Specification of the Asset Administration Shell

Part 3b: Data Specification – Units of Measurement

SPECIFICATION

IDTA Number: 01003-b
Version 3.0

Specification of the Asset Administration Shell

This specification is part of the Asset Administration Shell Specification series.

Version

This is the version 3.0 of the specification IDTA-01003-b. This version 3.0 is the initial version of the document.

Metamodel Versions

This document uses the following parts of the “Specification of the Asset Administration Shell” series:

- IDTA-01001 Part 1: Metamodel in version 3.2 [22]
- IDTA-01001 Part 3a: Data Specification - IEC61360 in version 3.1 [22]

Notice

IDTA Document Number: IDTA-01003-b-3-0

DOI: https://doi.org/10.62628/idta.01003-b-3.0

July 2026

How to Get in Contact

Contact: IDTA

Sources: GitHub

Feedback:

- new issues, bugs
- Questions and Answers

Editorial Notes

History

This document, version 3.0, is the initial version.

Versioning

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

Conformance

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]:

- MUST word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the

specification.

- MUST NOT This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
- 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.
- 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.
- 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.)

Terms and Definitions

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.

If available, definitions were taken from IEC 63278-1 Edition 1.0 2023-12 or IEC 61360-1 Edition 4.0 2017-07.

attribute

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

[SOURCE: IEC 63278-1:2023; 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 63278-1:2023, 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]

coded value

value that can be looked up in a dictionary and can be translated

[SOURCE: ECLASS[2]]

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]

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 to Glossary Industrie 4.0]

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

representation of an aspect of an *asset*

[SOURCE: IEC 63278-1:2023]

SubmodelElement

element of a *Submodel*

[SOURCE: IEC 63278-1:2023]

Abbreviations Used in Document

Abbreviation	Description
AAS	Asset Administration Shell
AASX	Package file format for the Asset Administration Shell

Abbreviation	Description
API	Application Programming Interface
BLOB	Binary Large Object
CDD	IEC Common Data Dictionary
GUID	Globally unique identifier
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
NIST	National Institute of standards and Technology
PDF	Portable Document Format
RDF	Resource Description Framework
RFC	Request for Comment
SI	Système international d’unités
UML	Unified Modelling Language
UoM	Unit of Measurement
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

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

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

[2] In IEC61360:2017, this refers to a "term" of a value list

Preamble

Preamble

Scope of This Document

The Asset Administration Shell (see Part 1 of the document series, IDTA-01001) 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 for modelling Units of Measurement (UoM). In this document the UoM addresses the concepts of UNITS and their relationship to QUANTITIES. In this version of the document a very simple model is provided that can be extended in future versions. The focus lies in providing a minimal set of attributes to describe a Unit of Measurement and reference to external dictionaries for further information. The definition of new UoM is currently not in scope.

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.

Normative References

[IDTA-01001] "Specification of the Asset Administration Shell. Part 1: Metamodel", IDTA-01001-3-1. Industrial Digital Twin Association.

[IDTA-01003-a] "Specification of the Asset Administration Shell Part 3a: Data Specification - IEC 61360", IDTA-01003-a-3-1. Industrial Digital Twin Association.

> Note: The semantic identifiers for the classes and attributes etc. defined in this document are derived conformant to the grammar for semantic IDs for data specifications as defined in Part 1 of the document series, IDTA-01001

Structure of the Document

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

Terms and Definitions 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.

Introduction gives a short introduction of Asset Administration Shell types and how this document is related to them.

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

Predefined Data Specification Templates shows how the data specification template is related to Part 1 and its elements.

Specification (normative) is the main normative part of the document. It specifies the data specification template for Units of Measurement.

Primitive and Simple Data Types (normative) specifies the data types used in the data specification.

Mappings to Data Formats to Share I4.0-Compliant Information (normative) 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, *Summary and Outlook* summarizes the content and gives an outlook on future work.

The Annex contains an extensive list of modelling examples for the most common dictionaries for UoM (Examples). It also provides information about UML (UML) and the tables used to specify UML classes as used in this specification (UML Table Templates). Backus Naur Form introduces the Backus-Naur-Form used in the document series.

Metamodel changes compared to previous versions are described in *Change Log*.

The bibliography can be found in *Bibliography*.

Introduction

Introduction

This document is part of the series "Specification of the Asset Administration Shell" that provide the specifications for interoperable usage of the Asset Administration Shell. This part defines a technology-neutral specification of a data specification template, enabling the description of concept descriptions for Units of Measurement. This UML metamodel serves as the basis for deriving several different formats for exchanging Asset Administration Shells, e.g. for XML, JSON, RDF as described in Part 1 of the document series (IDTA-01001). Data Specification Templates are implemented using the embedded data specification approach. This means, that the implementation is part of the overall schemas as defined for IDTA-01001.

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

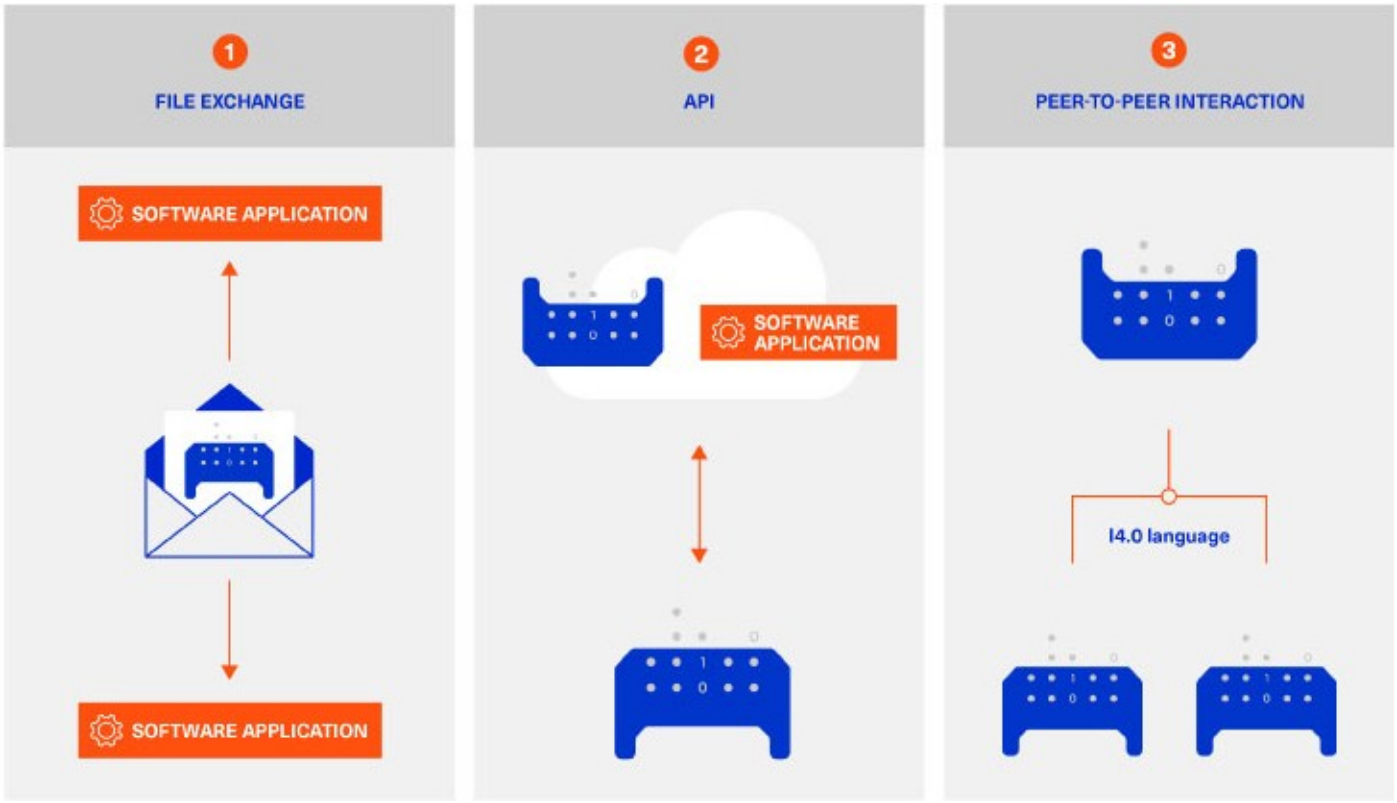


Figure 1. Types of Information Exchange via Asset Administration Shells

File exchange (1) is described in Part 5 of this document series (IDTA-01005).

The API (2) is specified in Part 2 of the document series "Specification of the Asset Administration Shell" (IDTA-01002) [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.

General

Assets are described by characteristics according to which assets can be compared. Characteristics that make assets comparable by their magnitude are called quantity, e.g. a length or a temperature. A Unit of Measurement (UoM) defines the magnitude of a quantity, e.g. metre is a unit of the quantity length with a defined magnitude. Referencing a Unit of Measurement allows to state the magnitude of a charasteristic in comparison to that unit, e.g. "The cube has the length of 3 metres.". Not all quantities can be compared even if they share the same Unit of Measurement. E.g. torque and energy both have the physical unit newton-meters (Nm), but they belong to different quantities that cannot be compared. For that reason, in definitions units and quantities should always be given as a pair [100], [101], [102].

Note: Countable quantities are a special kind of quantity whose magnitude is given in discrete values, e.g. "number of pages of the document".

The data specification template specified in this document allows to use Units of Measurement by providing a minimal set of attributes conformant to the relevant standards, especially IEC 61360 and ISO/IEC 80000.

Different use cases for modelling Units of Measurement have to be distinguished. These use cases build up on one another:

- Use Case 1: Stating that a value has a specific UoM
- Use Case 2: Stating that a value has a specific UoM and providing information for interpreting that UoM (focus of this document)
- Use Case 3: Stating that a value has a user-defined UoM and providing a complete definition of that unit

Use Case 1 can be addressed using Part 3a: Data Specification - IEC61360 (IDTA-01003-a) [22]. This part provides the two attributes that allow to state a UoM for SubmodelElements with a numerical value, i.e. *Property* and *Range*: *DataSpecificationIec61360/unitId* and *DataSpecificationIec61360/unit*. The Data Specification Units of Measurement is not needed for this use case.

Use Case 2 is the topic of the current version of this document. In addition to Use Case 1, the aim is to provide a minimal set of additional attributes to describe a Unit of Measurement and reference to external dictionaries for further information.

Use Case 3 is currently out of scope, but might be addressed in future versions of this document.

Related Standards

Multiple data dictionaries that define units and quantities exist with varying size and scope, e.g. ECLASS, IEC CDD or UNECE Recommendation 20. The lists of units ranges from basic SI units over vertical specific units to non-physical units such as “piece”. The attributes for describing UoM defined in this document have been chosen to be as generic as possible, to allow the usage of any of these dictionaries.

In the following some examples for existing data dictionaries are given.

- The IEC maintains a collection of more than 2,100 units and 350 quantities in the IEC CDD, which complies with the IEC TS 62720 standard.[1] ECLASS and IEC CDD are continuously working on harmonizing their dictionaries, including the UoM. To assure interoperability when using units these dictionaries relate physical units to their SI unit and provide the respective conversion factors. (However, this is not yet available for all physical units).
- UNECE Rec. 20 focuses on providing three character alphabetic and alphanumeric codes for representing Units of Measurement used in international trade.[2],[3] It is widely used, but in some apsects outdated and inconsistent. UNECE Rec. 21 does not list units for continuous quantities, but rather discrete units used in logistics such as *package*, *pallet* or *item*.
- The Bureau International des Poids et Mesures (BIPM) is the international organization through which Member States work together on matters related to metrology. As the home of the International System of Units (SI) it also lists a set of SI-related units and quantities as part of the SI digital framework.[4]. This dictionary is still work inprogress.
- While not being a data dictionary themselves, some existing meta models also provide templates to model units. OPC UA is maintained by the OPC Foundation does not define a unit data dictionary, but provides generic means to model units and quantities.[5],[6]. The Semantic Aspect Meta Model (SAMM) provides a catalog of units based on UNCE Rec. 20[7] in machine readable form and with unique identifiers. SAMM additionally provides the means to define new units of measurement.

Predefined Data Specification Templates

Overview

A data specification template specifies which additional attributes shall be added to an element instance that are not part of the metamodel. 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*.

This specification template *DataSpecificationUnitOfMeasurement* supports the modeling of Units of Measurement for submodel elements with values.

Figure 2 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*.

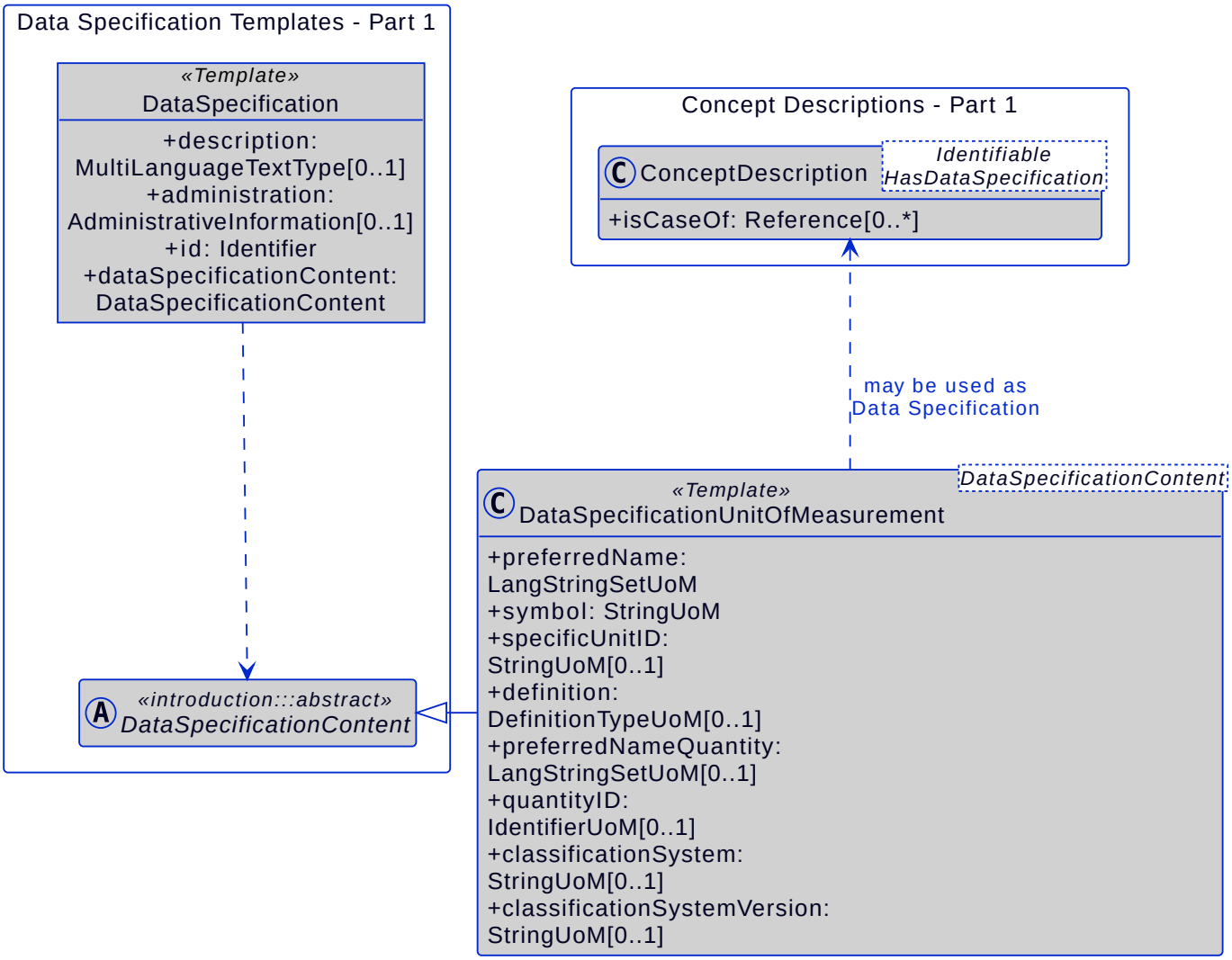


Figure 2. Overview Relationship Metamodel Part 1 & Data Specification Unit of Measurement

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 UML Table Templates and UML.

Using Data Specification Unit of Measurement

Using the template *DataSpecificationIec61360* as specified in IDTA-01003-a is a prerequisite for using the template *DataSpecificationUnitOfMeasurement* specified in this document. *DataSpecificationIec61360* provides the attributes *DataSpecificationIec61360/unitId* and *DataSpecificationIec61360/unit*. Thereby, *unit* is a string with the textual

representation of the unit of the value of the SubmodelElement. *unitId* is a reference with a matching unique ID of that unit.

When using the Data Specification Unit of Measurement the Reference *DataSpecificationIec61360/unitId* does not reference an external data dictionary. Instead it references a *ConceptDescription* via its *ConceptDescription/id*. The referenced *ConceptDescription* shall have the *DataSpecificationContent DataSpecificationUnitOfMeasurement*. This relation is illustrated exemplary in Figure 3.

In that example a property with the *idShort* "heightOfController7411" has been modelled. This property has the *value* of "54,4" and its semantics is detailled by the ECLASS data dictionary entry "0173-1#02-BAA020#011". In this example a *ConceptDescription* is supplied which contains this semantic information. The *semanticId* references the appropriate *ConceptDescription* with the *DataSpecificationContent DataSpecificationIec61360*. Here it is defined that the property is a height and the property's *value* is given in the *unit* "mm". Analogously, the *unitId* references the *ConceptDescription* with the *DataSpecificationContent DataSpecificationUnitOfMeasurement*. Here the definition of the unit and its quantity is given. In this example the definition is supplied by the ECLASS data dictionary entry "0173-1#05-AAA480#003".

Property	
idShort	"heightOfController7411"
description	"Height of Controller"
semanticId	type=ExternalReference, key/type=GlobalReference, key/value=0173-1#02-BAA020#011
valueType	xs:real
value	54,4

ConceptDescription with DataSpecificationIec61360	
id	"0173-1#02-BAA020#011"
idShort	"height"
dataSpecification	https://admin-shell.io/DataSpecificationTemplates/DataSpecificationIec61360/3
preferredName	"height"
shortName	"height"
unit	mm
unitId	type=ExternalReference, key/type=GlobalReference, key/value=0173-1#05-AAA480#003
symbol	"h"
dataType	REAL_MEASURE
definition	"with objects with preferred position of use, the dimension which is generally measured oriented to gravity and generally measured perpendicular to the supporting surface"

ConceptDescription with DataSpecificationUnitOfMeasurement	
id	"0173-1#05-AAA480#003"
idShort	"millimeter"
dataSpecification	https://admin-shell.io/DataSpecificationTemplates/DataSpecificationUnitOfMeasurement/3
preferredName	"millimeter"@en
symbol	"mm"
specificUnitID	"0173-1#05-AAA480#003"
definition	"0,001 fold of the SI base unit metre"@en
preferredNameQuantity	"distance"@en
quantityID	"0173-1#Z4-BAJ199#003"
classificationSystem	"ECLASS"
classificationSystemVersion	"14.0"

Figure 3. Exemplary usage of Data Specification Unit of Measurement

General Remarks for Working with Units of Measurement

This subsection states some general remarks for working with Units of Measurement that have proven to be helpful when kept in mind.

- If only a *semanticId* is provided for a quantitative or measurable property but no *ConceptDescription* with *DataSpecification61360* then the primary/default unit according to the respective data dictionary shall be used.
- If the *ConceptDescription* is provided for a quantitative or measurable property a *unit* or a *unitId* shall be supplied in *DataSpecification61360* (IDTA-01003-a).
- If the the respective data dictionary does not define a primary or default unit for a quantitative or measurable property, then a *ConceptDescription* shall be supplied that defines *unit* or *unitId*.
- If a *unit* or *unitId* is supplied that differs from the original property's data dictionary entry, then the entry in the *ConceptDescription* takes precendence.

- Using a unit that differs from the primary or default unit of the property's data dictionary entry does not break the semantic of that property. The whole purpose of a stringent semantic UoM data dictionary is to create interoperability by enabling applications to convert the *value* of properties from one UoM to another.
- If needed, it is possible to model multiple properties that have the identical data dictionary entry with different UoM. However, because Concept Descriptions are unique (identifiable) this modelling task is quite complicated and therefore not recommended. Figure Figure 4 illustrates that the change of the unit does not break the semantic of the property, but it does break the addressability of the Concept Description. As the *id* of the Concept Description must be unique for different contents, the original *semanticId* can only be transported using the *isCaseOf* attribute.

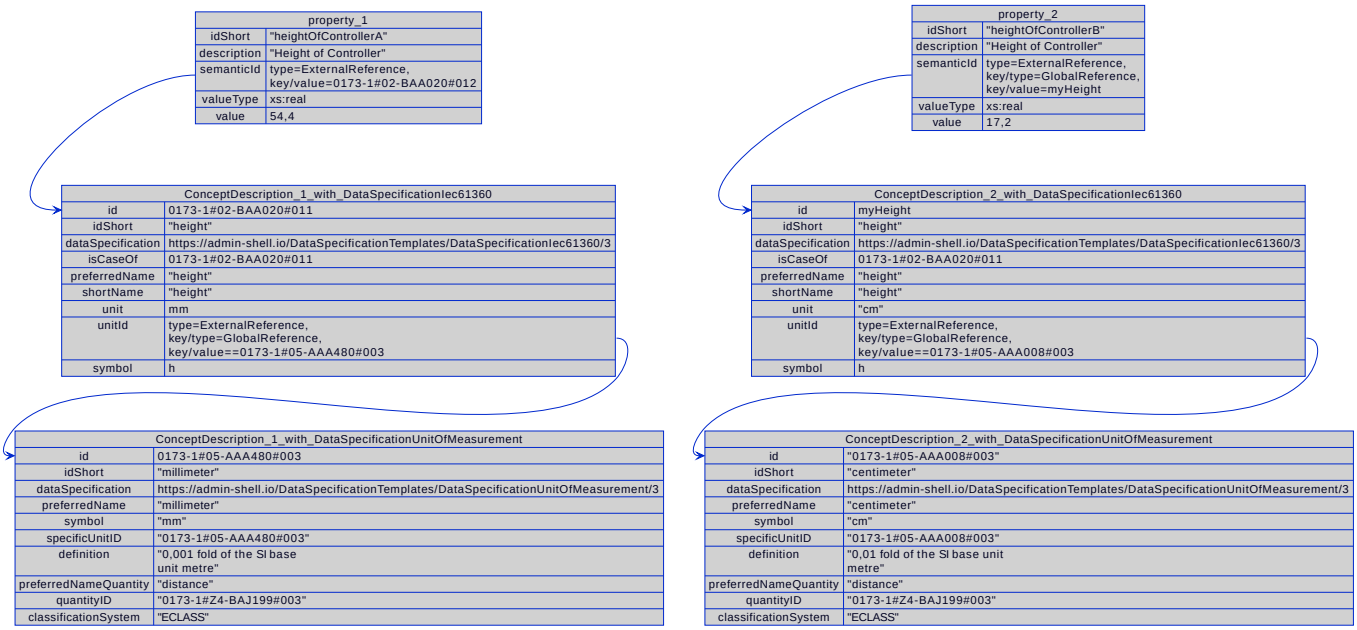


Figure 4. Illustration of modelling two properties with identical semantics, but different units

[1] https://cdd.iec.ch/

[2] https://unece.org/trade/uncefact/cl-recommendations

[3] https://service.unece.org/trade/uncefact/vocabulary/rec20/

[4] https://si-digital-framework.org

[5] https://reference.opcfoundation.org/Core/Part8/v105/docs/5.6.3

[6] https://reference.opcfoundation.org/Core/Part8/v105/docs/6

[7] https://eclipse-esmf.github.io/samm-specification/snapshot/units.html

Specification

Predefined Template for UoM for submodel elements with values (normative)

Data Specification Unit of Measurement Template Specification

This specification is only valid in combination with IDTA-01001-3 and IDTA-01003-a-3.

Template:	Unit of Measurement		
administration:	version: 3	revision: 0	creator: IDTA
id:	https://admin-shell.io/DataSpecificationTemplates/DataSpecificationUoM/3		
dataSpecificationContent:	DataSpecificationUoM		
Description (EN):	Data specification template for concept descriptions for Units of Measurement.		

The ID of the data specification template was derived conformant to the grammar for semantic IDs for data specifications as defined in Part 1 of the document series, IDTA-01001, with the exception that the minor version was excluded because this ID is also part of the instances of Asset Administration Shells: This ID is used in *hasDataSpecification/dataSpecification*.

This namespace has the qualifier "UoM:".

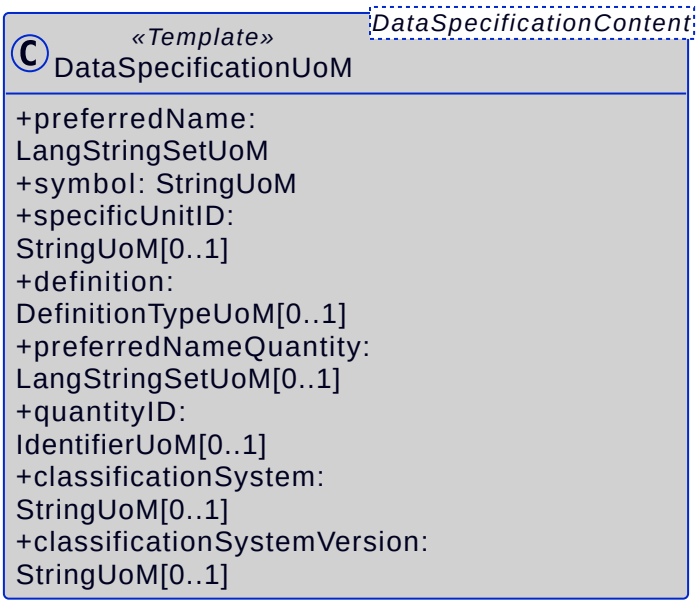


Figure 5. Metamodel of Data Specification Unit of Measurement

Identifikation of the Concept Description

The identification of the Concept Description, *conceptDescription/id*, should be identical to the unique global identifier of the Unit of Measurement, if available. If no unique global identifier is available, the *conceptDescription/id* can be chosen arbitrarily. In this case the unit's identifier as defined by the classification system must be denoted in the attribute *specificUnitID* (see below).

Data Specification Attributes

Class:	*DataSpecificationUoM <<Template>>*		
Explanation:	Content of data specification template for concept descriptions for Units of Measurement		
Inherits from:	DataSpecificationContent		
id:	https://admin-shell.io/DataSpecificationTemplates/DataSpecificationUoM/3/0/DataSpecificationUoM		
Attribute:	**ID**		
	Explanation	**Type**	**Card.**
preferredName	https://admin-shell.io/DataSpecificationTemplates/DataSpecificationUoM/3/0/DataSpecificationUoM/preferredName		
	Preferred name of the unit in different languages as defined by the classification system Example: "millimetre"	LangStringSetUoM	1
symbol	https://admin-shell.io/DataSpecificationTemplates/DataSpecificationUoM/3/0/DataSpecificationUoM/symbol		
	Symbol of the unit Example: “mm”	StringUoM	1
specificUnitID	https://admin-shell.io/DataSpecificationTemplates/DataSpecificationUoM/3/0/DataSpecificationUoM/specificUnitID		
	ID of the unit defined by the classification system Note: This is relevant if the classification system references the units differently from the unique *conceptDescription/id*. This can e.g. be UNECE common codes or proprietary keys. Example: "MMT"	StringUoM	0..1
definition	https://admin-shell.io/DataSpecificationTemplates/DataSpecificationUoM/3/0/DataSpecificationUoM/definition		
	Definition of the unit in different languages Example: "0,001 fold of the SI base unit metre"@en	DefinitionTypeUoM	0..1

preferredNameQuantity	https://admin-shell.io/DataSpecificationTemplates/DataSpecificationUoM/3/0/DataSpecificationUoM/preferredNameQuantity		
	Preferred name of the unit's quantity in different languages as defined by the classification system. *preferredNameQuantity* and *quantityID* must be consistent if both attributes are set. If the classification system lists multiple quantities for the unit, exactly one of those quantities must be selected in accordance with the properties purpose. Example: "distance"	LangStringSetUoM	0..1
quantityId	https://admin-shell.io/DataSpecificationTemplates/DataSpecificationUoM/3/0/DataSpecificationUoM/quantityId		
	ID of the unit's quantity. preferredNameQuantity and quantityID must be consistent if both attributes are set Example: 0173-1#Z4-BAJ199#002	IdentifierUoM	0..1
classificationSystem	https://admin-shell.io/DataSpecificationTemplates/DataSpecificationUoM/3/0/DataSpecificationUoM/classificationSystem		
	Name of the classification system used for the definition of units and quantities Note: The most prominent classification systems should be denoted as follows: "ECLASS", "IEC CDD", "UNECE Rec 20", "UNECE Rec 21", "BIPM", "OPC UA", "SAMM". This list is not exclusive. Example: "ECLASS Basic"	StringUoM	0..1
classificationSystemVersion	https://admin-shell.io/DataSpecificationTemplates/DataSpecificationUoM/3/0/DataSpecificationUoM/classificationSystemVersion		
	Version of the classification system used for the definition of units and quantities The version shshould be denoted exactly as provided by the classification system. Example: "15.0"	StringUoM	0..1

Primitive and Simple Data Types (normative)

All simple data types from Part 1 [1] apply also to the specifications described in this document.

Basic and Primitive Data Types

Table 1 lists the Primitives used in the metamodel. Primitive data types start with a capital letter.

Table 1. Primitive DataTypes Used in Metamodel

Primitive	Definition	Value Examples

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 *langString* depends on the serialization rules for a technology as defined in Part 1 Metamodel, IDTA-01001.
		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": [ { "language":"en", "text": "This is a multi-language value in English." }, { "language":"de", "text": "Das ist ein Multi-Language-Wert in Deutsch." } ]`

LangStringSetUoM	*LangStringSet* Each *langString* within the array of *strings* has a length of maximum 256 and a minimum of 1 characters.	(see examples for *LangStringSet*)

Primitive	Definition	Value Examples
DefinitionTypeUoM	*LangStringSet* Each *langString* within the array of *strings* has a length of maximum 2048 and a minimum of 1 characters.	"0,001 fold of the SI base unit metre"@en (also see examples for *LangStringSet*)
StringUoM	*string* A *string* with a length of maximum 256 and a minimum of 1 characters. Note: *string* is a xsd data type	"OPC UA" (also see examples for *LangStringSet*)
IdentifierUoM	*string* A *string* with a length of maximum 2048 and a minimum of 1 characters. Note: *string* is a xsd data type	"0173-1#02-BAA120#008" (also see examples for *LangStringSet*)

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

Part 1 of this document series (IDTA-01001) introduces different serialization formats: XML, JSON and RDF. Part 1 also introduces the implementation guide for embedded data specifications. This is why the different formats are described in IDTA-01001.

Summary and Outlook

This document provides a metamodel for specifying a data specification template for defining concept descriptions for Units of Measurement (UoM). The template includes a basic set of attributes to describe a Unit of Measurement and reference to external dictionaries for further information.

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

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

- the information model that is the basis for file exchange and interface payload definition (IDTA-01001, Part 1),
- interfaces and APIs for accessing the information of Asset Administration Shells (IDTA-01002, Part 2),
- security aspects including access control (IDTA-01004, part 4),
- a file exchange format AASX (IDTA-01005, Part 5).

Future versions of this document with an extended set of attributes are in discussion.

Annex

Examples

This section provides a number of examples to illustrate how to model Units of Measurement using the given data specification template.

Motivation

The publicly avbailable dictionaries for Units of Measurement vary greatly in their modelling philosophy and scope. They do not only differ in the number of units they contain, but also in the number of descriptive attributes per unit. Not all dictionaries were created to be "machine interpretable". Some dictionaries are multi-lingual, some make use of extended UNICODE symbols. To accomodate the different sources for units the given data specification template for Units of Measurement is very flexible to use. The examples below provide an orientation on how to use that flexibility to map the information from the most prominent public dictionaries to the given data specification template.

Examples

The following modelling rules have been applied to the examples and are highly recommended:

- The attributes for an UoM should be consistent, i.e. they should all come from the same source (data dictionary). It is counterproductive to mix sources, e.g. by providing an UNECE REC 20 ID, but using the definition from ECLASS. In case of discrepencies between the dictionaries the receiver of the model cannot resolve these conflicts and does not know which source has precedence.
- For the same reason the *classificationSystemVersion* should be consistent throughout an application. The units should not be mixed from different data dictionary versions.
- The source that was actually used and verified for the units should be used as a modelling basis. E.g., if the source for the unit definition is a SAMM model then this model should be used consistently to fill the attributes. Even though the units in SAMM are based on UNECE REC 20, the UNECE data dictionary should only be referenced if the modeller used this as the original source. It is good scientific practice to only cite the sources that were actually used and not the sources cited by sources.
- Sometimes a UoM is applicable for multiple quantities as denoted by the data dictionaries. However, when used for quantifying a properties value, the appropriate quantity has to be selected.

The tables below are read as follows:

- The first two columns, **attribute** and **card**, provide the respective attribute names and cardinality specified in this data specification template.
- The third column, **mapping recommendation**, provides the information on how to map a unit definition from the given data dictionary to the respective AAS atribute.
  - "-----" means that it is recommended to omit this optional attribute.
  - [no recommendation] means that an entry has to be provided, but no general mapping rule can be provided.
- For simplicity reasons the examples are only provided in English language even if the respective attribute has a multilanguage datatype.

ECLASS

The modelling of UoM in ECLASS Basic and Advanced is similiar, but not identical, even though they are based on the same dataset. Some attributes are named differently and some attributes are only available in Basic or Advanced.

Because the version of the property definition is coded into the IRDI/IRI of the property, it is not necessary to supply version information via the attribute *classificationSystemVersion*. However, it can be helpful to supply the catalogue version anyway, because it is not possible to directly infer the catalogue version from the property version. Snapshots of ECLASS models are depicted in Figures Figure 6 and Figure 7.

Table 2. Mapping recommendation and examples for ECLASS Basic

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2	Example 3
conceptDescription/id	1	*IrdiUN* of ECLASS unit [1]	0173-1#05-AAA480#005	https://api.eclass-cdp.com/0173-1-05-AAA480-003	0173-1#05-AAA793#002
preferredName	1	*PreferredName* of ECLASS unit	millimetre	millimetre	Kilogramm^{-1}
symbol	1	*ShortName* of ECLASS unit	mm	mm	1/kg
specificUnitID	0..1	Optional. Recommended to supply *IRDI* if *IRI* is provided as *conceptDescription/id*		0173-1#05-AAA480#003	
definition	0..1	*Definition* of ECLASS unit	0,001-fold of the SI base unit metre	0,001-fold of the SI base unit metre	reciprocal of the SI base unit kilogram
preferredNameQuantity	0..1	*NameOfDedicatedQuantity* of ECLASS unit	distance	distance	reciprocal mass
quantityID	0..1	-----[2]			
classificationSystem	0..1	"ECLASS"	ECLASS	ECLASS	ECLASS
classificationSystemVersion	0..1	*ECLASS version*	15.0	14.0	10.0

Table 3. Mapping recommendation and examples for ECLASS Advanced

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2	Example 3
conceptDescription/id	1	*id* of ECLASS unit [1]	0173-1#05-AAA480#005	https://api.eclass-cdp.com/0173-1-05-AAA480-003	0173-1#05-AAA793#002
preferredName	1	*preferred_name* of ECLASS unit	millimetre	millimetre	Kilogramm^{-1}
symbol	1	*short_name* of ECLASS unit	mm	mm	1/kg

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2	Example 3
specificUnitID	0..1	Optional. Recommended to supply *IRDI* if *IRI* is provided as conceptDescription/id		0173-1#05-AAA480#003	
definition	0..1	*definition* of ECLASS unit	0,001-fold of the SI base unit metre	0,001-fold of the SI base unit metre	reciprocal of the SI base unit kilogram
preferredNameQuantity	0..1	*QuantityReference/name* of ECLASS unit	distance	distance	reciprocal mass
quantityID	0..1	*QuantityReference/url* of ECLASS unit	0173-1#Z4-BAJ199#002	https://api.eclass-cdp.com/0173-1-Z4-BAJ199-002	0173-1#Z4-AAB646#002
classificationSystem	0..1	"ECLASS"	ECLASS	ECLASS	ECLASS
classificationSystemVersion	0..1	*ECLASS version*	15.0	14.0	10.0

	A	PX	PY
1	**PreferredName**	millimetre	cubic metre per cubic metre
2	**ShortName**	mm	m^3/m^3
3	**Definition**	0,001-fold of the SI base unit metre	power of the SI base unit metre with the exponent 3 divided by the power of the SI base unit metre with the exponent 3
4	**Source**		
5	**Comment**		
6	**SINotation**	mm	m^3/m^3
7	**SIName**		
8	**DINNotation**	mm	$m^3\ m^{-3}$
9	**ECEName**		
10	**ECECode**	MMT	J91
11	**NISTName**		
12	**IECClassification**	0112/2///62720#UAA862	0112/2///62720#UAA767
13	**IrdiUN**	0173-1#05-AAA480#005	0173-1#05-AAA481#006
14	**NameOfDedicatedQuantity**	distance	volume fraction

Figure 6. Snapshot of UoM descriptions in ECLASS Basic v16.0 (CSV-Export). Copyright by ECLASS e.V.

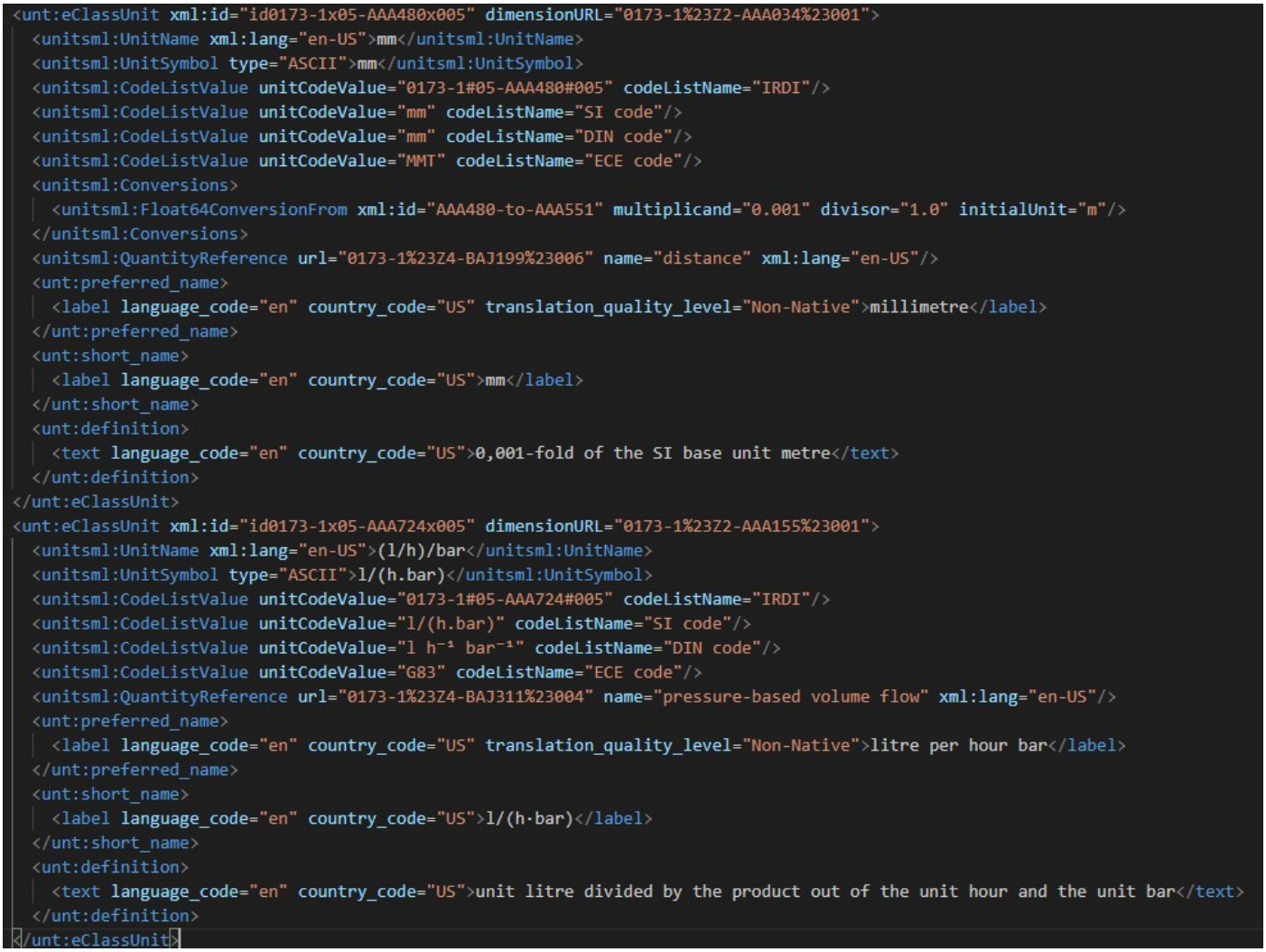


Figure 7. Snapshot of UoM model in ECLASS Advanced v16.0 (XML-Export). Copyright by ECLASS e.V.

IEC CDD

IEC CDD and ECLASS can be mapped in a very similiar way. Because the IEC CDD has no version of the overall catalogue, the atribute *classificationSystemVersion* should be omitted. Snapshots of IEC CDD models are depicted in Figures Figure 8 and Figure 9.

Table 4. Mapping recommendation and examples for IEC CDD

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2	Example 3
conceptDescription/id	1	*IRDI* of IEC CDD unit	0112/2///62720#UAA862#001	https://cdd.iec.ch/cdd/iec62720/iec62720.nsf/Units/0112-2---62720%23UAA296	0112/2///62720#UAA017#002
preferredName	1	*Preferred name* of IEC CDD unit	millimetre	Volt	ohm
symbol	1	*Short name* of IEC CDD unit	mm	V	¬

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2	Example 3
specificUnitID	0..1	Optional. Recommended to supply *IRDI* if *IRI* is provided as conceptDescription/id		0112/2///62720#UAB246#001	
definition	0..1	*Definition* of IEC CDD unit		SI derived unit with special name, defined as the potential difference between two points in an electric circuit when one joule of work is required to move one coulomb of charge, and for which the following relation applies: 1 V = 1 J/C (by Ohm' law: 1 V = 1 W/A)	SI derived unit with special name, for which the following relation applies: 1 ¬ = 1 $C \cdot A^{-1}$ = 1 S^{-1} = 1 $m^2 \cdot kg \cdot s^{-3} \cdot A^{-2}$
preferredNameQuantity	0..1	*Preferred name* of the units quantity [3]	length	electric potential difference	electric resistance
quantityID	0..1	*IRDI* of the units quantity [3]	0112/2///62720#UAD072#001	https://cdd.iec.ch/cdd/iec62720/iec62720.nsf/ListsOfUnitsAllVersions/0112-2---62720%23UAD237	0112/2///62720#UAD045#001
classificationSystem	0..1	"IEC CDD"	IEC CDD	IEC CDD	IEC CDD
classificationSystemVersion	0..1	"-----"			

Code:	0112/2///62720#UAB246
Version:	001
Revision:	03
IRDI:	0112/2///62720#UAB246#001
Preferred name:	tex
Synonymous name:	
Short name:	tex
Definition:	unit for the indication of the linear mass of textile fibers and yarns
Note:	
Remark:	
Definition source:	NIST Guide for the Use of the International System of Units (SI). NIST Special Publication 811, 2008 Edition. NIST (National Institute of Standards and Technology), Gaithersburg, MD.
Definition class:	
Base unit:	
Unit structure:	m^{-1} kg
Unit in text:	tex
Unit in SGML/XML:	<math xmlns="http://www.w3.org/1998/Math/MathML" display="block"> <!-- UAB246 --> <mrow> <mi mathvariant="normal">tex</mi> </mrow> </math> tex
UN/ECE code:	D34
Unit conversion:	
Formula in text:	
Data object identifier:	
Time stamp:	
Applicable properties:	
Applicable list of units:	0112/2///62720#UAD079 - lineic mass
Status level:	Standard
Published in:	
Version initiation date:	2014-02-11
Version release date:	2014-02-11
Revision release date:	2023-07-24
Responsible Committee:	IEC/SC 3D
Change request ID:	C00124
⊞ Version history:	001-03 (2023-11-07 12:19:50 by BATCH 00001255) standard

Figure 8. View of the IEC CDD entry for the UoM tex. Copyright by the International Electronical Commision (IEC)

Code:	0112/2///62720#UAD079
Version:	001
Revision:	02
IRDI:	0112/2///62720#UAD079#001
Preferred name:	lineic mass
Synonymous name:	
Short name:	
Definition:	quantity "lineic mass"
Note:	NOTE 1: SI coherent derived unit expressed in terms of base units: m^{-1} kg NOTE 2: For definition of "lineic mass" see Figure referenced in attribute "Drawing".
Remark:	
Definition source:	ISO 80000-4:2006, 4-6
Definition class:	
SI coherent derived unit:	0112/2///62720#UAA616 - kilogram per metre
Codes of units:	0112/2///62720#UAA616 - kilogram per metre 0112/2///62720#UAB376 - gram per millimetre 0112/2///62720#UAB070 - kilogram per millimetre 0112/2///62720#UAA485 - gram per metre 0112/2///62720#UAB495 - kilogram per kilometre 0112/2///62720#UAA828 - milligram per metre 0112/2///62720#UAB246 - tex 0112/2///62720#UAB244 - denier 0112/2///62720#UAA670 - pound (avoirdupois) per foot 0112/2///62720#UAB071 - pound (avoirdupois) per inch 0112/2///62720#UAB245 - pound (avoirdupois) per yard
Data object identifier:	
Drawing:	DUAD079
Time stamp:	
Applicable properties:	
Status level:	Standard
Published in:	IEC 62720
Version initiation date:	2014-02-11
Version release date:	2014-02-11
Revision release date:	2023-07-24
Responsible Committee:	IEC/SC 3D
Change request ID:	C00124
⊞ Version history:	001-02 (2023-11-07 12:09:59 by BATCH 00001255) standard

Figure 9. View of the IEC CDD entry for the quantity "lineic mass". Copyright by the International Electronical Commision (IEC)

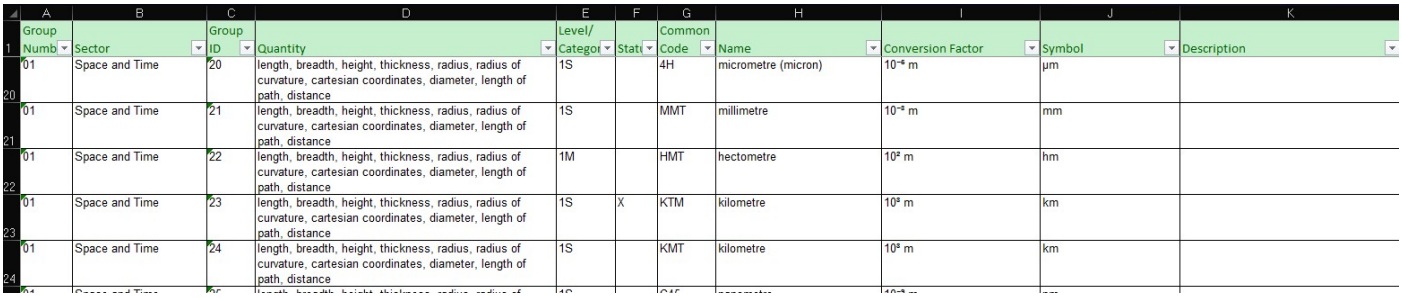
UNECE

In UNECE REC 20 and 21 the UoM are identified by a 2-3 letter code. While this code is unique within the specific list, it is not usable as a globally unique identifier. Therefore, a different identifier for the ConceptDescription has to be created. The examples illustrate different ways this could be achieved without giving a recommendation. UNECE snapshots are depicted in Figures Figure 10 and Figure 11.

Table 5. Mapping recommendation and examples for UNECE REC 20

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2	Example 3
conceptDescription/id	1	[no recommendation]	uncefact:UNECERec20Code/MMT	uncefact:rec20:kilocandela	https://service.unece.org/trade/uncefact/vocabulary/rec20#page

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2	Example 3
preferredName	1	*Name* of UNECE unit	millimetre	kilocandela	page
symbol	1	*Representation symbol* of UNECE unit	mm	kcd	
specificUnitID	0..1	*Common code* of UNECE unit	MMT	P33	ZP
definition	0..1	*Description* of UNECE unit		1000-fold of the SI base unit candela.	A unit of count defining the number of pages.
preferredNameQuantity	0..1	*Quantity* of UNECE unit	length	luminous intensity	
quantityID	0..1	-----			
classificationSystem	0..1	"UNECE Rec 20"	UNECE Rec 20	UNECE Rec 20	UNECE Rec 20
classificationSystemVersion	0..1	*Revision* of UNECE REC 20 specification	17	16	17



	A	B	C	D	E	F	G	H	I	J	K
1	Group Numb	Sector	Group ID	Quantity	Level/ Categor	Stat	Common Code	Name	Conversion Factor	Symbol	Description
20	01	Space and Time	20	length, breadth, height, thickness, radius, radius of curvature, cartesian coordinates, diameter, length of path, distance	1S		4H	micrometre (micron)	10^{-6} m	µm	
21	01	Space and Time	21	length, breadth, height, thickness, radius, radius of curvature, cartesian coordinates, diameter, length of path, distance	1S		MMT	millimetre	10^{-3} m	mm	
22	01	Space and Time	22	length, breadth, height, thickness, radius, radius of curvature, cartesian coordinates, diameter, length of path, distance	1M		HMT	hectometre	10^{2} m	hm	
23	01	Space and Time	23	length, breadth, height, thickness, radius, radius of curvature, cartesian coordinates, diameter, length of path, distance	1S	X	KTM	kilometre	10^{3} m	km	
24	01	Space and Time	24	length, breadth, height, thickness, radius, radius of curvature, cartesian coordinates, diameter, length of path, distance	1S		KMT	kilometre	10^{3} m	km	

Figure 10. View of exemplary UNECE REC 20 entries (Revision 17, Annex I). Copyright by the United Nations Economic Commission for Europe (UNECE)

Table 6. Mapping recommendation and examples for UNECE REC 21

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2	Example 3
conceptDescription/id	1	[no recommendation]	uncefact:UNECERec21Code:8A	uncefact:rec21:piece	https://service.unece.org/trade/uncefact/vocabulary/rec21/#Parcel
preferredName	1	*Name* of UNECE unit	Pallet, wooden	Piece	Parcel

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2	Example 3
symbol	1	*Name* of UNECE unit	Pallet, wooden	Piece	Parcel
specificUnitID	0..1	*Code* of UNECE unit	8A	PP	PC
definition	0..1	*Description* of UNECE unit	A platform or open-ended box, made of wood, on which goods are retained for ease of mechanical handling during transport and storage.	A loose or unpacked article.	
preferredNameQuantity	0..1	-----			
quantityID	0..1	-----			
classificationSystem	0..1	"UNECE Rec 21"	UNECE Rec 21	UNECE Rec 21	UNECE Rec 21
classificationSystemVersion	0..1	*Revision* of UNECE REC 21 specification	12	11	12

[example UNECE2] | *example_UNECE2.jpg*

Figure 11. View of exemplary UNECE REC 21 entries (Revision 12, Annexes V and VI). Copyright by United Nations Economic Commission for Europe

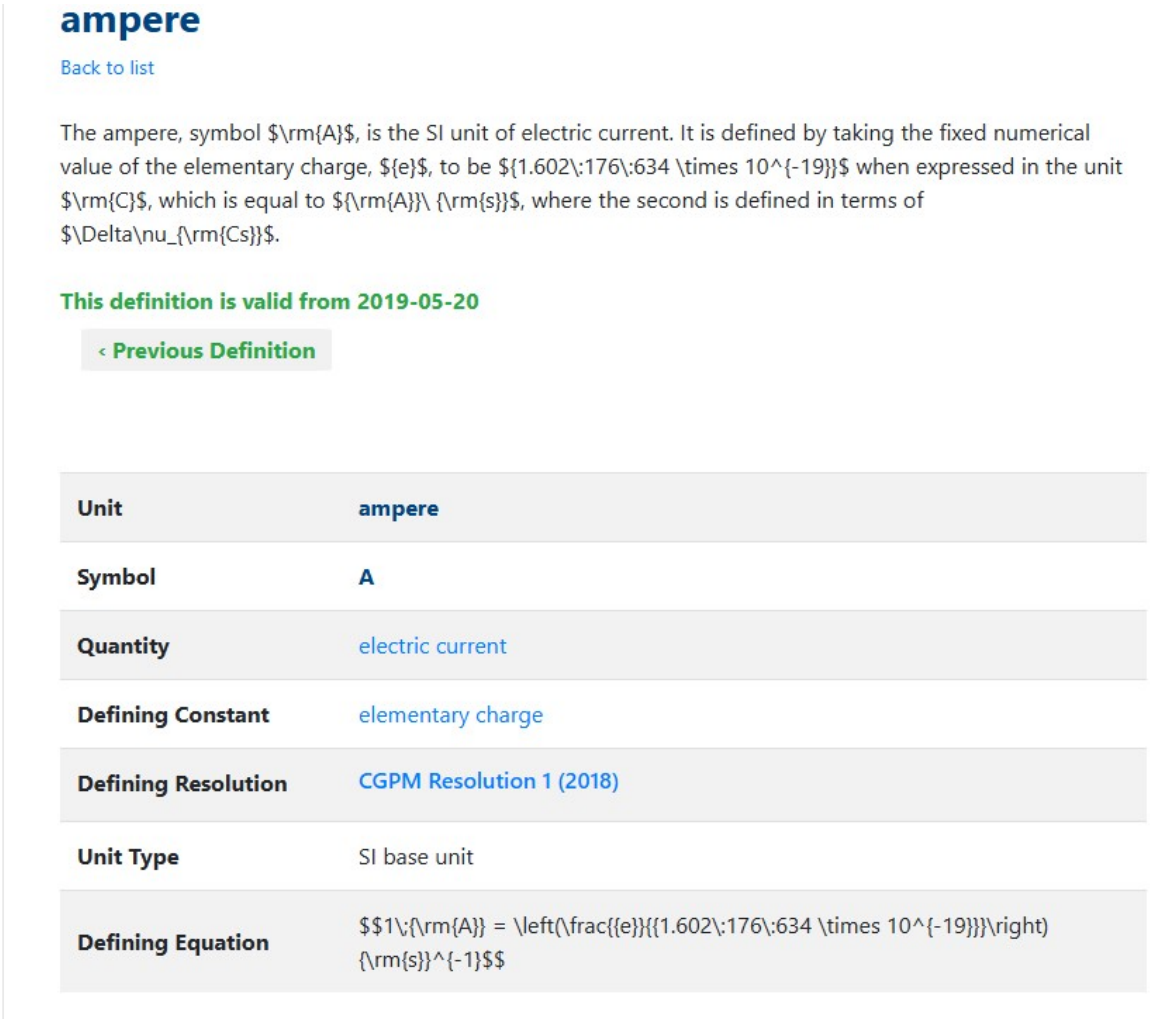
BIPM (Bureau International des Poids et Mesures) / SI Digital Framework

The BIPM focuses on the maintenance of the SI unit definitions, but also provides a definition of selected non-SI units and constants. In case of compound units the SI digital framework provides a method to generate the PID and semantic definition based on the SI units. However, in these cases a quantity, verbal description and full name are not provided. BIPM snapshots are depicted in Figures Figure 12 and Figure 13.

Table 7. Mapping recommendation and examples for the SI Digital Framework

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2	Example 3
conceptDescription/id	1	*PID* of BIPM (compound) unit	https://si-digital-framework.org/SI/units/ampere	https://si-digital-framework.org/SI/units/dalton	https://si-digital-framework.org/SI/units/millimetre
preferredName	1	*Unit* or *Compound unit* of BIPM unit	ampere	dalton	mm

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2	Example 3
symbol	1	*Symbol* of BIPM unit	A	Da	mm
specificUnitID	0..1	-----			
definition	0..1	*Definition* of BIPM unit	The ampere, symbol A, is the SI unit of electric current. It is defined by taking the fixed numerical value of the elementary charge, e, to be $1.602176634x10^{-19}$ when expressed in the unit C, which is equal to As, where the second is defined in terms of delta_v_cs.		
preferredNameQuantity	0..1	*Quantity* of the BIPM unit	electric current	mass	
quantityID	0..1	*PID* of the units quantity	https://si-digital-framework.org/quantities/ELCU	https://si-digital-framework.org/quantities/MASS	
classificationSystem	0..1	"BIPM"	BIPM	BIPM	BIPM
classificationSystemVersion	0..1	*valid from*	2019-05-20		



ampere

Back to list

The ampere, symbol $\rm{A}$, is the SI unit of electric current. It is defined by taking the fixed numerical value of the elementary charge, ${e}$, to be ${1.602\:176\:634 \times 10^{-19}}$ when expressed in the unit $\rm{C}$, which is equal to ${\rm{A}}\ {\rm{s}}$, where the second is defined in terms of $\Delta\nu_{\rm{Cs}}$.

This definition is valid from 2019-05-20

‹ Previous Definition

Unit	**ampere**
Symbol	**A**
Quantity	electric current
Defining Constant	elementary charge
Defining Resolution	CGPM Resolution 1 (2018)
Unit Type	SI base unit
Defining Equation	$$1\;{\rm{A}} = \left(\frac{{e}}{{1.602\:176\:634 \times 10^{-19}}}\right) {\rm{s}}^{-1}$$

Figure 12. View of the SI digital framework entry for the UoM Ampere. Copyright by BIPM

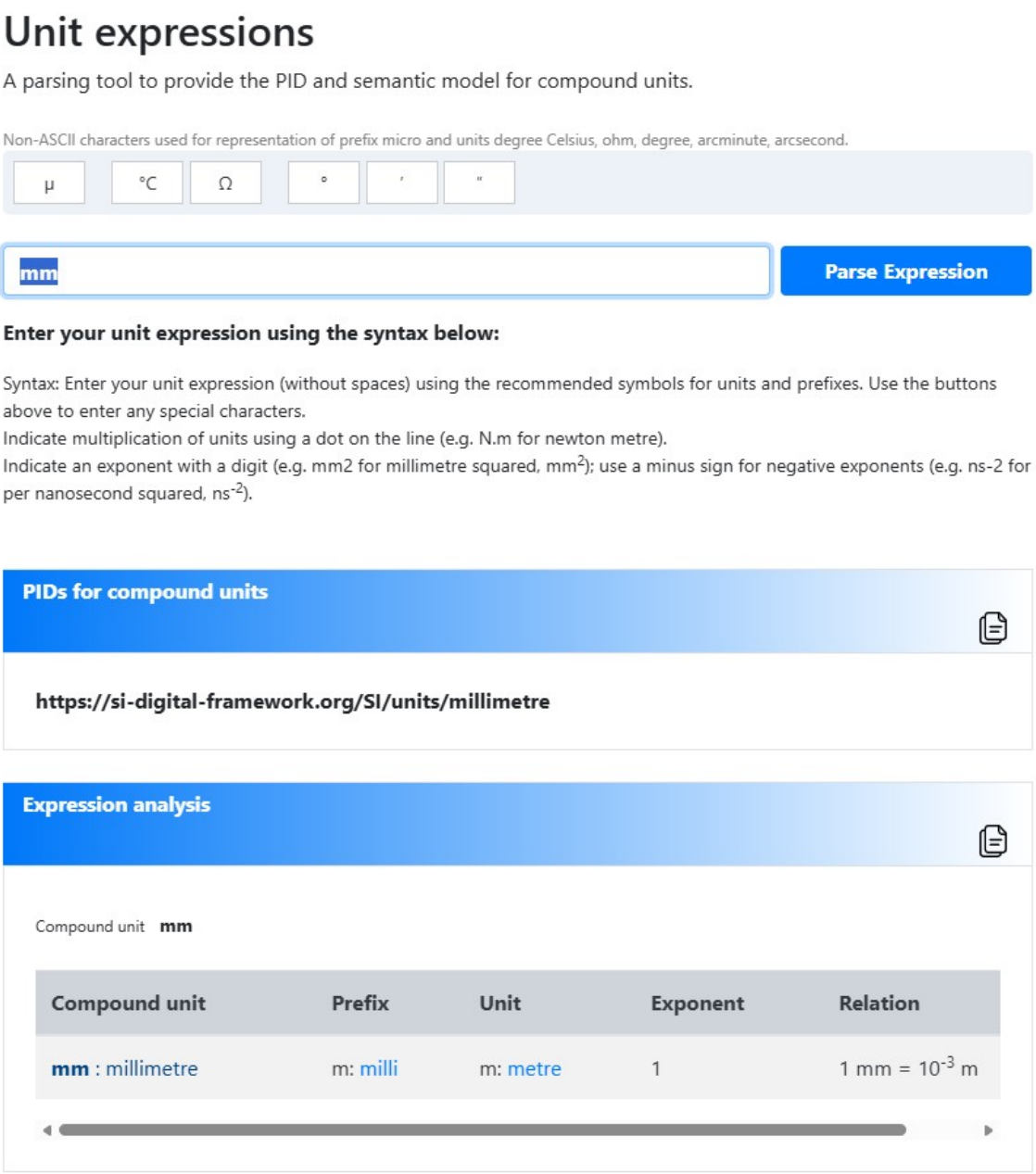


Figure 13. View of the SI digital framework entry for the compound unit millimeter. Copyright by BIPM

Other meta models

Table 8. Mapping recommendation and examples for OPC UA

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2	Example 3
conceptDescription/id	1	[no recommendation]	http://www.opcfoundation.org/UA/units/5131860	http://www.opcfoundation.org/OPC34100/5720146	http://example-company/myCompanionSpec/UAA862
preferredName	1	*description*	millimetre	watt hour	millimetre
symbol	1	_displayName	mm	W·h	mm
specificUnitID	0..1	*UnitID*	5131860	5720146	UAA862

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2	Example 3
definition	0..1	-----			
preferredNameQuantity	0..1	*QuantityType*	length	work	length
quantityID	0..1	-----			
classificationSystem	0..1	*namespaceUri*	http://www.opcfoundation.org/UA/units/un/cefact	http://www.opcfoundation.org/UA/units/un/cefact	http://www.opcfoundation.org/UA/units/ieccdd
classificationSystemVersion	0..1	-----			

Table 9. Mapping recommendation and examples for SAMM

attribute (of data specification)	card	mapping recommendation	Example 1	Example 2
conceptDescription/id	1	*Unit element URN*	urn:samm:org.eclipse.esmf.samm:unit:2.2.0#degreeCelsius	urn:samm:org.eclipse.esmf.samm:unit:2.2.0#percent
preferredName	1	*Preferred name* of unit	degreeCelsius	percent
symbol	1	*symbol* of unit	°C	%
specificUnitID	0..1	-----		
definition	0..1	*description*		
preferredNameQuantity	0..1	*quantityKind/preferredName* of unit	temperature	dimensionless
quantityID	0..1	*quantityKind* of unit	urn:samm:org.eclipse.esmf.samm:unit:2.2.0#temperature	urn:samm:org.eclipse.esmf.samm:unit:2.2.0#dimensionless
classificationSystem	0..1	"SAMM"	SAMM	SAMM
classificationSystemVersion	0..1	Version	2.2	2.2

OPC UA uses the attribute *namespaceUri* to denote the classification system that has been used as the source for the UoM. The mapping of the attributes from that classification system is described in OPC UA Specification Part 8, Section 5.6.3.4.

SAMM is based on UNECE Rec 20. The open source model is published as part of the Eclipse Semantic Modeling Framework (ESMF): https://eclipse-esmf.github.io/samm-specification/snapshot/units.html.

When using mete models such as OPC UA or SAMM as the source for UoM, one has to be aware that this introduces "double-mapping". E.g. UNECE Rec 20 attributes are mapped to an OPC UA nodeset and then the attributes of that nodeset are mapped to the AAS metamodel. Such "double-mapping" often leads to a degradation of the information and should be avoided, if possible. However, as stated above, the source that was actually used and veryfied for the units should be used as a modelling basis. I.e., if and only if the source for the unit definition is a SAMM model then this model should be used consistently to fill the attributes of the Concept Description template even though SAMM is based on UNECE Rec 20.

Examples of not recommended modelling approaches

The following table lists some modelling mistakes that should be avoided.

Table 10. Negative examples, i.e. discoraged moddeling approaches

attribute (of data specification)	card	Example 1	Example 2
conceptDescription/id	1	0112/2///62720#UAA023#001	MMT
preferredName	1	ångström	millimetre
symbol	0	Å	mm
specificUnitID	0..1	A11	MMT
definition	0..1	0,000 000 000 1-fold of the SI base unit metre	
preferredNameQuantity	0..1		
quantityID	0..1		
classificationSystem	0..1	UNECE	
classificationSystemVersion	0..1		
explanation		Not recommended, because information from multiple sources is mixed	"MMT" is not a unique ID and therefore not suitable as an identifier for conceptDescriptions. The omission of *classificationSystem* makes it impossible to use the original UNECE catalogue.

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.

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
```

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 14 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 represent the end of binary associations, see also Figure 22. In this case, the instance of *Class2* is navigable via the instance of the owning class *Class1*.[4]

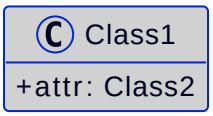


Figure 14. Class

Figure 15 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*.

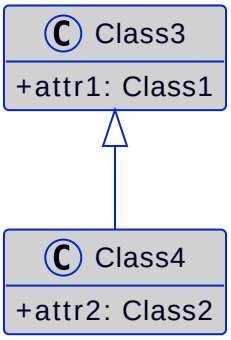


Figure 15. Inheritance/Generalization

Figure 16 defines the required and allowed multiplicity/cardinality within an association between instances of *Class1* and *Class2*. In this example, an instance of *Class2* is always related to exactly one instance of *Class1*. An instance of *Class1* is either related to none, one, or more (unlimited, i.e. no constraint on the upper bound) instances of *Class2*. The relationship can change over time.

Multiplicity constraints can also be added to attributes and aggregations.

The notation of multiplicity is as follows:

<lower-bound>.. <upper-bound>

where <lower-bound> is a value specification of type Integer - i.e. 0, 1, 2, … - and <upper-bound> is a value specification of type UnlimitedNatural. The star character (*) is used to denote an unlimited upper bound.

The default is 1 for lower-bound and upper-bound.

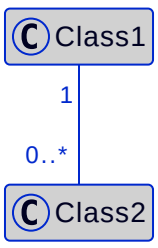


Figure 16. Multiplicity

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 17 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.

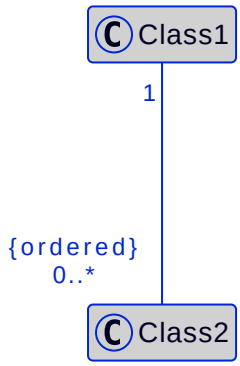


Figure 17. Ordered Multiplicity

Figure 18 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*.

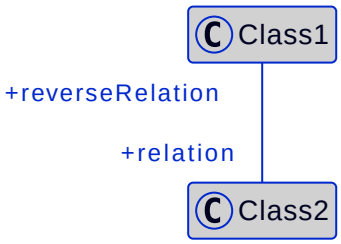


Figure 18. Association

Figure 19 depicts two classes connected by a unidirectional association from *Class1* to *Class2*. In this association, only the endpoint is navigable, meaning it is possible to navigate from an instance of *Class1* to an instance of *Class2*, but not the other way around. An instance of *Class1* can be in a 'relation' with an instance of *Class2*, and the association is labeled 'Reference'.

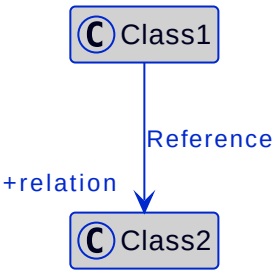


Figure 19. Association

A specialty in Figure 19 is that the label 'Reference' indicates the relationship between *Class1* and *Class2* is of a *Reference* type. This means that an instance of Class1 holds a reference to an instance of *Class2*. Furthermore, the instance of *Class2* is considered 'referable' according to the Asset Administration Shell metamodel, implying that it inherits from the predefined abstract class 'Referable' in the AAS framework. The structure of a reference to a model element of the Asset Administration Shell is explicitly defined.

Figure 20 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 20 shows the composition using an association relationship with a filled diamond as composition adornment.

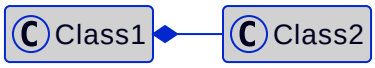


Figure 20. Composition (Composite Aggregation)

Figure 21 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 21 shows the shared aggregation using an association relationship with a hallow diamond as aggregation adornment.

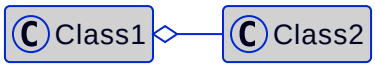


Figure 21. Aggregation

Figure 22 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.

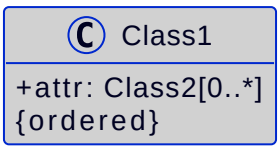


Figure 22. Navigable Attribute Notation for Associations

Figure 23 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.

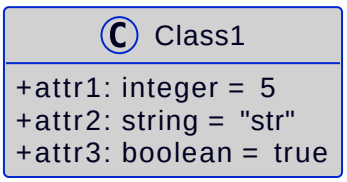


Figure 23. Default Value

Figure 24 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.

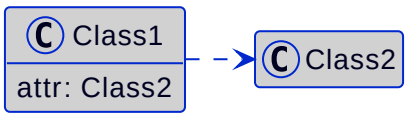


Figure 24. Dependency

Figure 25 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 25. Abstract Class

Figure 26 shows a package named "Package2". A package is a namespace for its members. In this example, the member belonging to *Package2* is class *Class2*.

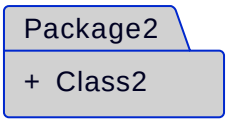


Figure 26. Package

Figure 27 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>>.

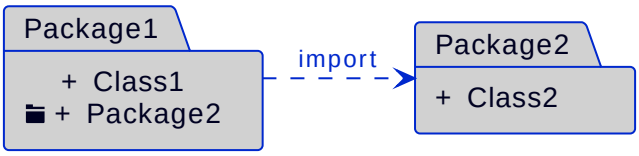


Figure 27. Imported Package

[annex::uml:::image-76-enumeration] 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 28 shows how a note can be attached to an element, in this example to class "Class1".

Figure 28. Note

Figure 29 shows how a constraint is attached to an element, in this example to class "Class1".

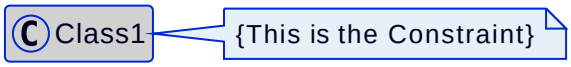


Figure 29. Constraint

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 to support such exceptions.

Templates, Inheritance, Qualifiers, and Categories

At first glance, there seems to be some overlapping within the concepts of data specification templates, extensions, inheritance, qualifiers, and categories introduced in the metamodel. This clause explains the commonalities and differences and gives hints for good practices.

In general, an extension of the metamodel by inheritance is foreseen. Templates might also be used as alternatives.

- Extensions can be used to add proprietary and/or temporary information to an element. Extensions do not support interoperability. They can be used as work-around for missing properties in the standard. In this case, the same extensions are attached to all elements of a specific class (e.g. to properties). However, in general, extensions can be attached in a quite arbitrary way. Properties are defined in a predefined way as key values pairs (in this case keys named "name").
- In contrast to extensions, templates aim at enabling interoperability between the partners that agree on the template. A template defines a set of attributes, each of them with clear semantics. This set of attributes corresponds to a (sub-)schema. Templates should only be used if different instances of the class follow different schemas and the templates for the schemas are not known at design time of the metamodel. Templates might also be used if the overall metamodel is not yet stable enough or a tool supports templates but not (yet) the complete metamodel. Typically, all instances of a specific class with the same category provide the same attribute values conformant to the template. In contrast to extensions, the attributes in the template have speaking names.

Note: categories are deprecated and should no longer be used.

- However, when using non-standardized proprietary data specification templates, interoperability cannot be ensured and thus should be avoided whenever possible.
- In case all instances of a class follow the same schema, inheritance and/or categories should be used.
- Categories can be used if all instances of a class follow the same schema but have different constraints depending on their category. Such a constraint might specify that an optional attribute is mandatory for this category (like the unit that is mandatory for properties representing physical values). Realizing the same via inheritance would lead to multiple inheritance – a state that is to be avoided[5].

Note: categories are deprecated and should no longer be used.

- Qualifiers are used if the structure and the semantics of the element is the same independent of its qualifiers. Only the quality or the meaning of the value for the element differs.
- Value qualifiers are used if only the quantity but not the semantics of the value changes. Depending on the application, either both value and qualifier define the "real" semantics together, or the qualifier is not really relevant and is ignored by the application. Example: the actual temperature might be good enough for non-critical visualization of trends, independent of whether the temperature is measured or just estimated (qualifier would denote: measured or estimated).
- Concept qualifiers are used to avoid multiplying existing semantically clearly defined concepts with the corresponding qualifier information, e.g. life cycle.

- Template qualifiers are used to guide the creation and validation of element instances.

Notes to Graphical UML Representation

Specific graphical modelling rules, which are used in this specification but not included in this form, are explained below [35].

Figure 30 shows different graphical representations of a composition (composite aggregation). In Variant A, a relationship with a filled aggregation diamond is used. In Variant B, an attribute with the same semantics is defined. And in Variant C, the implicitly assumed default name of the attribute in Variant A is explicitly stated. 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.

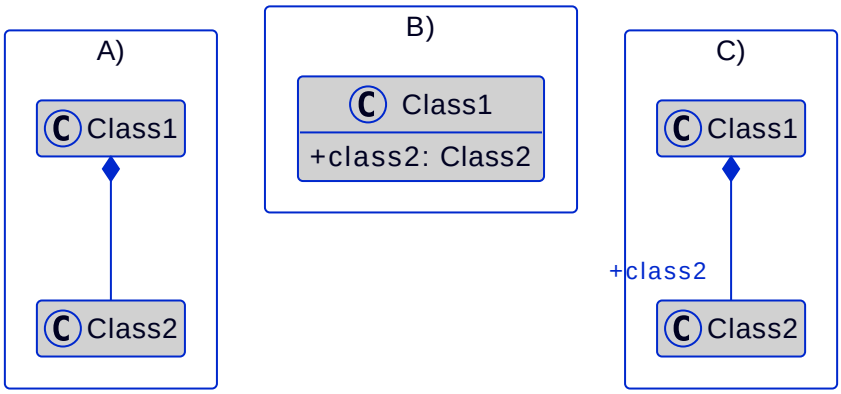


Figure 30. Graphical Representations of Composite Aggregation/Composition

Figure 31 shows a representation of a shared aggregation: a *Class2* instance can exist independently of a *Class1* instance. 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.

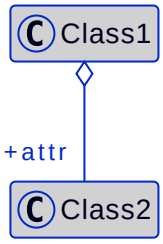


Figure 31. Graphical Representation of Shared Aggregation

Figure 32 show different graphical representations of generalization. Variant A is the classical graphical representation as defined in [12]. Variant B is a short form. 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.

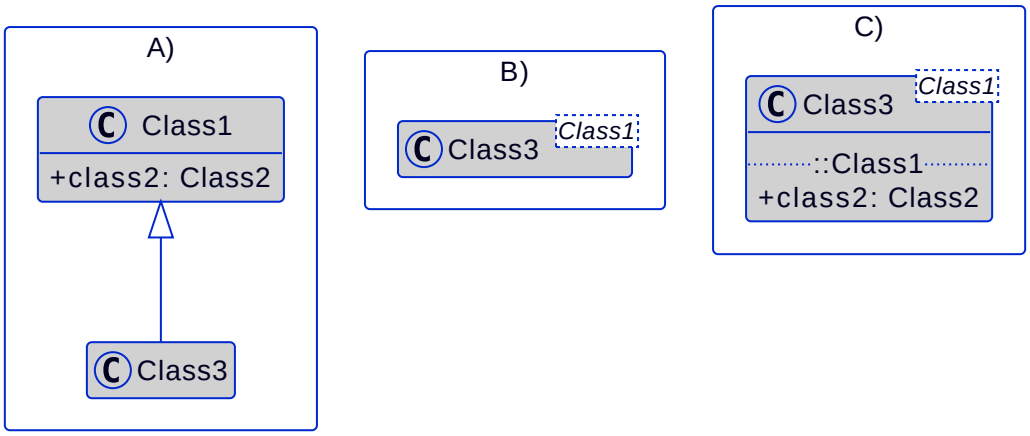


Figure 32. Graphical Representation of Generalization/Inheritance

Figure 33 depicts different graphical notations for enumerations in combination with inheritance. On the left side "Enumeration1" additionally contains the literals as defined by "Enumeration2".

Note 1: the direction of inheritance is opposite to the one for class inheritance. This can be seen on the right side of Figure 33 that defines the same enumeration but without inheritance.

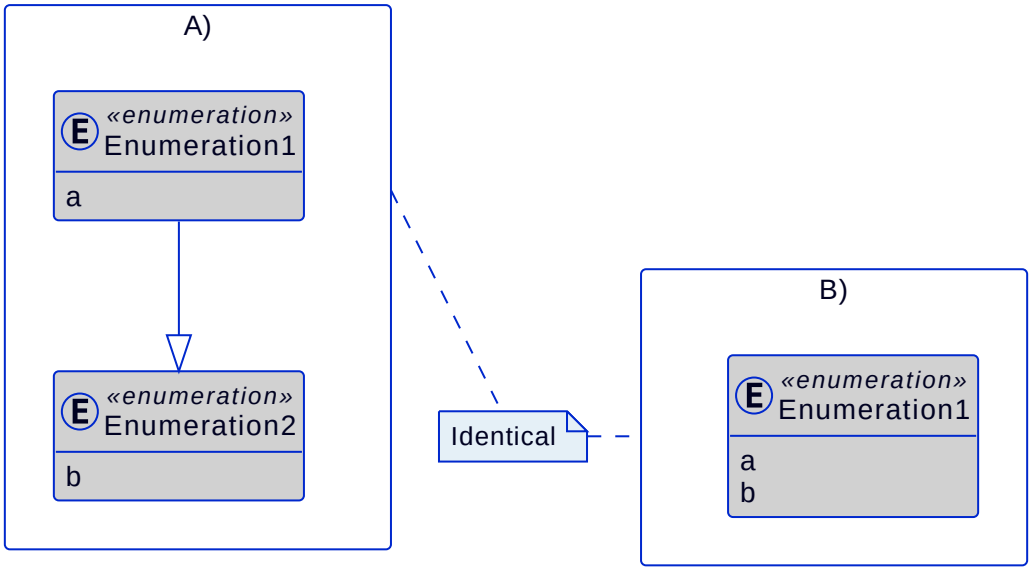


Figure 33. Graphical Representation for Enumeration with Inheritance

Note 2: in this specification all elements of an enumeration are ordered alphabetically.

Figure 34 shows an experimental class, marked by the stereotype "Experimental".

Figure 34. Graphical Representation for Experimental Classes

Figure 35 depicts a deprecated class, which is marked by the stereotype "Deprecated" whereas Figure 36 depicts a deprecated attribute within a class.

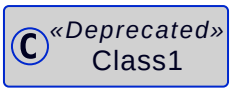


Figure 35. Graphical Representation for Deprecated Class

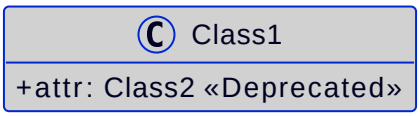


Figure 36. Graphical Representation for Deprecated Attribute

Figure 37 shows a class representing a template. It is marked by the stereotype "Template".

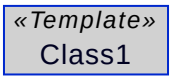


Figure 37. Graphical Representation of a Template Class

UML Table Templates

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 11. Class

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

ID is the metamodel ID of the class or attribute, conformant to the grammar defined in Text Serialization of Values of Type "Reference". A metamodel ID for a class attribute is concatenated by *<ID of metamodel element ID of class>/<relative metamodel element ID>*.

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:

- *<Class>:* Type is an object type (class); it is realized as composite aggregation (composition), and does not exist independent of its parent
- *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 UML , Figure Graphical Representation of Shared Aggregation; for more information on referencing see Referencing.
- *<Primitive>:* Type is no object type (class) but a data type; it is just a value, see Primitive and Simple Data Types
- *<Enumeration>:* Type is an enumeration, see Template for Enumerations

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 (optional) or the list is not empty. In the case of "0..3" there are at most 3 elements in the list. "1" means the attribute is mandatory. "1..**" or "1..3" means there is at least 1 element in the list. The "**" denotes as maximum an infinite number of elements of the corresponding type whereas "3" means that there are at most 3 elements in the list - analogous for other numbers.

Note 1: 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.

Note 2: attributes or attribute elements with data type "string" or "langString" are considered to consist of at least one character.

Note 3: optional lists, i.e. attributes with cardinality > 1 and minimum 0, are considered to consist of at least one element.

Examples for valid and invalid model references

If class type equal to "ModelReference<Submodel>", the following reference would be a valid reference (using the text serialization as defined in Text Serialization of Values of Type "Reference"):

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

This would be an invalid reference for "ModelReference<Submodel>" because it references a submodel element "Property":

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

If class type equal to "ModelReference<Referable>", the following references would be valid references (using the text serialization as defined in Text Serialization of Values of Type "Reference") because "Property" and "File" are Referables and "Submodel" itself is also Referable since all Identifiables are referable:

```
(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>" because FragmentReference is no Referable:

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

Template for Enumerations

Template 12. Enumeration

Enumeration:	<Enumeration Name> ["<<Experimental>>"] ["<<Deprecated>>"]
Explanation:	<Explanatory text>
Set of:	{<Enumeration> ";" }+ | "-"
ID:	<metamodel element ID>
Literal	**Explanation**
enumValue1>["<<Experimental>>"] ["<<Deprecated>>"]	<metamodel value ID>
	<Explanatory text>
<enumValue2> ["<<Experimental>>"] ["<<Deprecated>>"]	<metamodel value ID>
	<Explanatory text>

"**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.

"**Literal**" lists values of enumeration. All values that are element of one of the enumeration listed in "**Set of:**" are listed explicitly as well.

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 13. Primitives

Primitive	**ID**	
	Definition	**Value Examples**

<Name of Primitive>	<metamodel ID of Primitive>	
	<Explanatory text>	<Value examples>

Handling Constraints

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

If a constraint is mentioning the ID of the Template, then the same constraints shall also be valid for deprecated data specification template IDs.

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

[1] Alternatively ID can be provided as IRI starting with https://api.eclass-cdp.com/

[2] ECLASS basic does not provide the IDs of the quantities in its export

[3] The quantities are listed under "Applicable list of units"

[4] „ 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]

[5] Exception: multiple inheritance is used in this specification, but only in case of inheriting from abstract classes.

Change Log

General

The change notes list the notable changes made to the 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.

The change notes for a version consist of three parts:

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

If there are no changes the corresponding tables are omitted.

Changes V3.0

This is the first version of the document. It has been numbered as version 3.0 to be consistent with Part 1 of the specification series.

Bibliography

[1] IDTA-01001 "Specification of the Asset Administration Shell Part 1 – Metamodel". See [22].

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

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

[15] "Asset Administration Shell. Reading Guide". Industrial Digital Twin Association. November 2024. [Online]. Available: https://industrialdigitaltwin.org/wp-content/uploads/2024/11/2024-11_IDTA_AAS-Reading-Guide.pdf

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

[22] "Asset Administration Shell Specifications". [Online]. Available: https://industrialdigitaltwin.io

[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. [Online]. Available: https://industrialdigitaltwin.org/wp-content/uploads/2022/12/I40-IDTA-WS-Process-How-to-write-a-SMT-FINAL-.pdf

[100] "International vocabulary of metrology — Basic and general concepts and associated terms (VIM)". 2008. BIPM. [Online]. Available: http://www.bipm.org/utils/common/documents/jcgm/JCGM_200_2008.pdf

[101] "ISO/IEC 80000-1. Quantities and units - Part 1: General". 2022. ISO/IEC.

[102] "IEC 61360. Standard data element types with associated classification scheme". 2017. IEC.