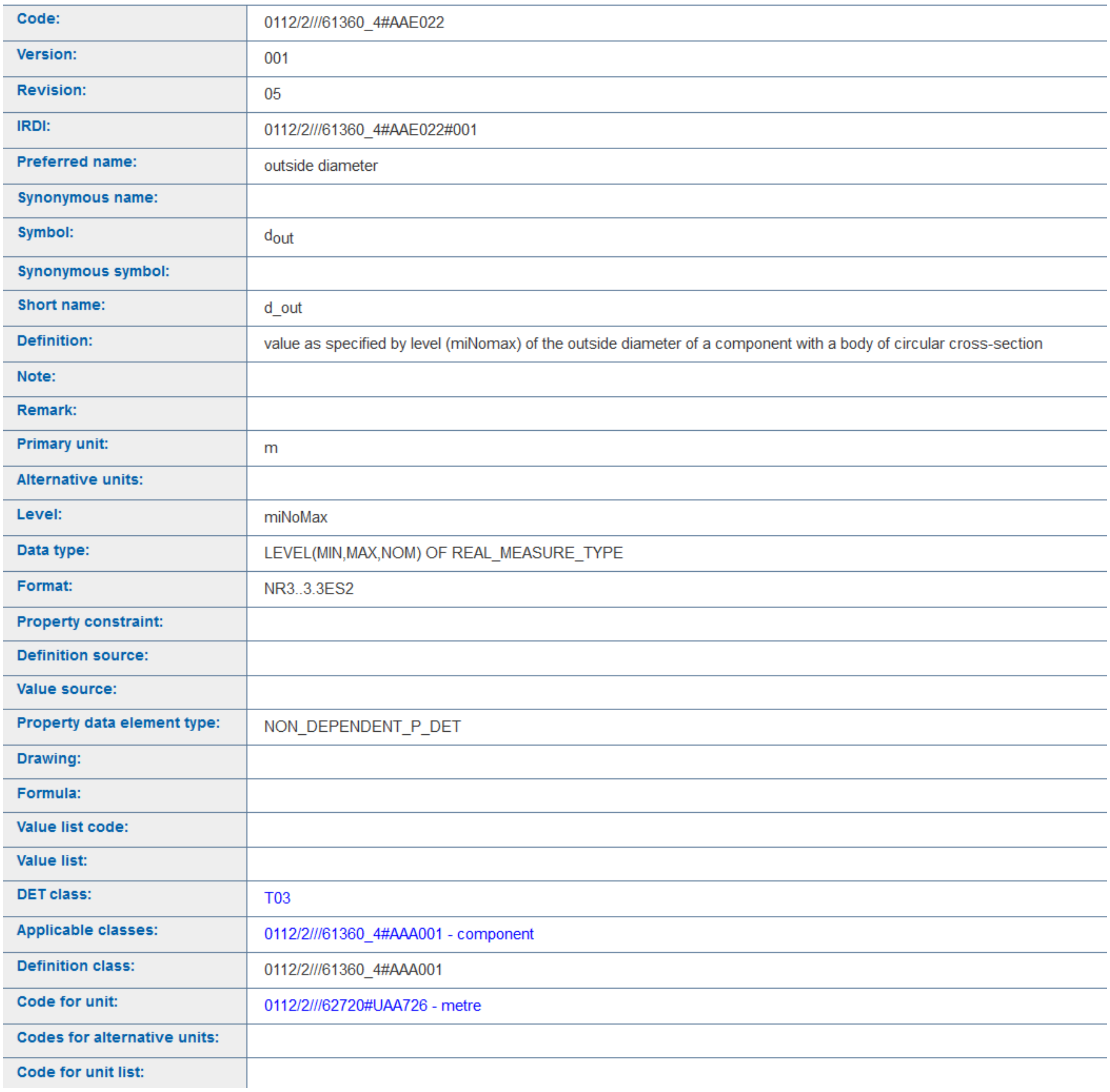
Value	0173-1#07-BAB657#003
Classification	open circuit, external cooling
short name	
Definition:	

Figure 4 Example Value Description from ECLASS



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

Figure 5 Example Value Description from ECLASS Advanced (Editor Modus)



Code:	0112/2///61360_4#AAE022
Version:	001
Revision:	05
IRDI:	0112/2///61360_4#AAE022#001
Preferred name:	outside diameter
Synonymous name:	
Symbol:	d_{out}
Synonymous symbol:	
Short name:	d_out
Definition:	value as specified by level (miNomax) of the outside diameter of a component with a body of circular cross-section
Note:	
Remark:	
Primary unit:	m
Alternative units:	
Level:	miNoMax
Data type:	LEVEL(MIN,MAX,NOM) OF REAL_MEASURE_TYPE
Format:	NR3..3.3ES2
Property constraint:	
Definition source:	
Value source:	
Property data element type:	NON_DEPENDENT_P_DET
Drawing:	
Formula:	
Value list code:	
Value list:	
DET class:	T03
Applicable classes:	0112/2///61360_4#AAA001 - component
Definition class:	0112/2///61360_4#AAA001
Code for unit:	0112/2///62720#UAA726 - metre
Codes for alternative units:	
Code for unit list:	

Figure 6 Example for Property with Level Type from IEC CDD

5 Predefined Data Specification Templates

5.1 Overview

A data specification template specifies 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 is one data specification template supporting IEC 61360 [IEC61360-1]:

- *DataSpecificationIec61360:* defining concept descriptions for both properties and coded values.

Figure 7 Overview Relationship Metamodel Part 1 a & Data Specifications IEC 61360 gives an overview of the data specification template and how it is used in combination with the information model as defined in Part 1 of the document series, namely *DataSpecification*, *DataSpecificationContent,* and *ConceptDescription*.

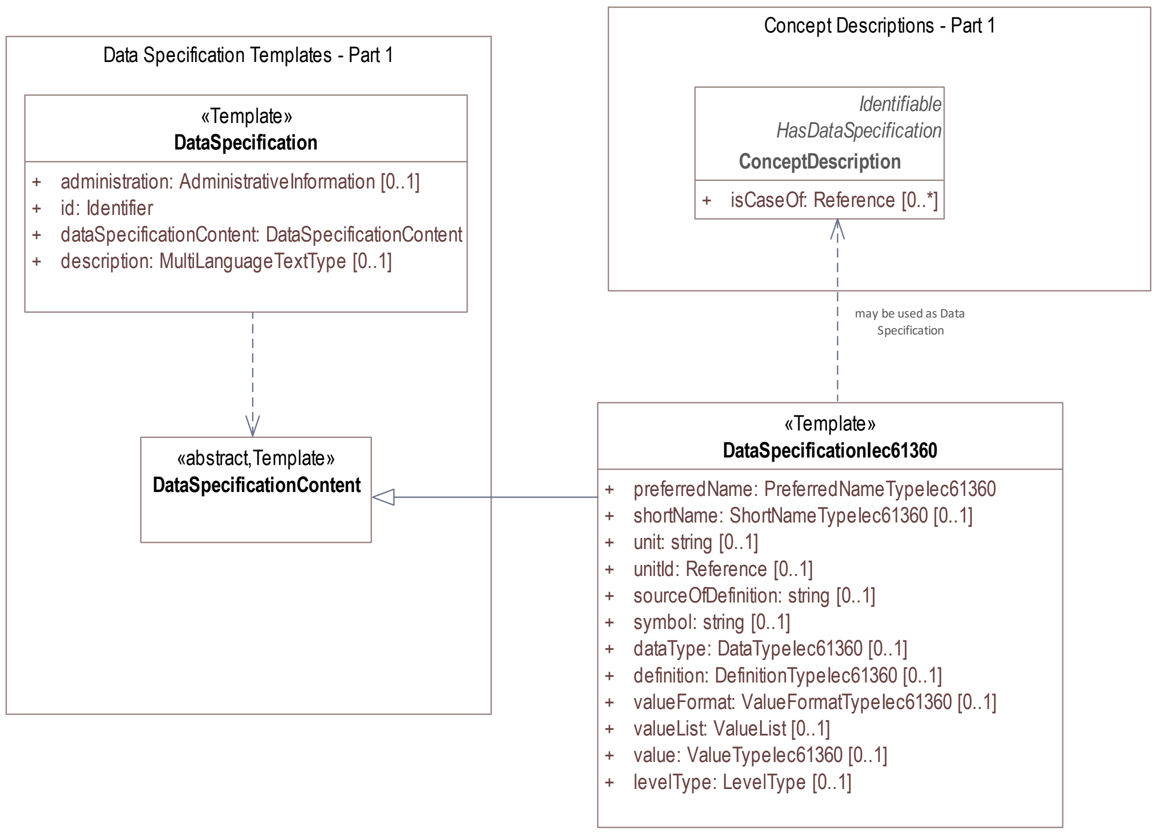

Data Specification Templates - Part 1
«Template»
DataSpecification
+ administration: AdministrativeInformation [0..1]
+ id: Identifier
+ dataSpecificationContent: DataSpecificationContent
+ description: MultiLanguageTextType [0..1]
«abstract,Template»
DataSpecificationContent
Concept Descriptions - Part 1
Identifiable
HasDataSpecification
ConceptDescription
+ isCaseOf: Reference [0..*]
may be used as Data Specification
«Template»
DataSpecificationIec61360
+ preferredName: PreferredNameTypeIec61360
+ shortName: ShortNameTypeIec61360 [0..1]
+ unit: string [0..1]
+ unitId: Reference [0..1]
+ sourceOfDefinition: string [0..1]
+ symbol: string [0..1]
+ dataType: DataTypeIec61360 [0..1]
+ definition: DefinitionTypeIec61360 [0..1]
+ valueFormat: ValueFormatTypeIec61360 [0..1]
+ valueList: ValueList [0..1]
+ value: ValueTypeIec61360 [0..1]
+ levelType: LevelType [0..1]

Figure 7 Overview Relationship Metamodel Part 1 a & Data Specifications IEC 61360

IEC 61360 is a standard that describes how to define the semantics of properties in a data dictionary. Part 1 does not prescribe how to define a concept description; it only supports the definition of concept descriptions. To do so, a data specification template needs to be assigned to the concept description. Which data specification is made available is defined via *HasDataSpecification/dataSpecification*.

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

Note: an xmi representation of the UML model can be found in the repository "aas-specs" in the github project admin-shell-io: https://github.com/admin-shell-io/aas-specs/.

6 Predefined Template for IEC61360 Properties, Value Lists, and Values (normative)

6.1 Data Specification IEC61360 Template Specification

Template:	IEC61360		
administration:	version: 3	revision: 0	creator: IDTA
id:	`https://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0`		
dataSpecificationContent:	DataSpecificationIec61360		
Description (EN):	Data specification template for concept descriptions for properties and values conformant to IEC 61360.		

The id of the template was derived conformant to the rules for semantic IDs for data specifications as defined in Part 1 of the document series [AAS-Part1]):

"`https://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0`"

This ID will be used in *hasDataSpecification/dataSpecification*.

This namespace has the qualifier "IEC:" Examples: *IEC:DataSpecificationIec61360/preferredName* or *IEC: DataSpecificationIec61360/levelType/Min* or *IEC:LevelType/Min*

6.2 Data Specification IEC61360 Attributes

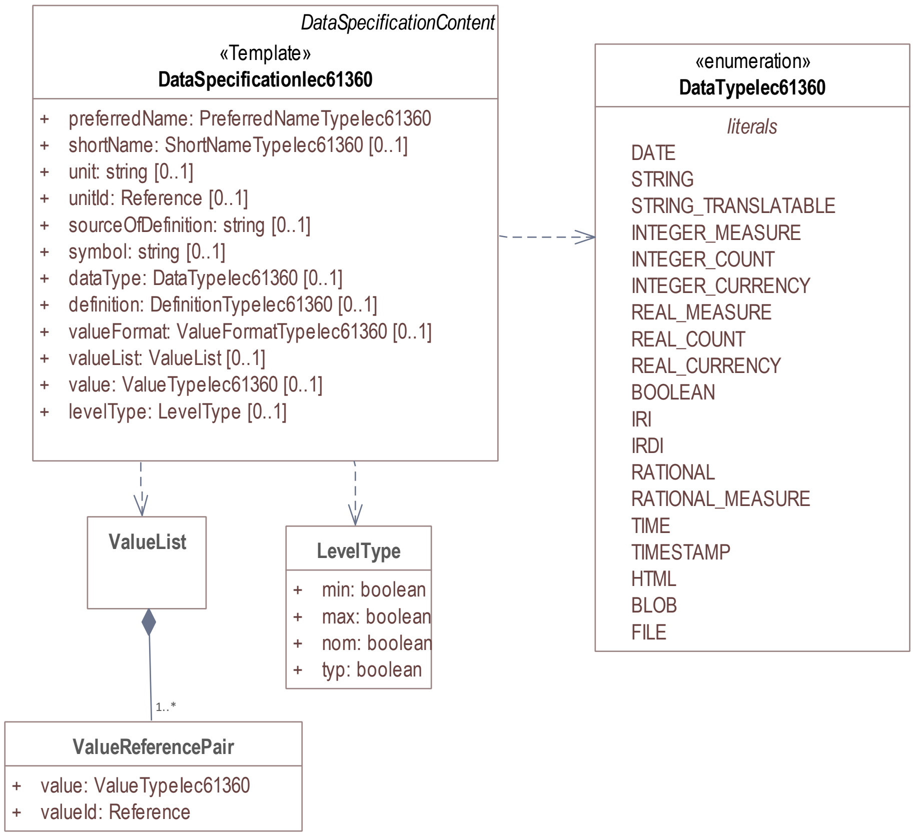

DataSpecificationContent
«Template»
DataSpecificationIec61360
+ preferredName: PreferredNameTypeIec61360
+ shortName: ShortNameTypeIec61360 [0..1]
+ unit: string [0..1]
+ unitId: Reference [0..1]
+ sourceOfDefinition: string [0..1]
+ symbol: string [0..1]
+ dataType: DataTypeIec61360 [0..1]
+ definition: DefinitionTypeIec61360 [0..1]
+ valueFormat: ValueFormatTypeIec61360 [0..1]
+ valueList: ValueList [0..1]
+ value: ValueTypeIec61360 [0..1]
+ levelType: LevelType [0..1]
«enumeration»
DataTypeIec61360
literals
DATE
STRING
STRING_TRANSLATABLE
INTEGER_MEASURE
INTEGER_COUNT
INTEGER_CURRENCY
REAL_MEASURE
REAL_COUNT
REAL_CURRENCY
BOOLEAN
IRI
IRDI
RATIONAL
RATIONAL_MEASURE
TIME
TIMESTAMP
HTML
BLOB
FILE
ValueList
LevelType
+ min: boolean
+ max: boolean
+ nom: boolean
+ typ: boolean
1..*
ValueReferencePair
+ value: ValueTypeIec61360
+ valueId: Reference

Figure 8 Metamodel of Data Specification IEC61360

Class:	DataSpecificationIec61360 <<Template>>		
Explanation:	Content of data specification template for concept descriptions for properties, values, and value lists conformant to IEC 61360. Note: for details, please refer to [IEC61360-1], property, value_list and term Constraint AASc-010: If *DataSpecificationIec61360/value* is not empty, *DataSpecificationIec61360/valueList* shall be empty, and vice versa. Note 1: it is also possible that both *DataSpecificationIec61360/value* and *SpecificationIec61360/valueList* are empty. This is the case for concept descriptions that define the semantics of a property but do not have an enumeration (*valueList)* as data type. Note 2: although it is possible to define a concept description for a value list, it is not possible to reuse this value list. It is only possible to directly add a value list as data type to a specific semantic definition of a property. Constraint AASc-009: If *DataSpecificationIec61360/dataType* is one of *INTEGER_MEASURE, REAL_MEASURE, RATIONAL_MEASURE, INTEGER_CURRENCY, REAL_CURRENCY*, then *DataSpecificationIec61360/unit* or *DataSpecificationIec61360/unitId* shall be defined.		
Inherits from:	DataSpecificationContent		
Attribute	**Explanation**	**Type**	**Card**
preferredName	Preferred name in different languages Note: for details, please refer to [IEC61360-1], preferred_name Constraint AASc-002: *DataSpecificationIec61360-/preferredName* shall be provided at least in English.	PreferredNameTypeIec61360	1
shortName	Short name Note: for details, please refer to [IEC61360-1], short_name	ShortNameTypeIec61360	0..1
unit	Unit in case of a quantitative property Note 1: for details, please refer to [IEC61360-1], unit_in_text Note 2: only the primary unit is supported.	string	0..1
unitId	Unique unit ID Unit and unitId need to be consistent if both attributes are set Note 1: for details, please refer to [IEC61360-1], unit_of_measure Note 2: it is recommended to use an external reference ID.	Reference	0..1
sourceOf-Definition	Source of definition Note: for details, please refer to [IEC61360-1], source_document_of_definition	string	0..1

Class:	DataSpecificationIec61360 <<Template>>		
symbol	Symbol Note: for details, please refer to [IEC61360-1], preferred_letter_symbol	string	0..1
dataType	Data Type Note: for details, please refer to [IEC61360-1], data_type	DataTypeIec61360	0..1
definition	Definition in different languages Note: for details, please refer to [IEC61360-1], definition	DefinitionTypeIec61360	0..1
valueFormat	Value Format Note: for details, please refer to [IEC61360-1], value_format	ValueFormatIec61360	0..1
valueList	Enumerated list of allowed values Note 1: for details, please refer to [IEC61360-1], enumerated_list_of_terms. Note 2: for ease of usage, the value list is modelled as value/valueId list in this data specification template.	ValueList	0..1
value	Value (typically within a value list) Note: for details, please refer to [IEC61360-1], term/preferred_letter_symbol_in_text	ValueTypeIec61360	0..1
levelType	Value represented by up to four variants of a numeric value in a specific role: MIN, NOM, TYP and MAX. Note: for details, please refer to [IEC61360-1], LEVEL_TYPE(MIN,NOM,TYP,MAX)	LevelType	0..1

Note 1: IEC 61360 also requires 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. The same applies to administrative information like the version and revision.
Note 2: *ConceptDescription/idShort* and *DataSpecificationIec61360/shortName* are very similar. However, in this case, *shortName* is explicitly added to the data specification.
Note 3: the same applies to *ConceptDescription/displayName* and *DataSpecificationIec61360/preferredName*.
Note 4: the same applies to *ConceptDescription/description* and *DataSpecificationIec61360/definition*.

6.3 Enumeration Data Type IEC61360

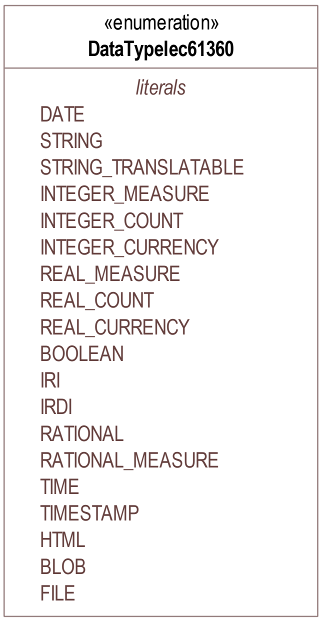


Figure 9 Metamodel of Data Type IEC 61360

Enumeration:	DataTypeIec61360
Explanation:	Enumeration of simple data types for an IEC 61360 concept description using the data specification template *DataSpecificationIec61360*
Set of:	--
Literal	**Explanation**
DATE	values containing a calendar date, conformant to ISO 8601:2004 Format yyyy-mm-dd Note: for details, please refer to [IEC61360-1], specific STRING_TYPE, the DATE_TYPE. Example from IEC 61360-1:2017: "1999-05-31" is the [DATE] representation of: "31 May 1999".
STRING	values consisting of a sequence of characters, which cannot be translated into other languages Note 1: for details, please refer to [IEC61360-1], specific STRING_TYPE, the NON_TRANSLATABLE_STRING_TYPE. Note 2: IEC61360 does not request to use more specific string types like TRANSLATBLE_STRING_TYPE, NON_TRANSLATABLE_STRING_TYPE, DATE_TIME_TYPE, DATE_TYPE, TIME_TYPE, IRDI_STRING, URI_TYPE, and HTML5_TYPE. It is requested to use the more specific data types in the ASS, if applicable[2].

[2] This is also requested in ECLASS, see https://eclass.eu/support/technical-specification/structure-and-elements/value

Enumeration:	DataTypeIec61360
Explanation:	Enumeration of simple data types for an IEC 61360 concept description using the data specification template *DataSpecificationIec61360*
STRING_TRANSLATABLE	values containing string, but which shall be represented as different strings in different languages Note: for details, please refer to [IEC61360-1], specific STRING_TYPE, the TRANSLATABLE_STRING_TYPE
INTEGER_MEASURE	values containing values that are a measure of the type INTEGER. In addition, such a value comes with a physical unit. Note: for details, please refer to [IEC61360-1], specific INTEGER (or INT_TYPE) NUMBER_TYPE, the INT_MEASURE_TYPE
INTEGER_COUNT	values containing values of the type INTEGER, but which are no currencies or measures Note 1: for details, please refer to [IEC61360-1], specific NUMBER_TYPE, the INT_TYPE (or just INTEGER). For more specific data types, INTEGER_MEASURE_TYPE or INTEGER_CURRENCY_TYPE may be used. Note 2: it is requested to use the more specific data types in the ASS, if applicable.
INTEGER_CURRENCY	values containing values of the type INTEGER, which are currencies Note: for details, please refer to [IEC61360-1], specific INTEGER NUMBER_TYPE, the INT_CURRENCY_TYPE
REAL_MEASURE	values containing values that are measures of the type REAL. In addition, such a value comes with a physical unit. Note: for details, please refer to [IEC61360-1], specific REAL NUMBER_TYPE, the REAL_MEASURE_TYPE
REAL_COUNT	values containing numbers that can be written as a terminating or non-terminating decimal; i.e. a rational or irrational number, which is neither a currency nor a measures Note 1: for details, please refer to [IEC61360-1], specific NUMBER_TYPE, the REAL_TYPE. For more specific data types REAL_MEASURE_TYPE or REAL_CURRENCY_TYPE may be used. Note 2: it is requested to use the more specific data types in the AAS, if applicable.
REAL_CURRENCY	values containing values of the type REAL, which are currencies Note: for details, please refer to [IEC61360-1], specific REAL NUMBER_TYPE, the REAL_CURRENCY_TYPE
BOOLEAN	values representing truth of logic or Boolean algebra (TRUE, FALSE) Note 1: for details, please refer to [IEC61360-1], BOOLEAN_TYPE.

Enumeration:	DataTypeIec61360
Explanation:	Enumeration of simple data types for an IEC 61360 concept description using the data specification template *DataSpecificationIec61360*
	Note 2: in IEC 61360, the values are Yes and No. In the AAS, the values are TRUE (for "Yes") and FALSE (for "No").
IRI	values containing values of the type STRING conformant to Rfc 3987 Note 1: for details, please refer to [IEC61360-1], specific STRING_TYPE, the URI_TYPE. Note 2: However, the AAS supports the more generic IRI. An IRI type particularly allows to express a URL or a URI. If the IRI represents an address to a file, FILE shall be used.
IRDI	values conforming to ISO/IEC 11179 series global identifier sequences Note 1: for details, please refer to [IEC61360-1], specific STRING_TYPE, the IRDI_STRING. Note 2: IRDI can be used instead of the more specific data types ICID or ISO29002_IRDI. Note 3: ICID values are values conformant to an IRDI, where the delimiter between RAI and ID is "#", while the delimiter between DI and VI is confined to "##". Note 4: ISO29002_IRDI values are values containing a global identifier that identifies an administrated item in a registry. The structure of this identifier complies with the 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).
RATIONAL	Values containing values of the type RATIONAL, which are no measures Examples: ½, ¾ or 7/2 Note 1: for details, please refer to [IEC61360-1], specific NUMBER_TYPE, the RATIONAL_TYPE. Note 2: it is requested to use the more specific data types in the AAS, if applicable.
RATIONAL_MEASURE	values containing values of the type RATIONAL. In addition, such a value comes with a physical unit. Note: for details, please refer to [IEC61360-1], specific RATIONAL NUMBER_TYPE, the RATIONAL_MEASURE_TYPE
TIME	values containing a time conformant to ISO 8601:2004 but restricted to what is allowed in the corresponding type in xml. Format hh:mm (ECLASS)

Enumeration:	DataTypeIec61360
Explanation:	Enumeration of simple data types for an IEC 61360 concept description using the data specification template *DataSpecificationIec61360*
	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). Note: for details, please refer to [IEC61360-1], specific STRING_TYPE, the TIME_TYPE
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) Note: for details, please refer to [IEC61360-1], specific STRING_TYPE, the DATE_TIME_TYPE.
FILE	values containing an address to a file. The values are of the type URI and can represent an absolute or relative path. Note: [IEC61360-1] does not explicitly support the file type. It would map to the URI_TYPE.
HTML	Values containing string with any sequence of characters, using the syntax of HTML5 (see W3C Recommendation 28:2014) Note: for details, please refer to [IEC61360-1], specific STRING_TYPE, the HTML5_TYPE.
BLOB	values containing the content of a file. Values may be binaries. *HTML conformant to HTML5* is a special blob. In IEC61360, *binary* is a sequence of bits, each bit being represented by "0" and "1" only. A binary is a blob. However, a blob may also contain other source code. Note: for details, please refer to [IEC61360-1], BINARY_TYPE

6.4 Level Type

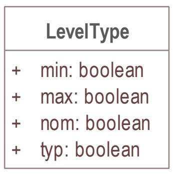


Figure 10 Metamodel of Level Type

Class:	LevelType		
Explanation:	Value represented by up to four variants of a numeric value in a specific role: MIN, NOM, TYP, and MAX. True means that the value is available, false means the value is not available. Note: for details, please refer to [IEC61360-1], LEVEL_TYPE EXAMPLE from [IEC61360-1]: in case of a property which is of the LEVEL_TYPE min/max − Note: for details, please refer to [IEC61360-1], LEVEL_TYPE		
Inherits from:	DataSpecificationContent		
Attribute	**Explanation**	**Type**	**Card.**
min	Minimum of the value	boolean	1
nom	Nominal value (value as designated)	boolean	1
typ	Value as typically present	boolean	1
max	Maximum of the value	boolean	1

Note: This is how the AAS deals with the following combinations of level types:

1. If all attributes are false, the concept is mapped to a Property and level type is ignored.
2. If a maximum of one attribute is set to true, the concept is mapped to a Property.
3. If min and max are set to true, the concept is mapped to a Range.
4. If more than one attribute is set to true, does not include min and max only (see second case), the concept is mapped to a *SubmodelElementCollection* with the corresponding number of Properties. Example: If the attributes min and nom are set to true, the concept is mapped to a *SubmodelElementCollection* with two Properties: min and nom. The data type of both Properties is the same.

In the cases 2 and 4, the *semanticId* of the Property or Properties within the *SubmodelElementCollection* needs to include information about the level type. Otherwise, the semantics is not described in a unique way. In [27], IRDI paths are introduced. However, no rules of how to map IRDI paths to *References* for semanticIds have yet been defined.

It is not recommended to use level type when defining concept descriptions for Properties, except for ranges (i.e. min and max). This is considered to be a deprecated way of defining property sets. See also [27], where one proposal on how to deal with level type is to remove the level type and to define several properties instead.

6.5 Value List Attributes

"*ValueList*" allows to define an enumeration type for a property. The value list is a set of value reference pairs.

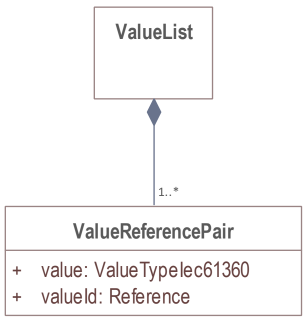


Figure 11 Metamodel of Value List

Class:	ValueList		
Explanation:	A set of value reference pairs Note: for details, please refer to [IEC61360-1], value_list/enumerated_list_of_terms.		
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.		
Inherits from:	--		
Attribute	**Explanation**	**Type**	**Card.**
value	the value of the referenced concept definition of the value in *valueId.*	ValueTypeIec61360	1
valueId	Global unique ID of the value Note: it is recommended to use a global reference.	Reference	1

7 Mapping IEC 61360 Data Types to XSD Data Types

Using a concept description requires mapping the data type of the concept description to a conformant type in xsd (for example in *Property/valueType*).

Examples for the different IEC 61360 data types can be found here: https://eclass.eu/support/technical-specification/structure-and-elements/value.

Data Type IEC 61360	**xsd Value Type**[3]	**Example Values IEC 61360**[4]
DATE	xs:date	1979-01-15
STRING	xs:string	"DN 700" "10 Mbps"
STRING_TRANSLATABLE	*Mapped to MultiLanguageProperty, i.e. type MultiLanguageText* Note: for details, please see Part 1 of the document series "Details of the Asset Administration Shell".	
INTEGER_MEASURE	xs:integer	1 10 111
INTEGER_COUNT	xs:integer	1 10 111
INTEGER_CURRENCY	xs:integer	1 10 111
REAL_MEASURE	xs:double or xs:float (depending on needed precision)	1.5 102.35

[3] *Property/valueType*, *Range/valueType*, etc. are each of type *DataTypeDefXsd.* Note: for submodel elements like *Blob* and *File* or *MultiLanguageProperty and ReferenceElement*, there is no explicitly defined *valueType* attribute because the data type is implicitly defined and fix (*BlobType*, *PathType* or *MultiLanguageTextType, Reference*).

[4] Source for most examples for the different IEC 61360 data types: https://eclass.eu/support/technical-specification/structure-and-elements/value. The IRDI example for STRING was moved to IRDI.

Data Type IEC 61360	xsd Value Type[3]	Example Values IEC 61360[4]
REAL_COUNT	xs:double or xs:float (depending on needed precision)	1.5 102.35
REAL_CURRENCY	xs:double or xs:float (depending on needed precision)	1.5 102.35
BOOLEAN	xs:boolean with "Yes" mapped to "true" and "No" mapped to "false"	Yes No
IRI	xs: anyURI or mapped to ReferenceElement	http://www.eclass-cdp.com
IRDI	xs:string *or mapped to ReferenceElement* Note: for details, please see Part 1 of the document series "Details of the Asset Administration Shell".	0173-1#01-ADG629#001
RATIONAL	xs:string	1/3 1 2/3
RATIONAL_MEASURE	xs:string	1/3 1 2/3
TIME	xs:time	12:45
TIMESTAMP	xs:dateTime	1979-01-15T12:45:00Z
FILE	*Mapped to submodel element File, i.e. type PathType* Note: for details, please see Part 1 of the document series "Details of the Asset Administration Shell".	./documents/example.pdf
HTML	*Mapped to submodel element Blob, i.e. type BlobType* Note: for details, please see Part 1 of the document series "Details of the Asset	

Data Type IEC 61360	**xsd Value Type**[3]	**Example Values IEC 61360**[4]
	Administration Shell".	
BLOB	*Mapped to submodel element Blob, i.e. type BlobType* Note: for details, please see Part 1 of the document series "Details of the Asset Administration Shell".	

Figure 12 Mapping IEC 61360 Data Types to xsd Data Types

8 Category of Concept Descriptions

Note: the attribute category of referables was set to deprecated in V3.0 of Part 1. Hence this clause informs about the meaning, in case applications are still using the attribute category.

Although the IEC 61360 attributes listed in this template are defined for properties and values 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 Table 1, Table 2 and Table 4.

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

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

See Figure 7 Overview Relationship Metamodel Part 1 a & Data Specifications IEC 61360 on how data specification templates are related to concept descriptions. Figure 13 lists all categories used for concept descriptions[5].

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

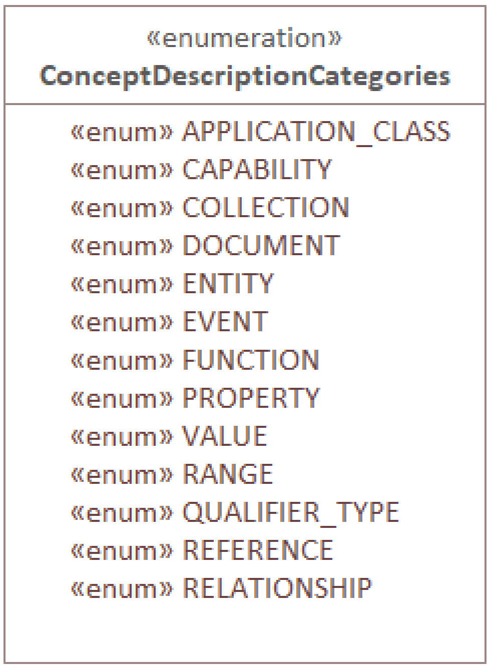


Figure 13 Categories of Concept Descriptions (non-normative)

[5] Note: although the possible categories are listed as enumeration, no enumeration has been defined for Referable/category.

Attribute[6]	Property	Property	Property	Multi-Language-Property	Range
Category of Concept Description	VALUE	PROPERTY	PROPERTY	PROPERTY	PROPERTY
Category of Submodel-Element	**CONSTANT**	**VARIABLE**	**PARAMETER**	**--**	**--**
preferredName[7]	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[8]	m[8]	m[8]	STRING_TRANSLATABLE	INTEGER_* or REAL_*
definition	(m)	m	m	m	m
valueFormat	o	o	o	--	o
valueList	--	o	o	--	--
value	m	--	--	--	--
valueId	o	--	--	--	--
levelType	--	--	--	--	Min = true Max = true

Table 1 IEC61360 Data Specification Template for Properties and Ranges

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

7 Mandatory in at least one language. Preferably, an English preferred name should always be defined.

8 All IEC 61360 data types except STRING_TRANSLATABLE, IRI, IRDI, HTML, FILE, BLOB.

Attribute[6]	Reference-Element	File[9]	Blob[9]	Capability[9]	Relationship-Element[9]	AnnotatedRelationship-Element[9]
Category of Concept Description	REFERENCE	DOCUMENT	DOCUMENT	CAPABILITY	RELATIONSHIP	RELATIONSHIP
Category of Submodel-Element	--	--	--	--	--	--
preferredName[7]	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	--	--	--	--	--	--

Table 2 IEC61360 Data Spec. Template for Other Data Elements, Relationships Elements and Capabilities

[9] Template only used until explicit template is available for defining the corresponding types of elements.

Attribute	SubmodelElementList[9]	SubmodelElementCollection[9]	Operation[9]	EventElement[9]	Entity[9]
Category of Concept Description	COLLECTION	ENTITY	FUNCTION	EVENT	ENTITY
Category of Submodel-Element	**--**	**--**	**--**	**--**	**--**
preferredName[7]	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	--	--	--	--	--

Table 3 IEC612360 Data Specification Template for Other Submodel Elements

Attribute	**Submodel**[9]	**Qualifier**[9]	**SpecificAssetId**
Category of Concept Description	APPLICATION_CLASS	QUALIFIER_TYPE	PROPERTY
Category of Element	**--**	**--**	**--**
preferredName	m	m	m
shortName	(m)	(m)	(m)
unit	--	--	
unitId	--	--	--
sourceOfDefinition	o	o	o
symbol	--	--	--
dataType	--	m	m
definition	m	m	m
valueFormat	--	o	o
valueList	--	o	--
value	--	--	--
valueId	--	--	--
levelType	--	--	--

Table 4 Other Elements with semanticId

9 Cross-Constraints and Invariants for Predefined Data Specifications (normative)

9.1 General

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

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

Note: these constraints include elements of Part 1, V3.0 of the document series "Details of the Asset Administration Shell" [AAS-Part1].

9.2 Constraints for Data Specification IEC61360

Constraint AASc-3a-004: For a *ConceptDescription* with *category PROPERTY* or *VALUE* using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), *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*.

Note: categories are deprecated since V3.0 of Part 1 of the document series "Details of the Asset Administration Shell".

Constraint AASc-3a-005: For a *ConceptDescription* with *category* REFERENCE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), *DataSpecificationIec61360/dataType* shall be one of *STRING, IRI, IRDI.*

Note: categories are deprecated since V3.0 of Part 1 of the document series "Details of the Asset Administration Shell".

Constraint AASc-3a-006: For a *ConceptDescription* with *category* DOCUMENT using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), *DataSpecificationIec61360/dataType* shall be one of *FILE, BLOB, HTML*.

Note: categories are deprecated since V3.0 of Part 1 of the document series "Details of the Asset Administration Shell".

Constraint AASc-3a-007: For a *ConceptDescription* with *category* QUALIFIER_TYPE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), *DataSpecificationIec61360/dataType* is mandatory and shall be defined.

Note: categories are deprecated since V3.0 of Part 1 of the document series "Details of the Asset Administration Shell".

Constraint AASc-3a-008: For a *ConceptDescription* using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), *DataSpecificationIec61360/definition* is mandatory and shall be defined at least in English. Exception: the concept description describes a value, i.e. *DataSpecificationIec61360/value* is defined.

Constraint AASc-3a-003: For a *ConceptDescription* referenced via *ValueList/valueId* and using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), *DataSpecificationIec61360/value* shall be set.

Constraint AASc-3a-050: If the *DataSpecificationContent DataSpecificationIec61360* is used for an element, 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.

10 Primitive and Simple Data Types (normative)

10.1 Predefined Simple Data Types

The metamodel of the Asset Administration Shell [AAS-Part1] uses basic data types as defined in the XML Schema Definition (XSD)[10]. For an overview of the types used in this document, see Table 5. 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)[11].

Table 5 Simple Data Types Used in Metamodel

Source	Basic Data Type	Value Range	Sample Values
xsd	boolean	true, false	true, false
xsd	string	Character string (but not all Unicode character strings)	"Hello world", "Καλημέρα κόσμε", "ハローワールド"
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.

10.2 Basic and Primitive Data Types

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

Primitive	Definition	Value Examples
DefinitionTypeIec61360	*LangStringSet* Each langString within the array of strings has a length of maximum 1,023 and a minimum of 1 characters.	"Greatest permissible rotation speed with which the motor or feeding unit may be operated." Note: see Figure 2

LangStringSet	*Array of elements of type langString*
	
	Note 1: langString is a RDF data type.
	Note 2: a langString is a string value tagged with a language code.
	
	The realization of a technology depend on the serialization rules.
	
	Note: as defined in Part 1, [AAS-Part1].
		In xml:
		
		`<aas:langString lang="EN">This is a multi-language value in English</aas:langString>`
		
		`<aas:langString lang="DE"> Das ist ein Multi-Language-Wert in Deutsch </aas:langString>`
		
		In rdf:
		
		`"This is a multi-language value in English"@en ;`
		
		`"Das ist ein Multi-Language-Wert in Deutsch"@de`
		
		In JSON:
		
		`"description": [`
		`{`

[10] https://www.w3.org/XML/Core/, formerly https://www.w3.org/XML/Schema

[11] see: https://www.w3.org/TR/rdf11-concepts/

Primitive	Definition	Value Examples

		"language":"en",
		"text": "This is a multi-language value in English."
		},
		{
		"language":"de",
		"text": "Das ist ein Multi-Language-Wert in Deutsch."
		}
		]

PreferredNameTypeIec61360	*LangStringSet* Each *string* with a length of maximum 255 and minimum of 1 characters. Note 1: it is advised to keep the length of the name limited to 35 characters. Note 2: for details, please refer to [IEC61360-1], preferred_name	"max. rotation speed"@EN Note: see Figure 2.
ShortNameTypeIec61360	*LangStringSet* Each *string* with a length of maximum 18 and a minimum of 1 characters. Note: for details, please refer to [IEC61360-1], short_name	"d_out" Note: See Figure 6
ValueFormatTypeIec61360	*string* Note: for details, please refer to [IEC61360-1], value_format. The value format is based on ISO 13584-42 and IEC 61360-2.	"NR3..3.3ES2" Note: see Figure 6
ValueTypeIec61360	*string* with a length of maximum 2000 and minimum of 1 characters.	"Blue" "1000"

Table 6 Primitive DataTypes Used in Metamodel

11 Mappings to Data Formats to Share I4.0-Compliant Information (normative)

11.1 General

Part 1 of the document series introduces the need for different serialization formats and described when which format is used. Part 1 also introduces the implementation guide for embedded data specifications. Hence, only the links to the different schemas derived for the formats XML, JSON, and RDF are provided in the following. Further information can be found in [AAS-Part1].

11.2 XML

The metamodel of an Asset Administration Shell needs to be serialized for import and export scenarios. XML is a possible serialization format.

Note 1: the xml schema (.xsd files) is maintained in the repository "aas-spec" of the github project admin-shell-io [25]: aas-specs-3.0\schemas\xml.

Note 2: the mapping rules of how to derive the xml schema from the technology-neutral metamodel as defined in this specification can be found here: aas-specs-3.0\schemas\xml\Readme.md#xml-mappingrules.

Note 3: example files can be found here: aas-specs-3.0\schemas\xml\examples.

11.3 JSON

JSON[12] (JavaScript Object Notation) is a further serialization format that serializes the metamodel of an Assest Administration Shell for import and export scenarios.

Additionally, JSON format is used to describe the payload in the http/REST API for active Asset Administration Shells [14].

Note 1: the JSON schema (.json files) is maintained in the repository "aas-spec" of the github project admin-shell-io [25]: h aas-specs-3.0\schemas\json

Note 2: the mapping rules of how to derive the JSON schema from the technology-neutral metamodel as defined in this specification can be found here: aas-specs-3.0\schemas\json\Readme.md#json-mappingrules

Note 3: example files can be found here: aas-specs-3.0\schemas\json\examples.

11.4 RDF

The Resource Description Framework (RDF) [26] is the 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.

Note 1: the RDF scheme/OWL files (.ttl files) are maintained in the repository "aas-spec" of the github project admin-shell-io [25]: aas-specs-3.0\schemas\rdf

Note 2: the mapping rules of how to derive the RDF schema from the technology-neutral metamodel as defined in this specification can be found here: aas-specs-3.0\schemas\rdf\Readme.md#rdf-mappingrules

Note 3: example files can be found here: aas-specs-3.0\schemas\rdf\examples

[12] see: https://tools.ietf.org/html/rfc8259 or https://www.ecma-international.org/publications/standards/Ecma-404.htm

12 Summary and Outlook

This document provides a metamodel for specifying data specification templates for defining concept descriptions for properties and values. These data specification templates are conformant to IEC 61360.

This document is part of the document series "Asset Administration Shell in Detail".

Additional parts of the document series cover (see [14]):

- the information model that is the basis for file exchange and interface payload definition (Part 1),
- a file exchange format AASX (Part 5),
- interfaces and APIs for accessing the information of Asset Administration Shells (access, modify, query, and execute information and active functionality; Part 2),
- security aspects including access control (part 3),
- physical units as used to define the semantics of quantifiable properties in IEC 61360 (Part 3b).

Annex A. Background Information

General

This clause provides general information about sources of information and relevant concepts for the data specification under consideration, as well as its usage in the context of the Asset Administration Shell. It is not normative.

Data Elements (Part 1)

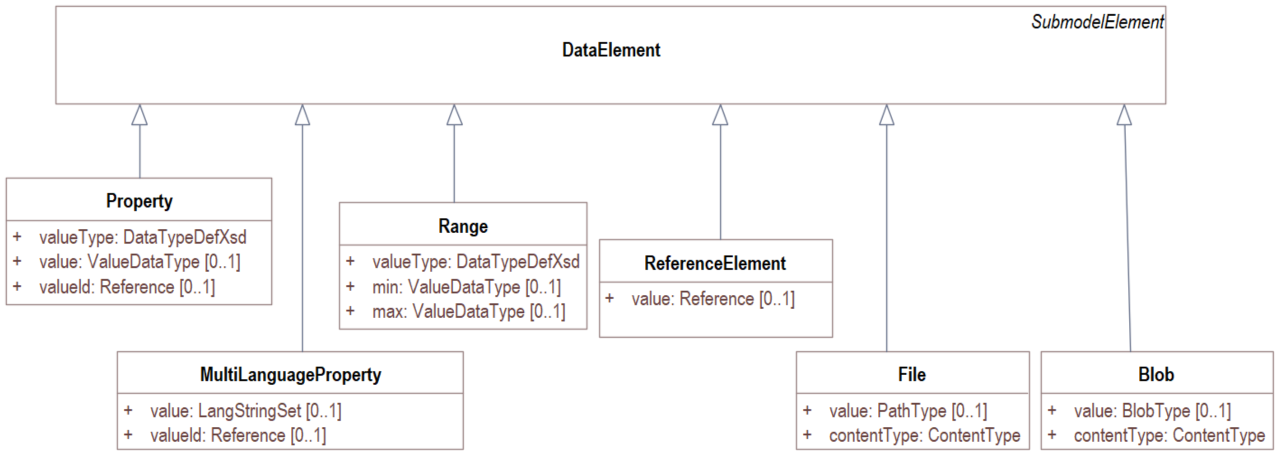

SubmodelElement
DataElement
Property
+ valueType: DataTypeDefXsd
+ value: ValueDataType [0..1]
+ valueId: Reference [0..1]
Range
+ valueType: DataTypeDefXsd
+ min: ValueDataType [0..1]
+ max: ValueDataType [0..1]
ReferenceElement
+ value: Reference [0..1]
MultiLanguageProperty
+ value: LangStringSet [0..1]
+ valueId: Reference [0..1]
File
+ value: PathType [0..1]
+ contentType: ContentType
Blob
+ value: BlobType [0..1]
+ contentType: ContentType

Figure 14 Metamodel of Data Elements (Part 1)

The data specification template IEC61360 is relevant for the definition of concept descriptions for data elements (Figure 14). Submodel Elements inherit from *hasSemantics*, i.e. they have a semanticId and optionally some additional supplementary semantic IDs (Figure *15*).

Figure 7 Overview Relationship Metamodel Part 1 a & Data Specifications IEC 61360 gives an overview of the relationship of concept descriptions (*ConceptDescription*) and data specifications (DataSpecification, *DataSpecificationContent* and *HasDataSpecification*) from Part 1 for this concrete data specification template.

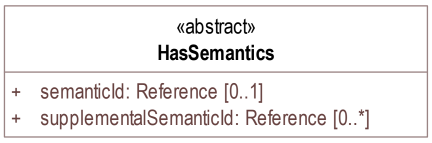

«abstract»
HasSemantics
+ semanticId: Reference [0..1]
+ supplementalSemanticId: Reference [0..*]

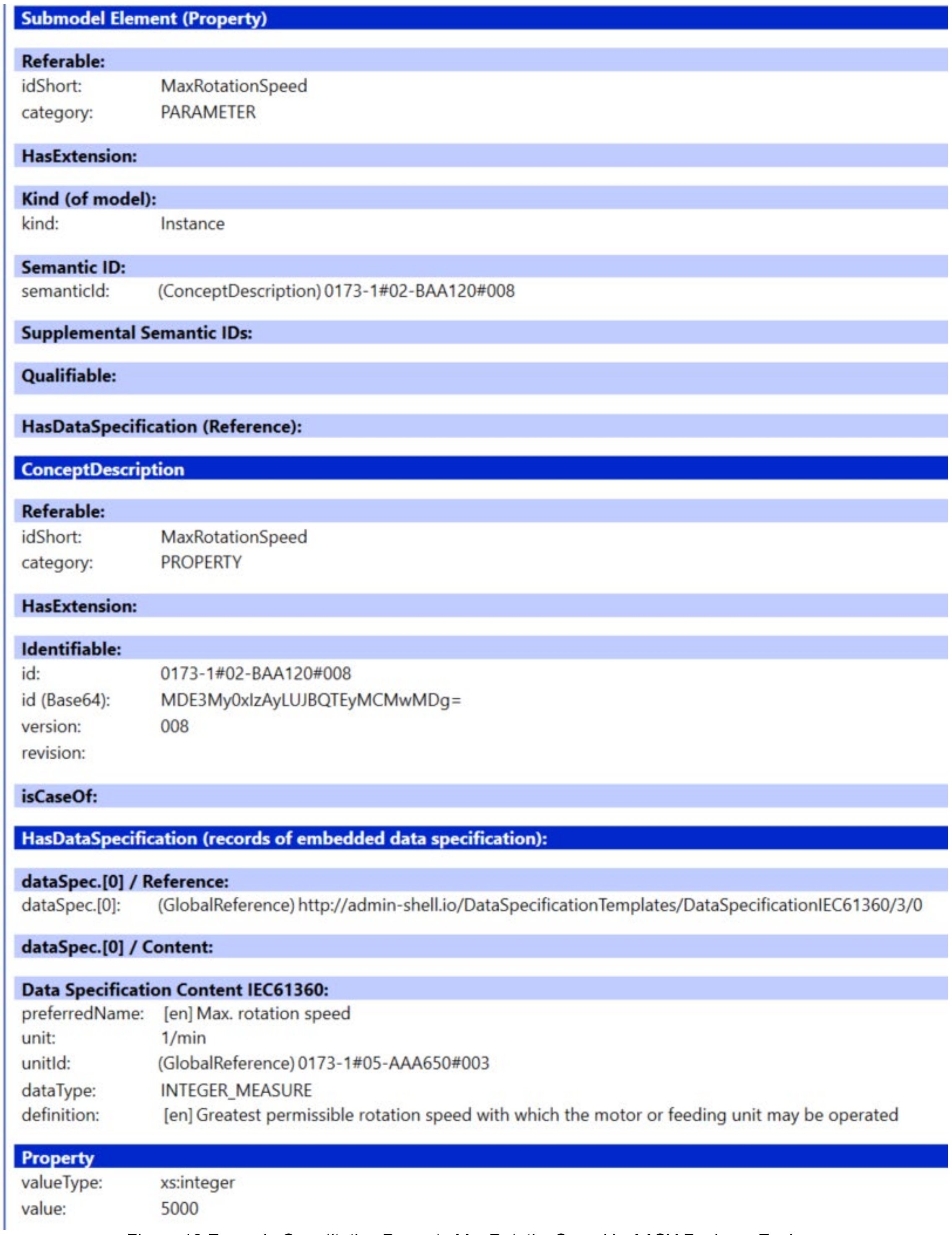
Figure 15 Metamodel of HasSemantics (Part 1)

Clause 8 describes how to use the data specification template to describe further of the metamodel as specified in Part 1 that may also have semantics assigned to them (by inheriting from HasSemantics): Submodel, all other SubmodelElements, SpecificAssetId, Qualifier, and Extension. In these cases, the preferred name and the definition are mainly used to provide a minimum of information on what the corresponding value is about.

Examples

Figure 16 shows an example of a property with idShort "MaxRotationSpeed" with a semantic ID referring to a concept description "MaxRotationSpeed". The concept description shows that MaxRotationSpeed is a quantitative property because the data type is one of *_MEASURE, namely INTEGER_MEASURE. In this case, the definition of a physical unit is mandatory. It is "1/min" for MaxRotationSpeed. A unique ID is also provided for this physical unit. Concept descriptions for physical units are described e.g. in Part 4b of this document series on the Details of the Asset Administration Shell.

The type INTEGER_MEASURE of the concept description is mapped to xs:integer of the property.



Submodel Element (Property)

Referable:

idShort:	MaxRotationSpeed
category:	PARAMETER

HasExtension:

Kind (of model):

kind:	Instance

Semantic ID:

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

Supplemental Semantic IDs:

Qualifiable:

HasDataSpecification (Reference):

ConceptDescription

Referable:

idShort:	MaxRotationSpeed
category:	PROPERTY

HasExtension:

Identifiable:

id:	0173-1#02-BAA120#008
id (Base64):	MDE3My0xIzAyLUJBQTEyMCMwMDg=
version:	008
revision:	

isCaseOf:

HasDataSpecification (records of embedded data specification):

dataSpec.[0] / Reference:

dataSpec.[0]:	(GlobalReference) http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/3/0

dataSpec.[0] / Content:

Data Specification Content IEC61360:

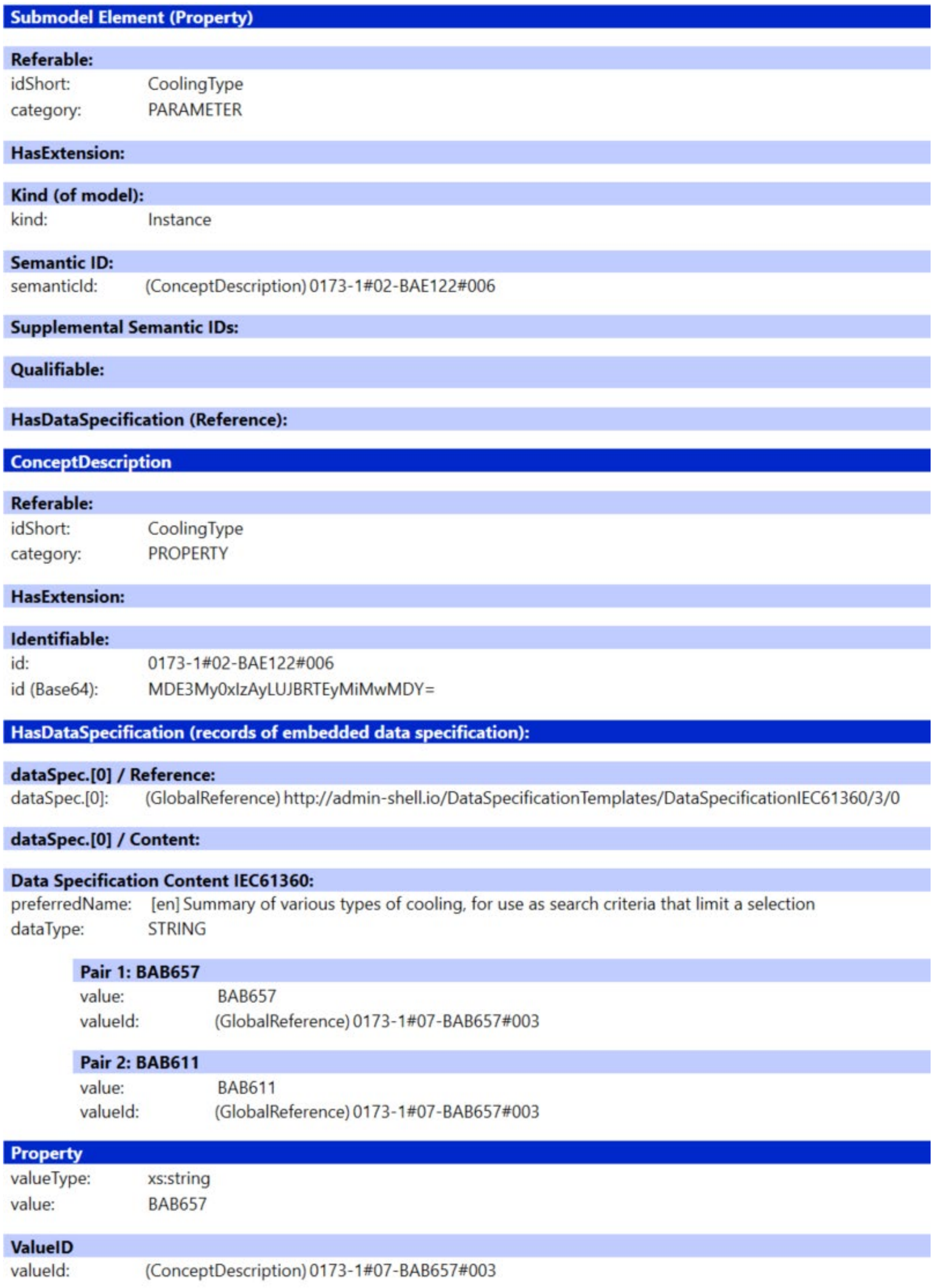
preferredName:	[en] Max. rotation speed
unit:	1/min
unitId:	(GlobalReference) 0173-1#05-AAA650#003
dataType:	INTEGER_MEASURE
definition:	[en] Greatest permissible rotation speed with which the motor or feeding unit may be operated

Property

valueType:	xs:integer
value:	5000

Figure 16 Example Quantitative Property MaxRotationSpeed in AASX Package Explorer

Figure 7 shows a property "CoolingType". Its semanticId references a concept description that defines a value list (*DataSpecificationIec612360/valueList*) with two values BAB657 and BAB611.



Submodel Element (Property)

Referable:

idShort:	CoolingType
category:	PARAMETER

HasExtension:

Kind (of model):

kind:	Instance

Semantic ID:

semanticId:	(ConceptDescription) 0173-1#02-BAE122#006

Supplemental Semantic IDs:

Qualifiable:

HasDataSpecification (Reference):

ConceptDescription

Referable:

idShort:	CoolingType
category:	PROPERTY

HasExtension:

Identifiable:

id:	0173-1#02-BAE122#006
id (Base64):	MDE3My0xIzAyLUJBRTEyMiMwMDY=

HasDataSpecification (records of embedded data specification):

dataSpec.[0] / Reference:

dataSpec.[0]:	(GlobalReference) http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/3/0

dataSpec.[0] / Content:

Data Specification Content IEC61360:

preferredName:	[en] Summary of various types of cooling, for use as search criteria that limit a selection
dataType:	STRING

Pair 1: BAB657

value:	BAB657
valueId:	(GlobalReference) 0173-1#07-BAB657#003

Pair 2: BAB611

value:	BAB611
valueId:	(GlobalReference) 0173-1#07-BAB657#003

Property

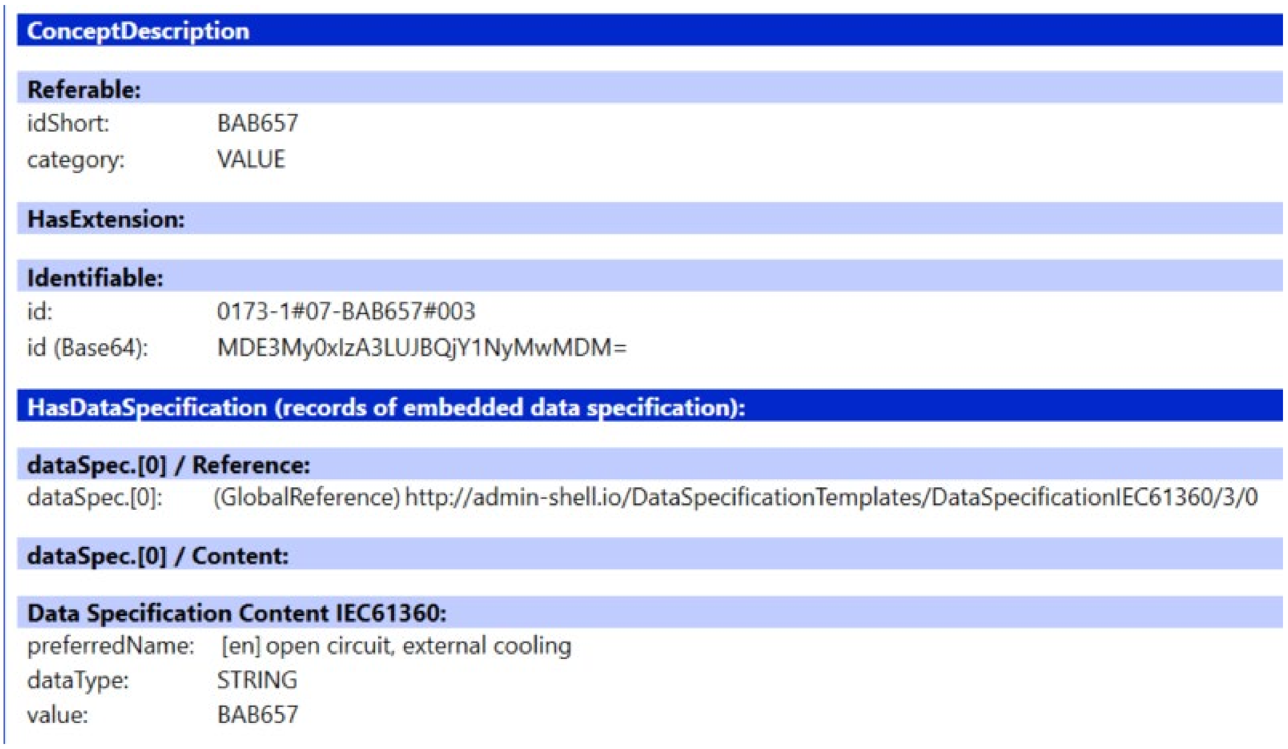
valueType:	xs:string
value:	BAB657

ValueID

valueId:	(ConceptDescription) 0173-1#07-BAB657#003

Figure 17 Example Property with Enumeration in AASX Package Explorer

Figure 18 shows the concept description for the value BAB657 that was used in the enumeration in Figure 17. Most attributes are not relevant (see Clause 8). However, it is mandatory to set the attribute *DataSpecificationIec61360/value*, the *preferredName* (open circuit, external cooling), and the data type (for enumeration, the data type is typically just STRING).



ConceptDescription

Referable:

idShort:	BAB657
category:	VALUE

HasExtension:

Identifiable:

id:	0173-1#07-BAB657#003
id (Base64):	MDE3My0xIzA3LUJBQjY1NyMwMDM=

HasDataSpecification (records of embedded data specification):

dataSpec.[0] / Reference:

dataSpec.[0]:	(GlobalReference) http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIEC61360/3/0

dataSpec.[0] / Content:

Data Specification Content IEC61360:

preferredName:	[en] open circuit, external cooling
dataType:	STRING
value:	BAB657

Figure 18 Example Value Concept Description in AASX Package Explorer

Referencing (Part 1)

Besides the abstract class *HasSemantics,* the referencing concept explained in Part 1 is also relevant (type *Reference*Figure 19). In the case of the data specification template IEC61360, the only relevant key types are "GlobalReference" and "ConceptDescription". In case the concept description is a shadow copy of an existing data dictionary and uses the same ID, it is recommended to use the Global Reference for the *DataSpecificationIec61360/unitId* or *ValueReferencePair/valueId*. Otherwise, a model reference with *Key/type* equal to *ConceptDescription* is used.

The same applies to *HasSemantics/semanticId* and semantic IDs in *HasSemantics/supplementalSemanticIds.*

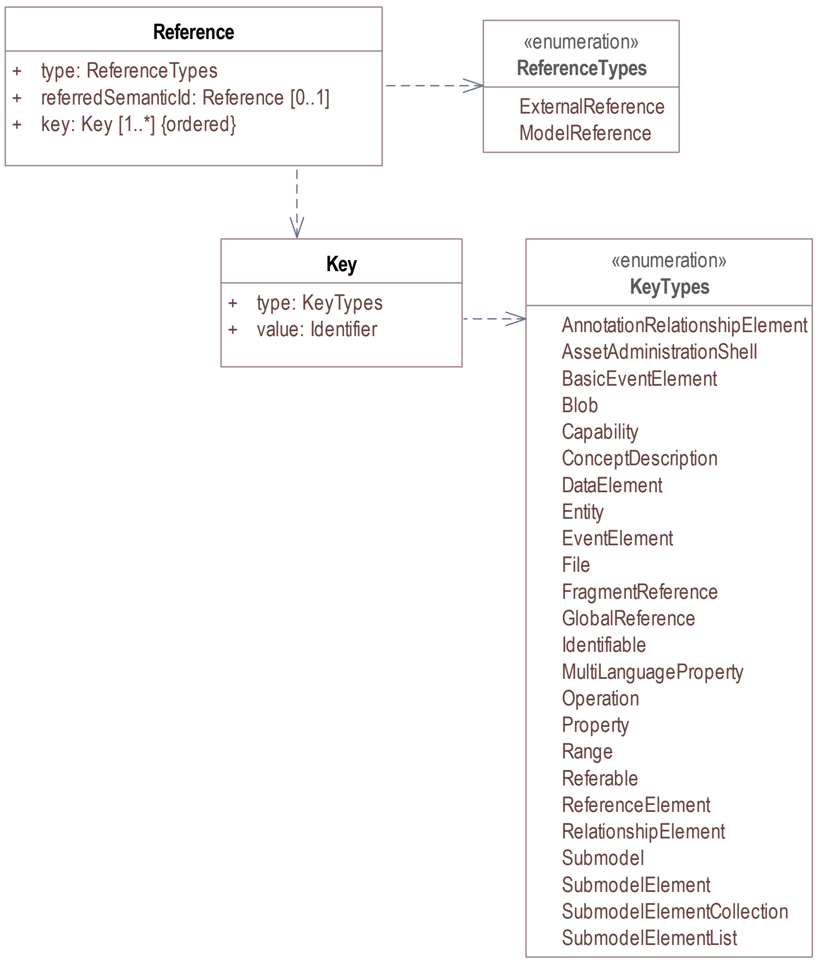


Figure 19 Metamodel of Reference (Part 1)

Annex B. Backus-Naur-Form

The Backus-Naur form (BNF) – a meta-syntax notation for context-free grammars – is used to define grammars. For more information see Wikipedia[13].

A BNF specification is a set of derivation rules, written as

```
<symbol> ::= __expression__
```

where:

- <symbol> is a nonterminal (variable) and the __expression__ consists of one or more sequences of either terminal or nonterminal symbols,
- ::= means that the symbol on the left must be replaced with the expression on the right,
- more sequences of symbols are separated by the vertical bar "|", indicating a choice, the whole being a possible substitution for the symbol on the left,
- symbols that never appear on a left side are terminals, while symbols that appear on a left side are non-terminals and are always enclosed between the pair of angle brackets <>,
- terminals are enclosed with quotation marks: "text". "" is an empty string,
- optional items are enclosed in square brackets: [<item-x>],
- items existing 0 or more times are enclosed in curly brackets are suffixed with an asterisk (*) such as <word> ::= <letter> {<letter>}*,
- Items existing 1 or more times are suffixed with an addition (plus) symbol, +, such as <word> ::= {<letter>}+,
- round brackets are used to explicitly to define the order of expansion to indicate precedence, example: (<symbol1> | <symbol2>) <symbol3>,
- text without quotation marks is an informal explanation of what is expected; this text is cursive if grammar is non-recursive and vice versa.

Example:

```
<contact-address> ::= <name> "e-mail addresses:" <e-mail-Addresses>
<e-mail-Addresses> ::= {<e-mail-Address>}*
<e-mail-Address> ::= <local-part> "@" <domain>
<name> ::= characters
<local-part> ::= characters conformant to local-part in RFC 5322
<domain> ::= characters conformant to domain in RFC 5322
```

Valid contact addresses:

```
Hugo Me e-mail addresses: Hugo@example.com

Hugo e-mail addresses: Hugo.Me@text.de
```

Invalid contact addresses:

```
Hugo

Hugo Hugo@ example.com

Hugo@example.com
```

[13] https://en.wikipedia.org/wiki/Backus%E2%80%93Naur_form

Annex C. Templates for UML Tables

General

The templates used for element specification are explained in this annex. For details for the semantics see Legend for UML Modelling.

For capitalization of titles, rules according to https://capitalizemytitle.com/ are used.

Template for Classes

Template for Classes:

Class:	<Class Name> [<<abstract>>] ["<<Experimental>>"] ["<<Deprecated>>"] ["<<Template>>"]		
Explanation:	<Explanatory text>		
Inherits from:	{<Class Name> ";" }+ | "-"		
Attribute	**Explanation**	**Type**	**Card.**
<attribute or association name> ["<<ordered>>"] ["<<Experimental>>"] ["<<Deprecated>>"]	<Explanatory text>	<Type>	<Card>

The following stereotypes can be used:

- <<abstract>>: Class cannot be instantiated but serves as superclass for inheriting classes
- <<Experimental>>: Class is experimental, i.e. usage is only recommended for experimental purposes because non backward compatible changes may occur in future versions
- <<Deprecated>>: Class is deprecated, i.e. it is recommended to not use the element any longer; it will be removed in a next major version of the model
- <<Template>>: Class is a template only, i.e. class is not instantiated but used for additional specification purposes (for details see parts 3 of document series)
- 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 is realized as composite aggregation (composition), and does not exist independent of its parent
- *Type:*
  - *<Class>:* Type is a class
  - *ModelReference<{Referable}>:* Type is a Reference with *Reference/type=ModelReference* and 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 D, for more information on referencing see Annex A
  - *<Primitive>: Type* is a primitive data type, see Clause 10
  - *<Enumeration>:* Type is an enumeration
- *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; there is always 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>", the following reference would be a valid reference (using the text serialization as defined in Part 1:

```
(Submodel)https://example.com/aas/1/1/1234859590
```

If class type equal to "ModelReference<Referable>", the following references would be valid references (using the text serialization as defined in part 1:

```
(Submodel)https://example.com/aas/1/1/1234859590
```

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

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

This would be an invalid reference for "ModelReference<Referable>", instead type "Reference" shall be used:

```
(Submodel)https://example.com/aas/1/1/1234859590, (File)myDocument
(FragmentReference)Hints
```

This would be an invalid reference for "ModelReference<Submodel>"

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

Template for Enumerations

Template for Enumerations:

Enumeration:	<Enumeration Name> ["<<Experimental>>"] ["<<Deprecated>>"]
Explanation:	
Set of:	{<Enumeration> ";" }+ | "-"
Literal	**Explanation**
<enumValue1>["<<Experimental>>"] ["<<Deprecated>>"]	<Explanatory text> Value of enumeration
<enumValue2> ["<<Experimental>>"] ["<<Deprecated>>"]	<Explanatory text> Value of enumeration, also included in one of the enumerations listed in "Set of:"

"Set Of" lists enumerations that are contained in the enumeration. It is only relevant for validation, making sure that all elements relevant for the enumeration are considered.

Enumeration values use Camel Case notation and start with a small letter. However, there might be exceptions in case of very well-known enumeration values.

Template for Primitives

Template for Primitive:

Primitive	Explanation	Value Examples
<Name of Primitive>	<Explanatory text>	Value examples

Handling of Constraints

Constraints are prefixed with **AASc-** followed by the number of the document in the Part 3 series (here "3a"), followed by a three-digit number. The "c" in "AASc-" was motivated by "Concept Description". The numbering of constraints is unique within the namespace AASc; a number of a constraint that was removed will not be used again.

Note: in the Annex listing the metamodel changes, constraints with prefix AASs- or AASc- are also listed. These are security or data specification constraints and are now part of the split document Legend for UML Modelling.

Annex D. Legend for UML Modelling

OMG UML General

This annex explains the UML elements used in this specification. For more information, please refer to the comprehensive literature available for UML. The formal specification can be found in [12].

Figure 20 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 21. In this case, the instance of *Class2* is navigable via the instance of the owning class *Class1*.[14]

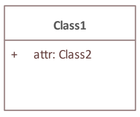


Figure 20 Class

Figure 21 shows that *Class4* inherits 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 *Class4* has the attributes *attr1* and *attr2,* whereas instances of *Class3* only have the attribute *attr1*.

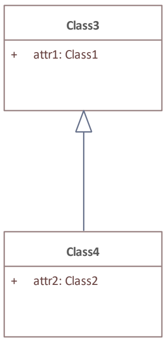


Figure 21 Inheritance/Generalization

Figure 22defines 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>

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.

[14] „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. [12]"

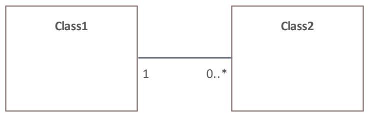


Figure 22 Multiplicity

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 and contain no duplicates. Figure 23 shows an ordered collection: the instances of *Class2* related to an instance of *Class1*. The stereotype <<ordered>> is used to denote that the relationship is ordered.

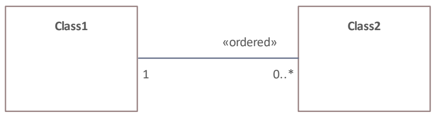


Figure 23 Ordered Multiplicity

Figure 24 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*.

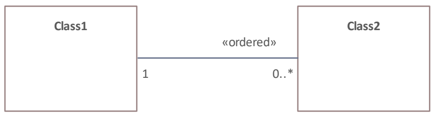


Figure 24 Association

Figure 25 shows a composition, also called a composite aggregation. A composition is a binary association, grouping 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*. Figure 26 shows the composition using an association relationship with a filled diamond as composition adornment.

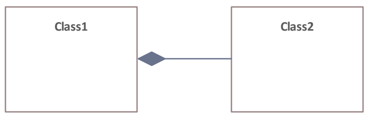


Figure 25 Composition (Composite Aggregation)

Figure 26 shows 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*. Figure 26 shows the shared aggregation using an association relationship with a hallow diamond as aggregation adornment.

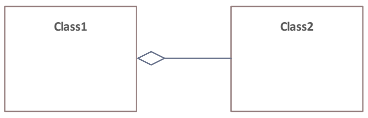


Figure 26 Aggregation

Figure 27 illustrates 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, it is added in curly brackets like in this example.

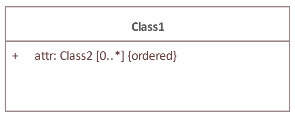


Figure 27 Navigable Attribute Notation for Associations

Figure 28 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 a specific setting for the property, the default value is evaluated to provide the initial values of the property.

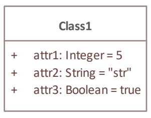


Figure 28 Default Value

Figure 29 shows that there is a dependency relationship between *Class1* and *Class2*. In this case, the dependency means that *Class1* depends on *Class2* because the type of attribute *attr* depends on the specification of class *Class2.* A dependency is depicted as dashed arrow between two model elements.

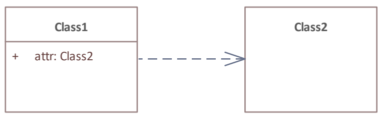


Figure 29 Dependency

Figure 30 shows an abstract class. It uses the stereotype <<abstract>>. There are no instances of abstract classes. They are typically used for specific member elements that are inherited by non-abstract classes.

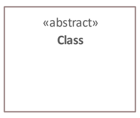


Figure 30 Abstract Class

Figure 31 shows a package named "Package2". A package is a namespace for its members. In this example, the member belonging to *Package2* is class *Class2*.

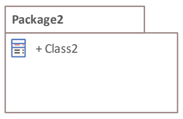


Figure 31 Package

Figure 32 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>>.

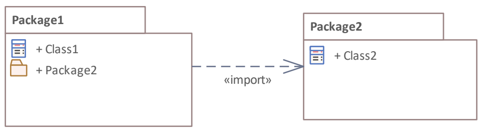


Figure 32 Imported Package

Figure *33* shows an enumeration with the name "Enumeration1". An enumeration is a data type with its values enumerated as literals. It contains two literal values, "a" and "b". It is a class with stereotype <<enumeration>>. The literals owned by the enumeration are ordered.

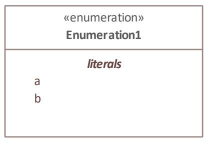


Figure 33 *Enumeration*[15]

Figure 34 shows the definition of the data type with the name "DataType1". A data type has instances that are identified only by their value. It is a class with stereotype <<dataType>>.

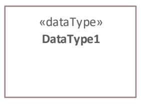


Figure 34 Data Type

Figure 35 shows a primitive data type with the name "int". Primitive data types are predefined data types, without any substructure. The primitive data types are defined outside UML.

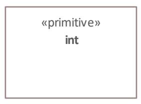


Figure 35 Primitive Data Type

[15] In Enterprise Architect, the single enumeration values also have a stereotype <<enum>> each.

Figure 36 shows how a note can be attached to an element, in this example to class "Class1".

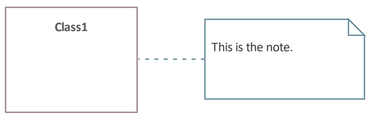


Figure 36 Note

Figure 37 shows how a constraint is attached to an element, in this example to class "Class1".

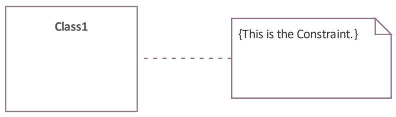


Figure 37 Constraint

UML Naming Rules

The following rules are used for naming of classes, attributes etc.:

- all names use CamelCase; for exceptions see rules for Enumeration values,
- class names always start with a capital letter,
- attribute names always start with a small letter,
- primitive types start with a capital letter; exception: predefined types of XSD like string,
- enumerations start with a capital letter,
- names of member ends of an association start with a capital letter,
- all stereotypes specific to the Asset Administration Shell specification start with a capital letter, e.g. "<<Deprecated>>"; predefined stereotypes in UML start with a small letter, e.g. "<<abstract>>" or "<<enumeration>>".

In UML, the convention is to name associations and aggregations in singular form. The multiplicity is to be taken into account to decide on whether there are none, a single, or several elements in the corresponding association or aggregation.

Note: a plural form of the name of attributes with cardinality >=1 may be needed in some serializations (e.g. in JSON). In this case, it is recommended to add an "s". In case of resulting incorrect English (e.g. isCaseOf ➔ isCaseOfs), it must be decided whether or not to support such exceptions.

Notes to Graphical UML Representation

Specific graphical modelling rules, which are used in this specification but not included in this form, are explained below [12].

Figure 38 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. This document uses notation B.

It is assumed that only the end member of the association is navigable per default, 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, it is assumed that the name is identical to the class name but starting with a small letter – compared to Variant C.

Class2 instance only exists if the parent object of type *Class1* exists.

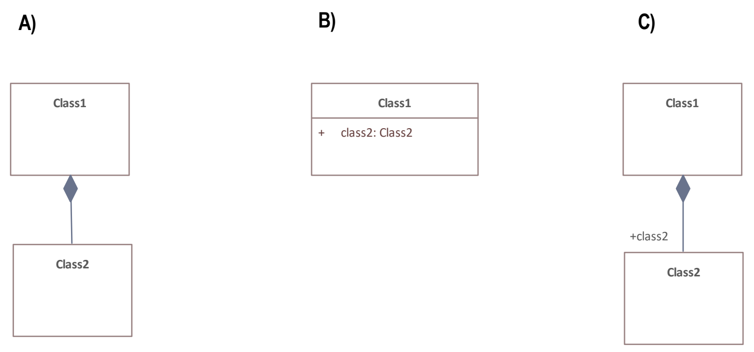


Figure 38 Graphical Representations of Composite Aggregation/Composition

Figure *39* shows different representations of a shared aggregation: a *Class2* instance can exist independently of a *Class1* instance; it only references the instances of *Class2*. Now an attribute with the same semantics is defined In Variant B. The reference is denoted by a star added after the type of the attribute.

It is assumed that only the end member of the aggregation association is navigable per default, 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 specialty in Figure *39* 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").

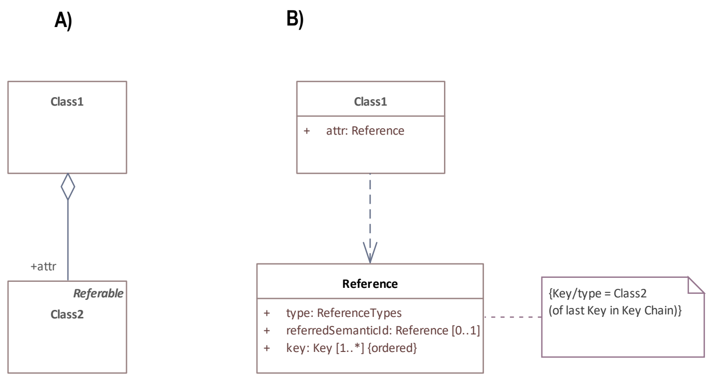


Figure 39 *Graphical Representation of Shared Aggregation*

Figure 40 show different graphical representations of generalization. Variant A is the classical graphical representation as defined in [12]. Variant B is a short form, if *Class1* is not on the same diagram. The name of the class that *Class3* is inheriting from is depicted in the upper right corner.

Variant C not only shows which class Class3 instances are inheriting from, 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.

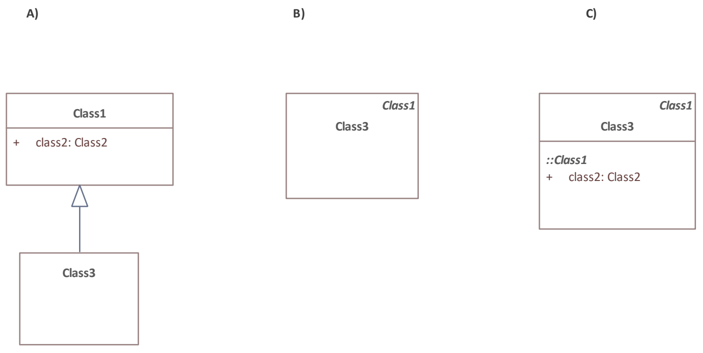


Figure 40 Graphical Representation of Generalization/Inheritance

Figure 41 depicts different graphical notations for enumerations in combination with inheritance. In Variant A, "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 visualizes which literals are inherited by which enumeration. Since the literals within an enumeration are ordered, the order of classes it is inheriting from is important.

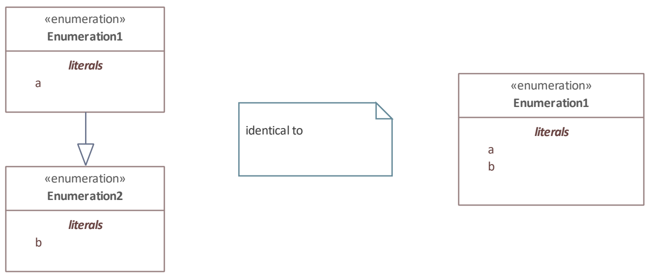


Figure 41 Graphical Representation for Enumeration with Inheritance

Figure 42 shows an experimental class, marked by the stereotype "Experimental".

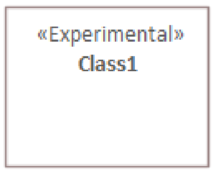


Figure 42 Graphical Representation for Experimental Classes

Figure 43 depicts a deprecated class, which is marked by the stereotype "Deprecated".

Figure 44 shows a class representing a template. It is marked by the stereotype "Template".

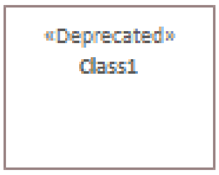


Figure 43 Graphical Representation for Deprecated Elements

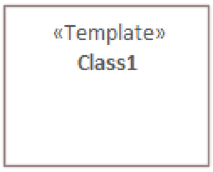


Figure 44 Graphical Representation of a Template Class

Annex E. Metamodel Changes

General

This Annex lists the changes from version to version of the metamodel together with major changes in the overall document. Non-backward compatible changes (nc) are marked as such.

nc="x" means not backward compatible, if no value is added in the table, the change is backward compatible.

nc="(x)" means that the change made was implicitly contained or stated in the document before and is now being formalized. Therefore, the change is considered to be backward compatible.

Changes for the data specification templates of the metamodel are listed in separate tables each.

Each subclause consists of three parts:

1. changes w.r.t. previous version,
2. new elements in metamodel w.r..t previous version,
3. new, changed, or removed Constraints w.r.t previous version.

Note: before V3.0RC03, the security metamodel was also part of this document. Therefore, security metamodel changes were also listed using the three subclauses as described above.

Changes V3.0 vs. Part 1 V2.0.1

Major Changes:

- CHANGE: was part of part 1 in former versions of the document series until V3.0RC02
- CHANGE: string types replaced by explicit types with length restrictions, etc.
- CHANGE: id of data specification IEC62360 changed (camel case)
- NEW: additional IEC 61360 data types: IRI, IRDI, HTML, FILE, BLOB
- EDITORIAL: mapping to IEC 61360 notes added
- NEW: new terms added to Clause "Terms, Definitions and Abbreviations" (maximum value, minimum value, nominal value, non-quantitative property, quantitative property)
- NEW: Clause "Normative References" in Preamble
- NEW: SpecificAssetId added to table with categories of concept descriptions
- NEW: constraints added for applying categories to concept descriptions
- UPDATE: data mappings IEC 61360 to xsd data types as used in part 1
- CHANGE: no IEC 61360 data type RATIONAL_* allowed any longer for RANGE; instead, INTEGER_* is used
- CHANGE: all IEC 61360 data types allowed for Property, except STRING_TRANSLATABLE, IRI, IRDI, HTML, FILE, BLOB (before only STRING_TRANSLATABLE was excluded)
- CHANGE: LevelType changed from Enumeration to Class, Table added
- CHANGE: Names containing IEC renamed to camel case using Iec, e.g. DataSpecificationIEC61360

nc	V3.0 vs. Part 1 V2.0.1	Comment
x	DataSpecificationIEC61360	Renamed to DataSpecificationIec61360
	DataSpecificationContent	Stereotype <<Template>> added
x	DataTypeIEC61360	Renamed to DataTypeIec61360 Some new values added: BLOB, FILE, HTML, IRDI; URL renamed to IRI

nc	V3.0 vs. Part 1 V2.0.1	Comment
x	DataSpecificationIec61360/valueId	Removed, the valueId is identical to the ID of the concept description
x	LevelType	Changed from enumeration to complex data type with four Boolean attributes because more than one value can be selected
x	ValueList/valueReferencePairs	Bugfix, was ValueList/valueReferencePairTypes before
x	ValueReferencePair/value	Type changed from ValueDataType to string

Table 7 Changes

nc	V3.0 vs. Part 1 V2.0.1 New Elements	Comment
x	DataTypeIec61360	Renamed, before: DataTypeIEC61360 Values remain, some new values added, see separate entries
	DataTypeIec61360/BLOB	New value, compared to DataTypeIEC61360
	DataTypeIec61360/FILE	New value, compared to DataTypeIEC61360
	DataTypeIec61360/HTML	New value, compared to DataTypeIEC61360
	DataTypeIec61360/IRDI	New value, compared to DataTypeIEC61360
x	DataTypeIec61360/IRI	Renamed, before URL in DataTypeIEC61360
x	DataSpecificationIec61360	Renamed, before: DataSpecificationIEC61360 Some attribute types changed, see separate entries
x	DataSpecificationIec61360/definition	Type changed from LangStringSet to DefinitionTypeIec61360 compared to DataSpecificationIEC61360/definition
x	DataSpecificationIec61360/levelType	Type changed from enumeration to complex type (name stayed LevelType) compared to DataSpecificationIEC61360/levelType
x	DataSpecificationIec61360/preferredName	Type changed from LangStringSet to PreferredNameTypeIec61360 with limited max. length compared to DataSpecificationIEC61360/preferredName
x	DataSpecificationIec61360/shortName	Type changed from LangStringSet to ShortNameTypeIec61360 with limited max. length compared to DataSpecificationIEC61360/shortName
x	DataSpecificationIec61360/value	Type changed from ValueDataType to ValueTypeIec61360

nc	V3.0 vs. Part 1 V2.0.1 New Elements	Comment
x	DataSpecificationIec61360/valueFormat	Type changed from string to ValueFormatTypeIec61360 compared to DataSpecificationIEC61360/valueFormat
	ValueTypeIec61360	New type for values

Table 8 New Elements in Metamodel

Nc	V3.0 vs. Part 1 V2.0.1	New, Update, Removed, Reformulated	Comment
	AASc-3a-002	New	Updated version of AASd-076, renamed to AASc-3a-002 because applicable to data specification IEC61360 Constraint AASc-3a-002: DataSpecificationIec61360/preferredName shall be provided at least in English.
(x)	AASc-3a-003	New	Constraint AASc-3a-003: For a *ConceptDescription* referenced via *ValueList/valueId* and using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), *DataSpecificationIec61360/value* shall be set.
(x)	AASc-3a-004	New	Constraint AASc-004: For a ConceptDescription with category PROPERTY or VALUE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), DataSpecificationIec61360/dataType is mandatory and shall be defined.
(x)	AASc-3a-005	New	Constraint AASc-005: For a ConceptDescription with category REFERENCE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), DataSpecificationIec61360/dataType is STRING by default.
(x)	AASc-3a-006	New	Constraint AASc-006: For a ConceptDescription with category DOCUMENT using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), DataSpecificationIec61360/dataType shall be one of the following values: STRING or URL.
(x)	AASc-3a-007	New	Constraint AASc-007: For a ConceptDescription with category QUALIFIER_TYPE using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), DataSpecificationIec61360/dataType is mandatory and shall be defined.
(x)	AASc-3a-008	New	Constraint AASc-3a-008: For a ConceptDescription using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), DataSpecificationIec61360/definition is mandatory and shall be defined at least in English. Exception: the concept description describes a value, i.e. DataSpecificationIec61360/value is defined.
(x)	AASc-3a-009	New	Constraint AASc-009: If DataSpecificationIec61360/dataType is one of INTEGER_MEASURE, REAL_MEASURE, RATIONAL_MEASURE, INTEGER_CURRENCY, REAL_CURRENCY, then

Nc	**V3.0 vs. Part 1 V2.0.1**	**New, Update, Removed, Reformulated**	**Comment**
			DataSpecificationIec61360/unit or DataSpecificationIec61360/unitId shall be defined.
(x)	AASc-3a-010	New	Constraint AASc-010: If DataSpecificationIec61360/value is not empty, DataSpecificationIec61360/valueList shall be empty, and vice versa
	AASc-3a-050	New	Constraint AASc-050: If the DataSpecificationContent DataSpecificationIec61360 is used for an element, 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

Table 9 New, Changed or Removed Constraints

Changes V3.0 vs. Part 1 V3.0RC02

Major Changes:

- CHANGE: was part of Part 1 in former versions of the document series until V3.0RC02
- CHANGE: string types replaced by explicit types with length restrictions, etc.
- CHANGE: id of data specification IEC62360 changed (camel case)
- EDITORIAL: mapping to IEC 61360 notes added
- NEW: new terms added to Clause "Terms, Definitions and Abbreviations" (maximum value, minimum value, nominal value, non-quantitative property, quantitative property)
- NEW: Clause "Normative References" in Preamble
- NEW: SpecificAssetId added to table with categories of concept descriptions
- UPDATE: data mappings IEC 61360 to xsd data types as used in part 1
- CHANGE: no IEC 61360 data type RATIONAL_* allowed any longer for RANGE

Bugfixes:

- LevelType changed from Enumeration to Class, Table added
- IEC 61360 Data Specification Template for Properties and Ranges: footnote corrected, data types like Iso29002Irdi and Icid are subsumed in IRDI, no camel case writing but capital letters and underscore
- Renaming constraints relevant for concept descriptions from AASd- to AASc-

nc	**V3.0 vs. Part 1 V3.0RC02**	**Comment**
x	DataSpecificationIEC61360	Renamed to DataSpecificationIec61360
x	DataTypeIEC61360	Renamed to DataTypeIec61360
x	ValueReferencePair/value	Type changed from string to ShortNameTypeIec61360 with limited max. length

Table 10 Changes

nc	V3.0RC01 vs. Part 1 V3.0RC02 New Elements	Comment
x	DataSpecificationIec61360	Renamed, before: DataSpecificationIEC61360
x	DataSpecificationIec61360/definition	Type changed from MultiLanguageSet to DefinitionTypeIec61360 compared to DataSpecificationIEC61360/definition
x	DataSpecificationIec61360/levelType	Type changed from enumeration to complex type (name stayed LevelType) compared to DataSpecificationIEC61360/levelType
x	DataSpecificationIec61360/preferredName	Type changed from MultiLanguageSet to PreferredNameTypeIec61360 with limited max. length compared to DataSpecificationIEC61360/preferredName
x	DataSpecificationIec61360/shortName	DataSpecificationIEC61360/shortName
x	DataSpecificationIec61360/value	Type changed from ValueDataType to ValueTypeIec61360
x	DataSpecificationIec61360/valueFormat	Type changed from string to ValueFormatTypeIec61360 compared to DataSpecificationIEC61360/valueFormat
x	DataTypeIec61360	Renamed, before: DataTypeIEC61360
x	LevelType	Changed from enumeration to complex data type with four Boolean attributes because more than one value can be selected

Table 11 New Elements in Metamodel

Nc	V3.0 vs. Part 1 V3.0RC02	New, Update, Removed, Reformulated	Comment
	AASd-050	Removed	Renamed from AASd-050 to AASc-3a-050, see new AASc-3a-050 + update renamed elements
	AASc-002	Removed	Renamed from AASc-002 to AASc-3a-002 + update renamed elements
	AASc-003	Removed	Renamed from AASc-003 to AASc-3a-003 + update renamed elements
	AASc-004	Removed	Renamed from AASc-004 to AASc-3a-004 + update renamed elements
	AASc-005	Removed	Renamed from AASc-005 to AASc-3a-005 + update renamed elements
	AASc-006	Removed	Renamed from AASc-006 to AASc-3a-006 + update renamed elements
	AASc-007	Removed	Renamed from AASc-007 to AASc-3a-007 + update renamed elements
	AASc-008	Removed	Renamed from AASc-008 to AASc-3a-008 + update renamed elements
	AASc-009	Removed	Renamed from AASc-009 to AASc-3a-009 + update renamed elements
	AASc-010	Removed	Renamed from AASc-010 to AASc-3a-010 + update renamed elements
	AASc-3a-002	New	Renamed from AASc-002 to AASc-3a-002 + update renamed elements

Nc	V3.0 vs. Part 1 V3.0RC02	New, Update, Removed, Reformulated	Comment
	AASc-3a-003	New	Renamed from AASc-003 to AASc-3a-003 and changed to no longer contain category Constraint AASc-3a-003: For a *ConceptDescription* referenced via *ValueList/valueId* and using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), *DataSpecificationIEC61360/value* shall be set.
	AASc-004	New	Renamed from AASc-004 to AASc-3a-004, + update renamed elements + editorial changes
	AASc-005	New	Renamed from AASc-005 to AASc-3a-005, + update renamed elements + editorial changes
	AASc-006	New	Renamed from AASc-006 to AASc-3a-006, + update renamed elements + editorial changes
	AASc-007	New	Renamed from AASc-007 to AASc-3a-007, + update renamed elements + editorial changes
	AASc-3a-008	New	Renamed from AASc-008 to AASc-3a-008 and changed to no longer contain category Constraint AASc-3a-008: For a *ConceptDescription* using data specification template IEC61360 (http://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3/0), *DataSpecificationIec61360/definition* is mandatory and shall be defined at least in English. Exception: the concept description describes a value, i.e. *DataSpecificationIec61360/value* is defined.
	AASc-009	New	Renamed from AASc-009 to AASc-3a-009, + update renamed elements + editorial changes
	AASc-010	New	Renamed from AASc-010 to AASc-3a-010, + update renamed elements + editorial changes
	AASc-3a-050	New	Renamed from AASd-050 to AASc-3a-050 + update renamed elements + version updated Constraint AASc-3a-050: If the *DataSpecificationContent DataSpecificationIec61360* is used for an element, 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*

Table 12 New, Changed or Removed Constraints

Annex F. 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 reference 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] "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

[6 DIN SPEC 92000:2019-09 "Data Exchange on the Base of Property Value Statements (PVSX)", 2019 September.

[7] "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

[8] "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

[9] "Industrial automation systems and integration — Exchange of characteristic data — Part 10: Characteristic data exchange format", Technical Specification ISO/TS 29002-10:2009(E), 2009

[10] "Smart Manufacturing - Reference Architecture Model Industry 4.0 (RAMI4.0)", IEC PAS 63088, International Electrotechnical Commission (IEC), 2017

[11] 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

[12] "OMG Unified Modelling Language (OMG UML)". Formal/2017-12-05. Version 2.5.1. December 2018. [Online] Available: https/www.omg.org/spec/UML/

[13] T. Preston-Werner "Semantic Versioning". Version 2.0.0. Accessed: 2020-11-13. [Online] Available: https://semver.org/spec/v2.0.0.html

[14] "Details of the Asset Administration Shell – Interoperability at Runtime – Exchanging Information via Application Programming Interfaces". Part 2. See [22].

[15] "Asset Administration Shell. Reading Guide". Plattform Industrie 4.0 in cooperation with IDTA. November 2020. See [22].

[16] Top Level Project "Eclipse Digital Twin" Available: https://projects.eclipse.org/projects/dt

[17] OPC 30270: OPC UA for Asset Administrastion Shell (AAS). 2021-06-04. [Online]. Available: https://reference.opcfoundation.org/v104/I4AAS/v100/docs/

[18] OPC Unified Architecture Specification. Part 5 Information Model. [Online]. Available: https://opcfoundation.org/developer-tools/specifications-unified-architecture

[19] OPC UA Information Models. [Online]. Available: https://opcfoundation.org/developer-tools/specifications-opc-ua-information-models

[20] IEC 63278-1 "Asset Administration Shell for industrial applications – Part 1: Asset Administration Shell structure". 95/925/CDV

[21] "Registered AAS Submodel Templates". Industrial Digital Twin Association. Available: https://industrialdigitaltwin.org/en/content-hub/submodels

[22] "Asset Administration Shell Specifications – Quicklinks to Different Versions & Reading Guide". [Online]. Available: https://www.plattform-i40.de/IP/Redaktion/EN/Standardartikel/specification-administrationshell.html

[23] (German) "I4.0-Sprache. Vokabular, Nachrichtenstruktur und semantische Interaktionsprotokolle der I4.0-Sprache", Discussion Paper. Plattform Industrie 4.0 [Online] Available: https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/hm-2018-sprache.html

[24] "How to create a submodel template specification". Dec. 2022. Industrial Digital Twin Association. Available: https://industrialdigitaltwin.org/wp-content/uploads/2022/12/I40-IDTA-WS-Process-How-to-write-a-SMT-FINAL-.pdf

[25] "AAS Repository. Repository for Information and Code for the Asset Administration Shell". https://github.com/admin-shell-io

[26] F. Manola, E. Miller "RDF 1.1 Primer" W3C Recommendation, 2014, [Online]. Available: https://www.w3.org/TR/rdf11-primer/

[27] Modelling the Semantics of Data of an Asset Administration Shell with Elements of ECLASS. June 2021. [Online]. Available: https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/Whitepaper_Plattform-Eclass.pdf

www.industrialdigitaltwin.org