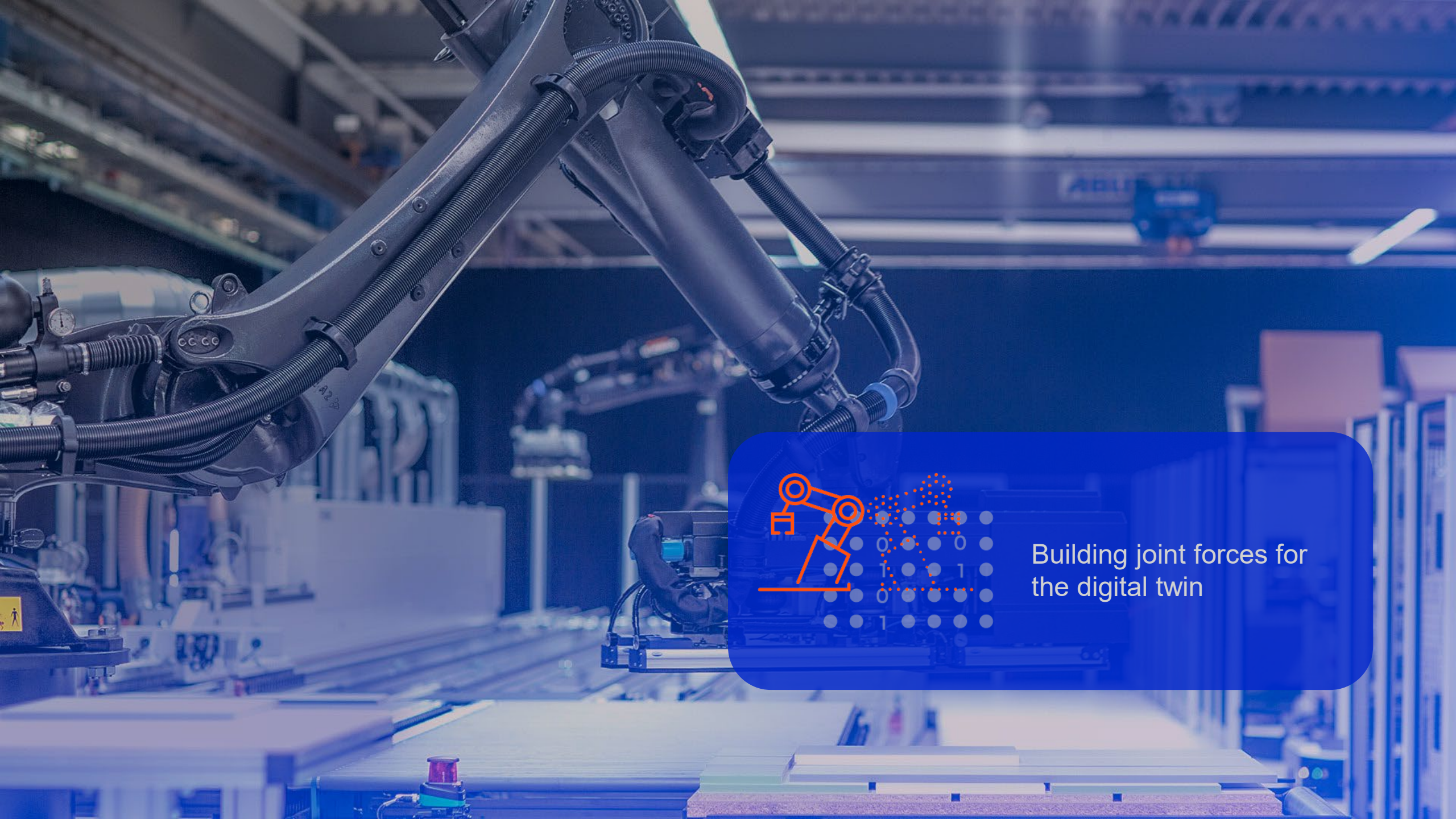
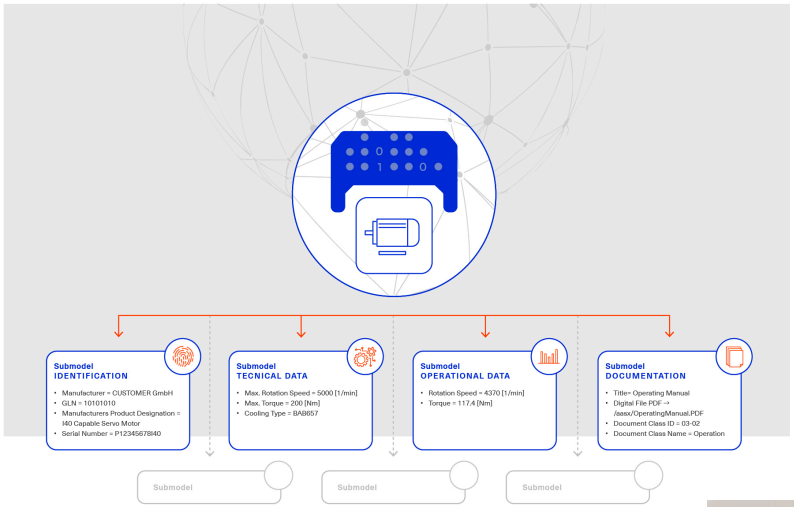
Industrial Digital Twin Association

Tutorial

Specification Asset Administration Shell Part 1 Metamodel V3.0 April 2023

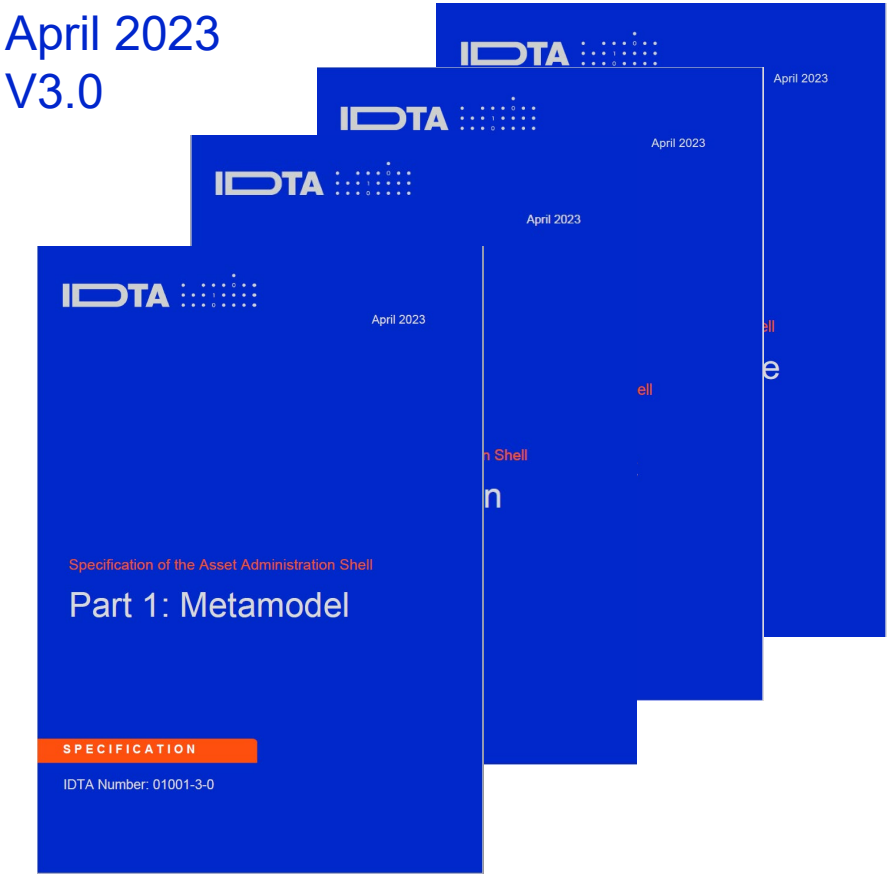

Building joint forces for
the digital twin

From Concept to Specification



2018
V1.0

April 2023
V3.0



Specifications

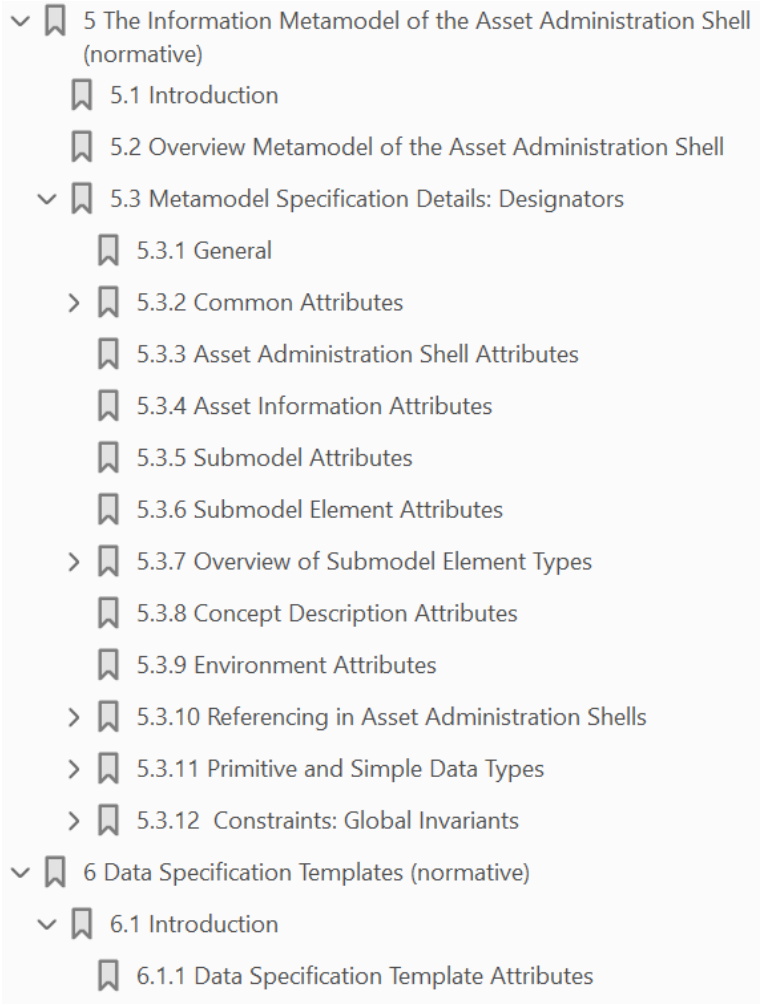
V3.0
April 2023

This Tutorial is about Part 1

IEC 63278

For whom is this tutorial?

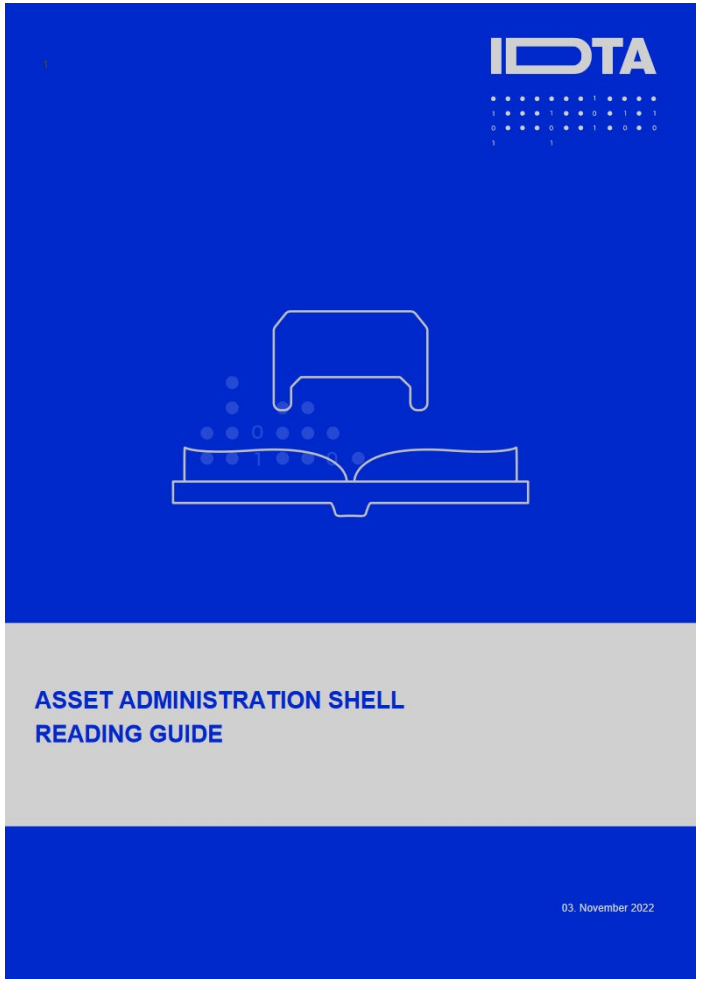
- You should have an idea of the benefits of introducting digital twins to your domain
- You should know about the basic concepts of the Asset Administration Shell
- You should have basic knowledge in UML modeling
- You are an architect and want to learn more about the underlying information model of the Asset Administration Shell
- You are a developer and want to upgrade to the new version or start your first implementation



For whom is this tutorial?

- You feel dissappointed and do not know how to start?
  Have a look at the Asset Administration Shell Reading Guide!
  It is updated on a regular basis.

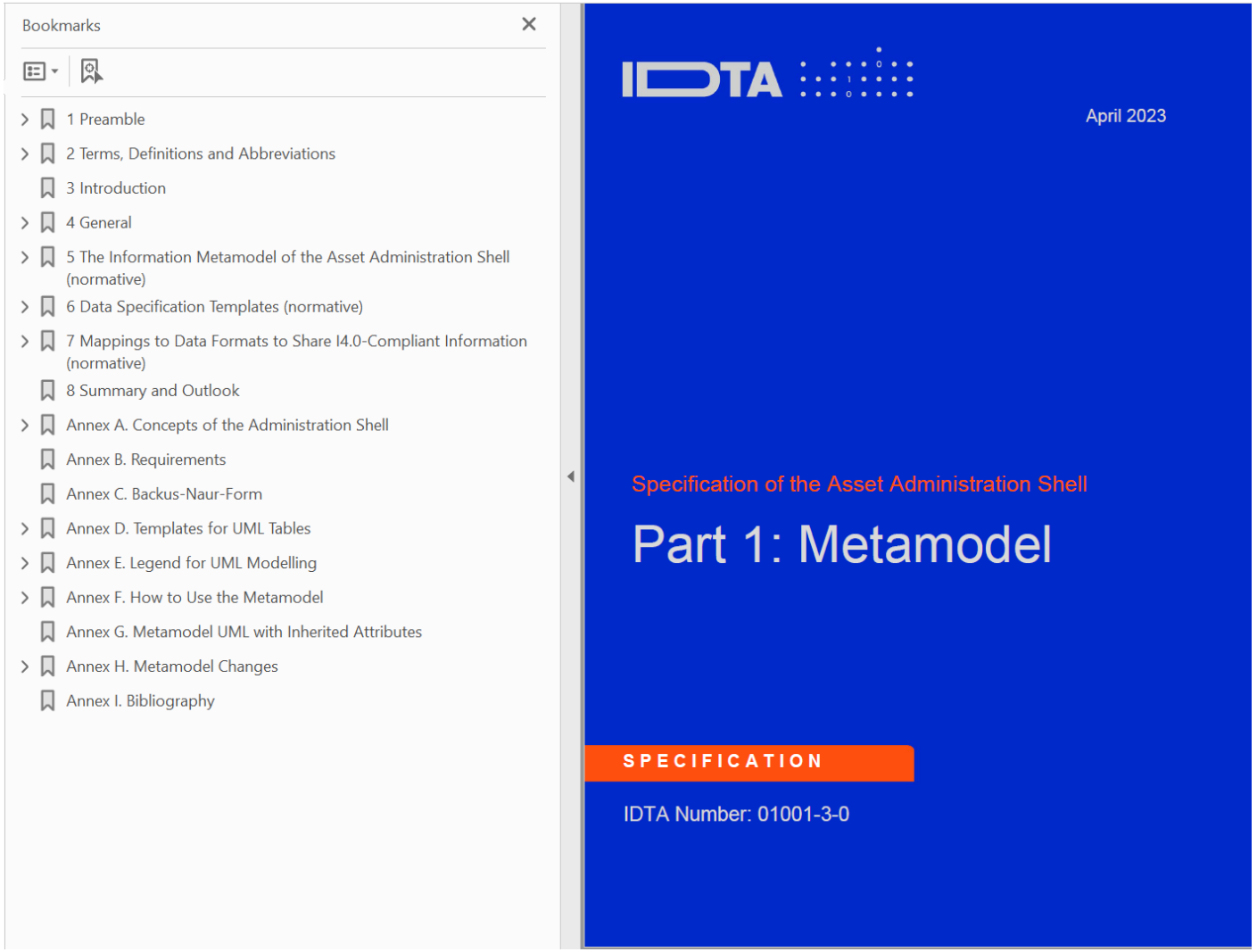
- **Where to start:** If you have never heard of the AAS
- **For the generally interested reader:** If you want to learn more about the subject
- **For decision makers:** If you are interested in the business side of I4.0
- **For software developers and architects:** If you want to know how to create software for the AAS
- **For users of the AAS and domain experts:** If you are interested in using the AAS for specific tasks
- **Security and AI:** If you want to deep dive into these special topics.




Get started

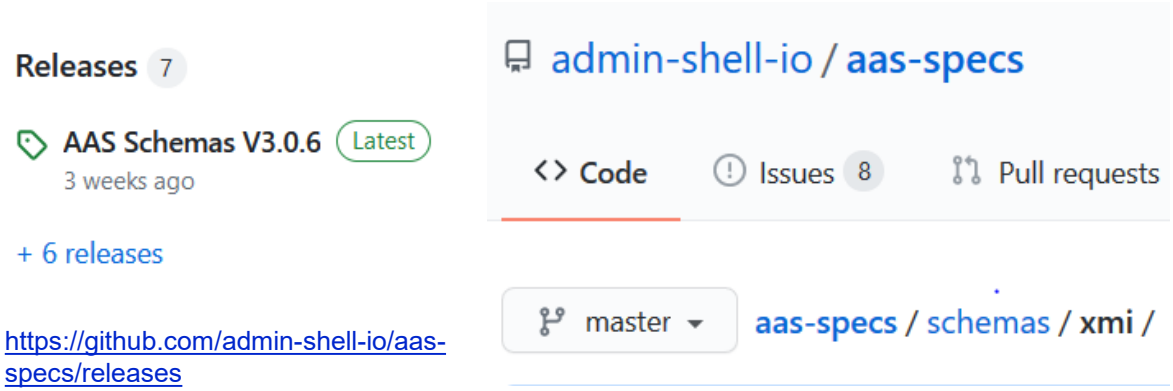
Download Specification

https://industrialdigitaltwin.org/content-hub/



Import XMI to your UML tooling

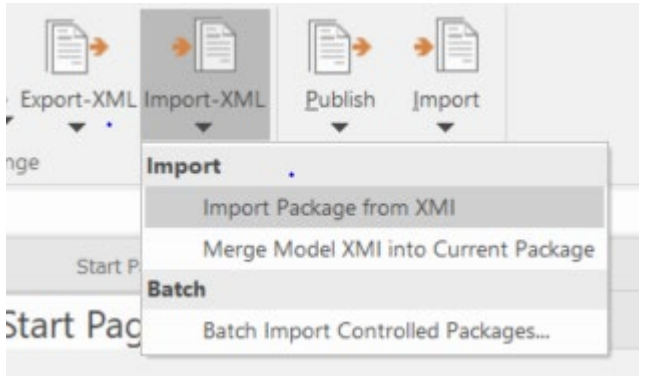
1.



https://github.com/admin-shell-io/aas-specs/releases

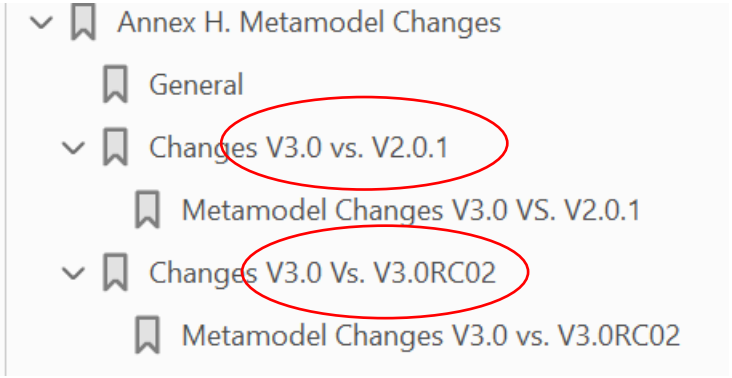
https://github.com/admin-shell-io/aas-specs/tree/master/schemas/xmi

2.



1. Fetch release of AAS you are interested in
2. Import xmi file into UML tool (best with Enterprise/Architect)

Metamodel Changes



Note for Experts: <Notes for tutorial listeners who have knowledge of previous versions of the specification (V2.0 or Release Candidates of V3.0).

If you do not know previous versions you can ignore these notes.>

14 - Birgit Boss: Details of the Asset Administration Shell V3.0 (Release...

V3.0RC02

https://www.youtube.com/watch?v=QR-nOl6cuOI

UML Legend

1. Get (re-)familiar with general UML modeling rules
2. Get familiar with specific graphical representation of UML in the specification (partly tool specific)

Annex E. Legend for UML Modelling

OMG UML General

This annex explains the UML elements used in this specification. For more information, please comprehensive literature available for UML. The formal specification can be found in [35].

Figure 63 shows a class with name "Class1" and an attribute with name "attr" of type *Class2*. A owned by the class. Some of these attributes may represents the end of binary associations, s 70. In this case, the instance of *Class2* is navigable via the instance of the owning class *Class*

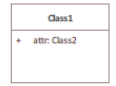


Figure 63 Class

Figure 64 shows that *Class4* inherits all member elements from *Class3*. Or in other word, *Clas* generalization of *Class4*, *Class4* is a specialization of *Class3*. This means that each instance c also an instance of *Class3*. An instance of *Class4* has the attributes *attr1* and *attr2*, whereas in *Class3* only have the attribute *attr1*.

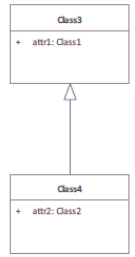
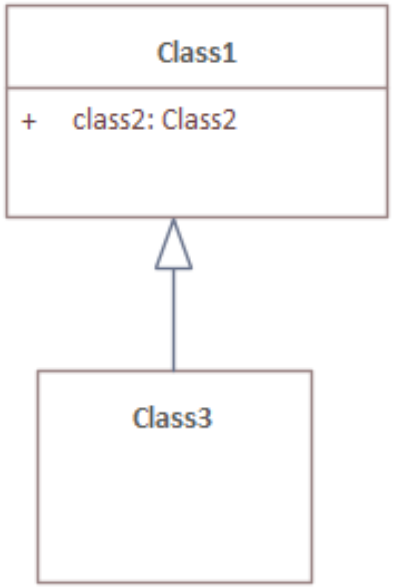


Figure 64 Inheritance/Generalization

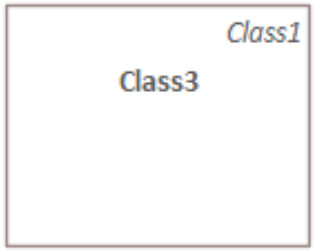
Figure 65 defines the required and allowed multiplicity/cardinality within an association betwee *Class1* and *Class2*. In this example, an instance of *Class2* is always related to exactly one inst *Class1*. An instance of *Class1* is either related to none, one, or more (unlimited, i.e. no constra

UML Generalization/Inheritance

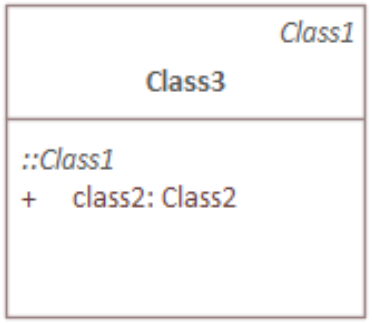
A)



B)



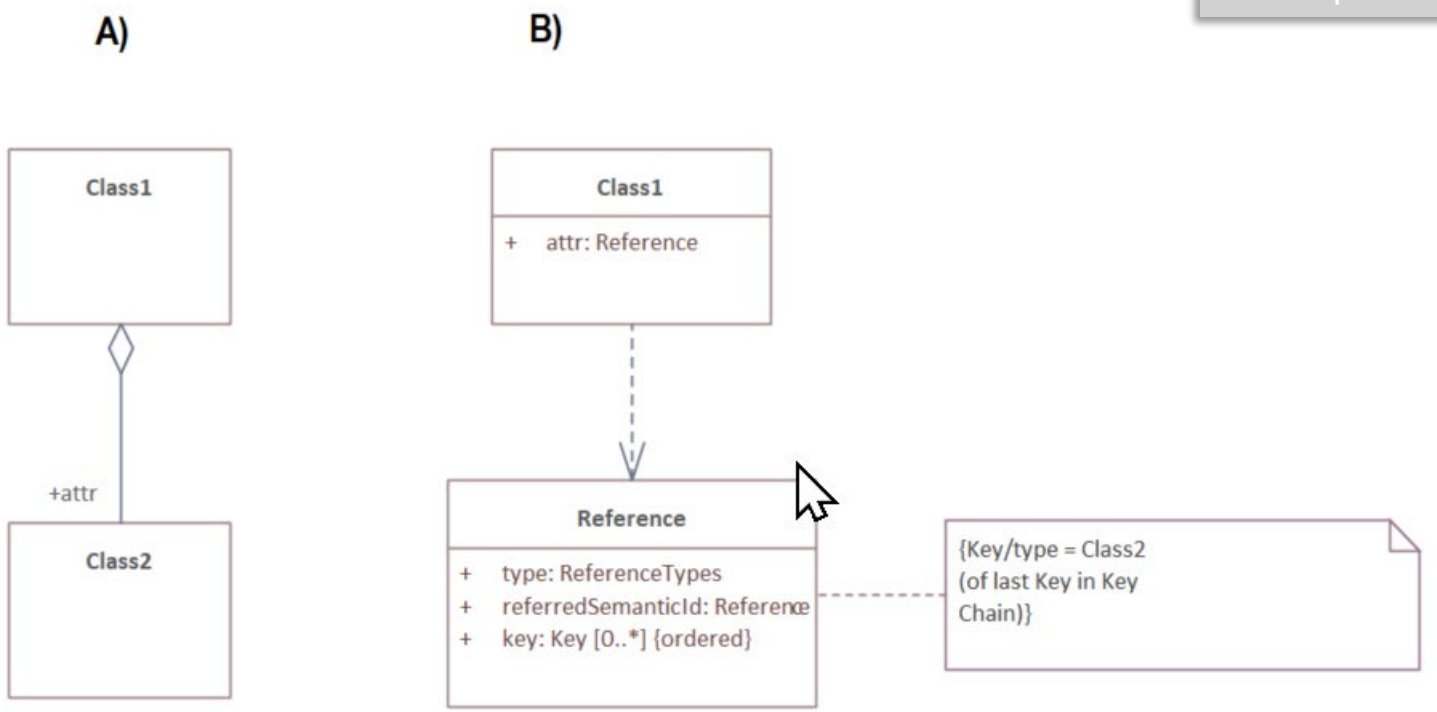
C)



Hint: Graphical representation tool specific

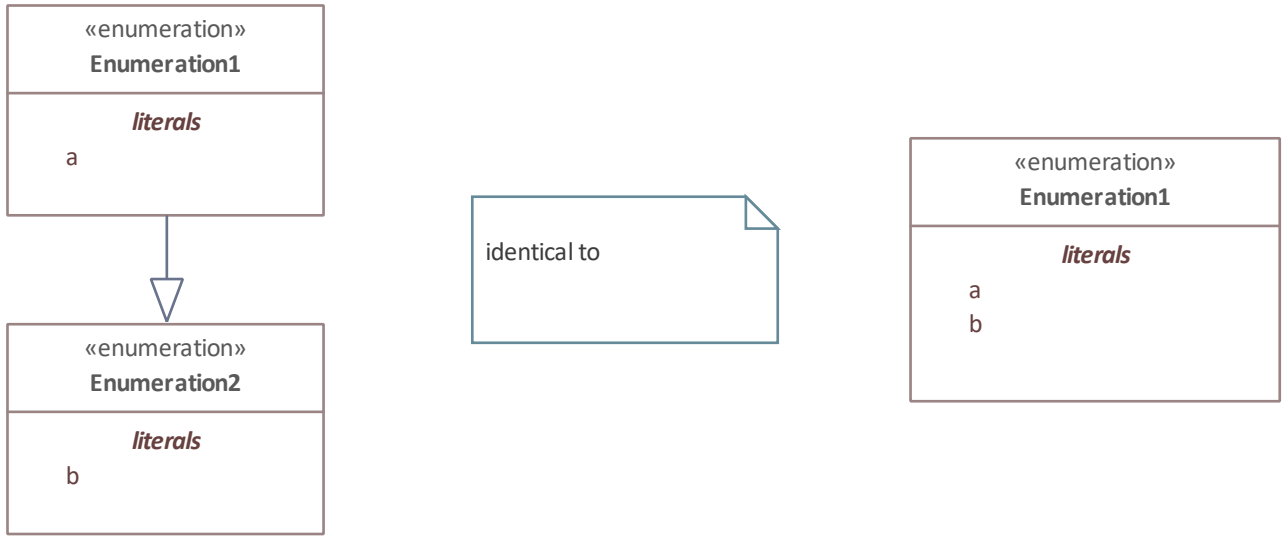
UML Shared Aggregation

Note: Referencing of Referables is an important concept to understand when implementing the AAS



Note for experts: In previous versions a notation of class attributes with reference (*) was used additionally to the notation with the association with the diamond.

Enumerations



Note 1: Inheritance between enumerations is not widely used. It is only used for graphical illustration of relationships between enumerations

Templates

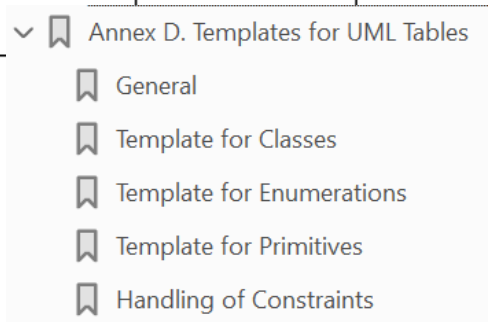
Template for Classes

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>

Note for experts: ModelReference<SubmodelElement> is equal to former notation SubmodelElement*

Note for experts: no kind column any longer, instead different notation for Type



Example for Class Specification

5.3.7.12 Property Attributes

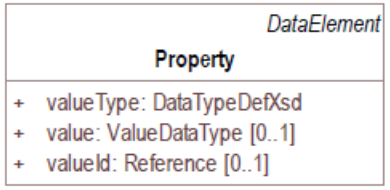
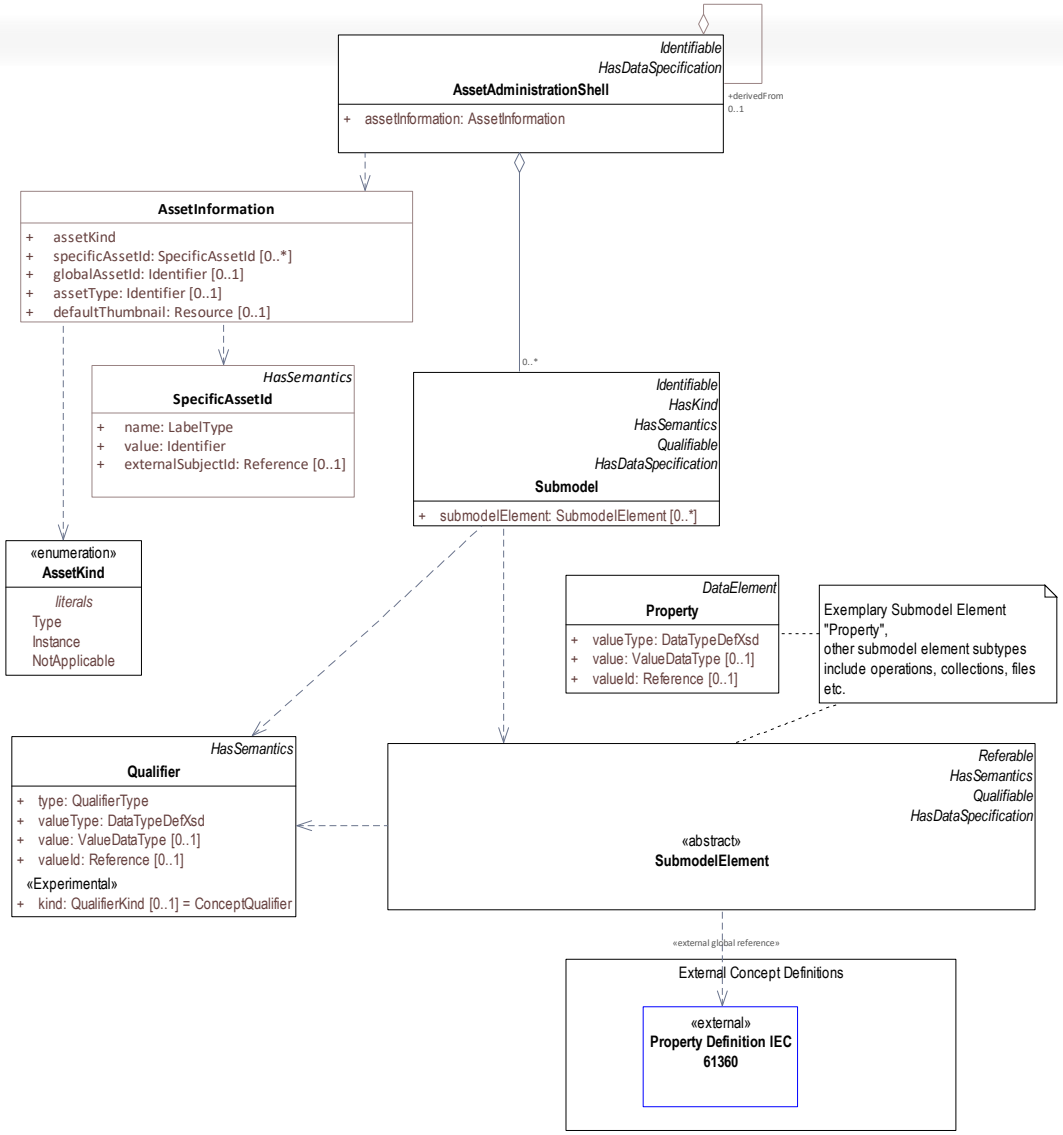


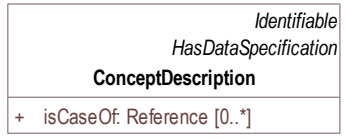
Figure 39 Metamodel of Properties

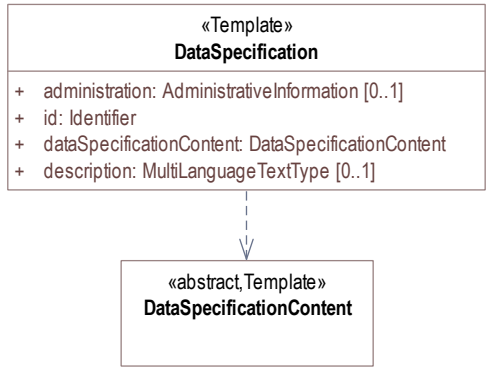
Class:	Property		
Explanation:	A property is a data element that has a single value. Constraint AASd-007: If both the *Property/value* and the *Property/valueId* are present, the value of *Property/value* needs to be identical to the value of the referenced coded value in *Property/valueId*.		
Inherits from:	DataElement		
Attribute	**Explanation**	**Type**	**Card.**
valueType	Data type of the value attribute	DataTypeDefXsd	1
value	The value of the property instance	ValueDataType	0..1
valueId	Reference to the global unique ID of a coded value Note: it is recommended to use an external reference.	Reference	0..1


Get warm

Overview – the goal is to be able to read this diagram





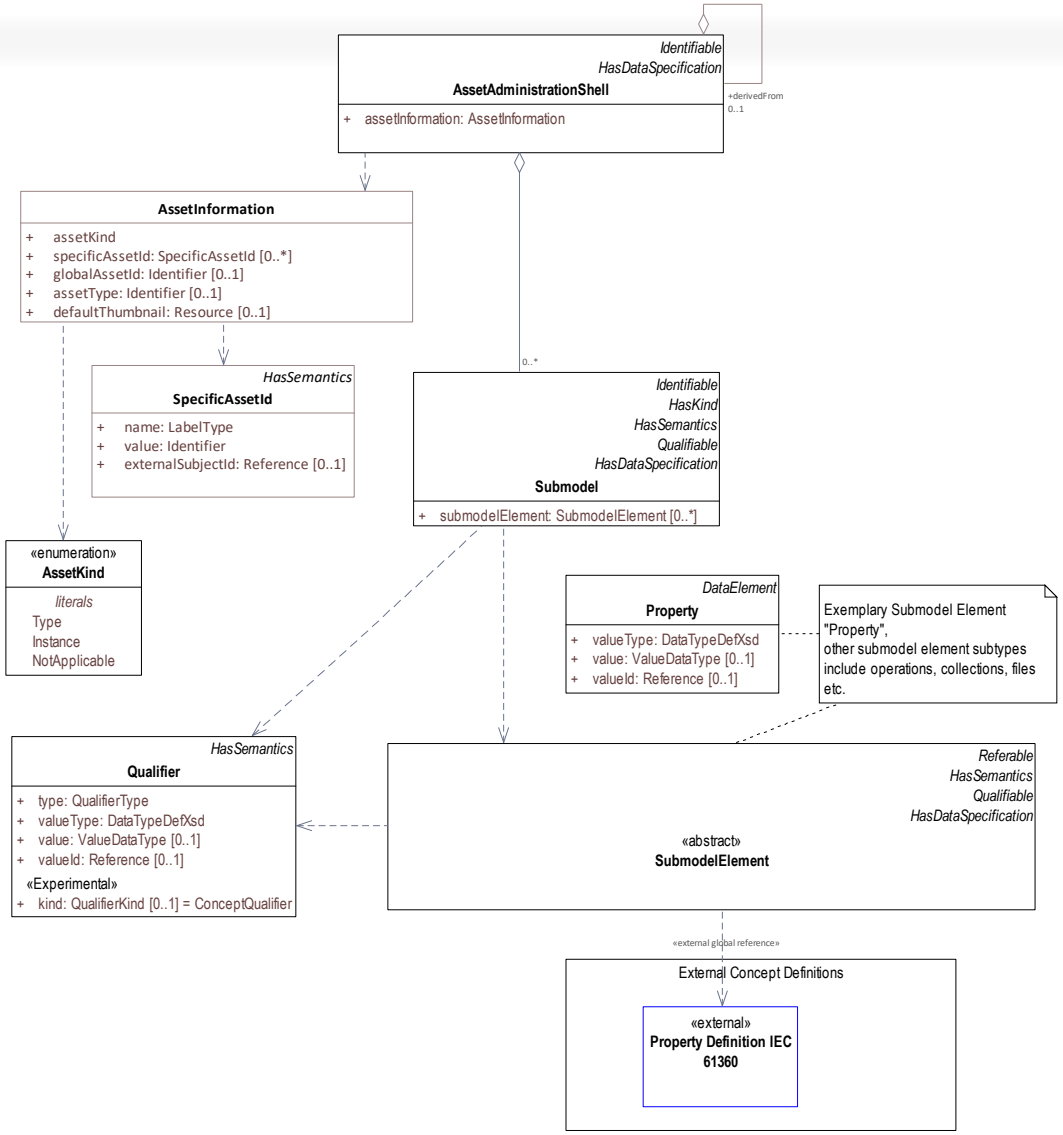


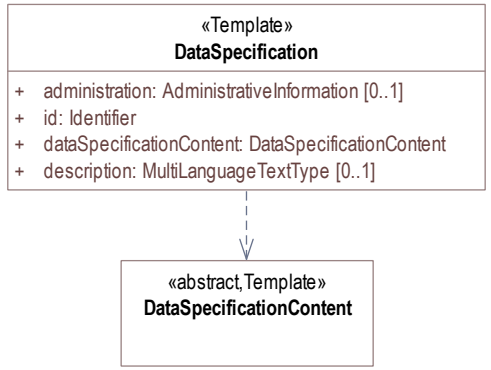
Overview – the goal is to be able to read this diagram

Note for Experts: no views supported any longer

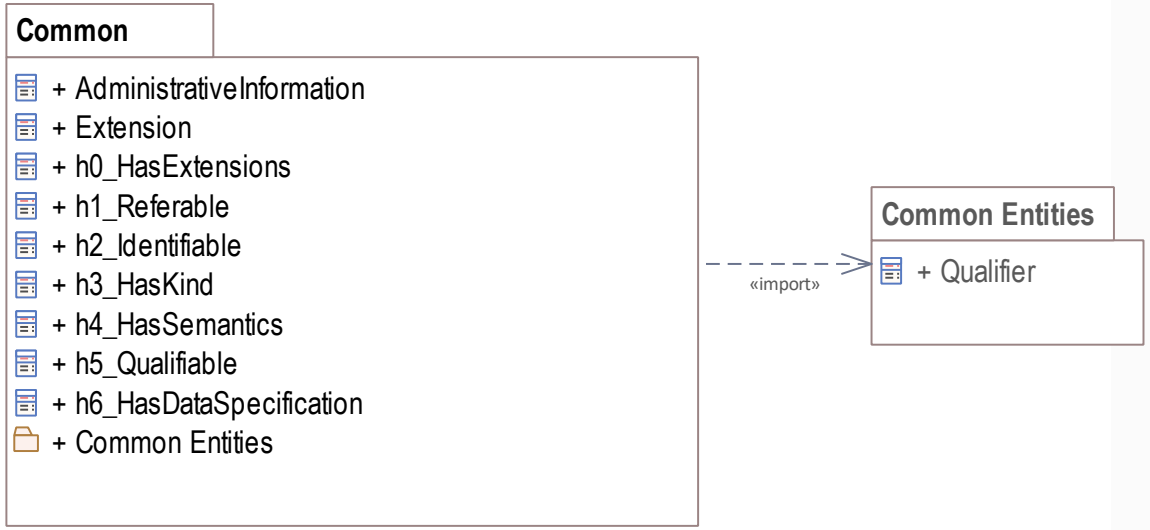
Note for Experts: no assets supported any longer, AssetAdministrationShell /assetInformation introduced instead

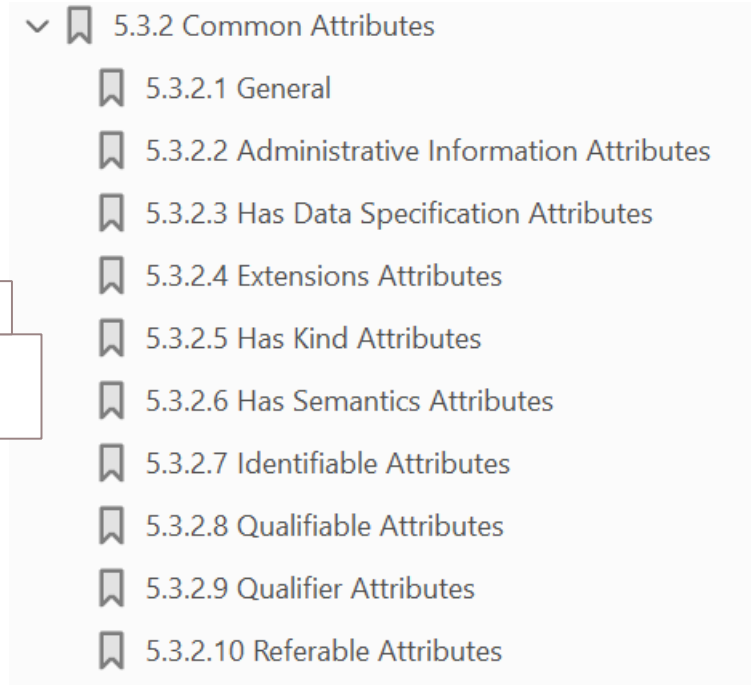
Note for Experts: Security in Part 4





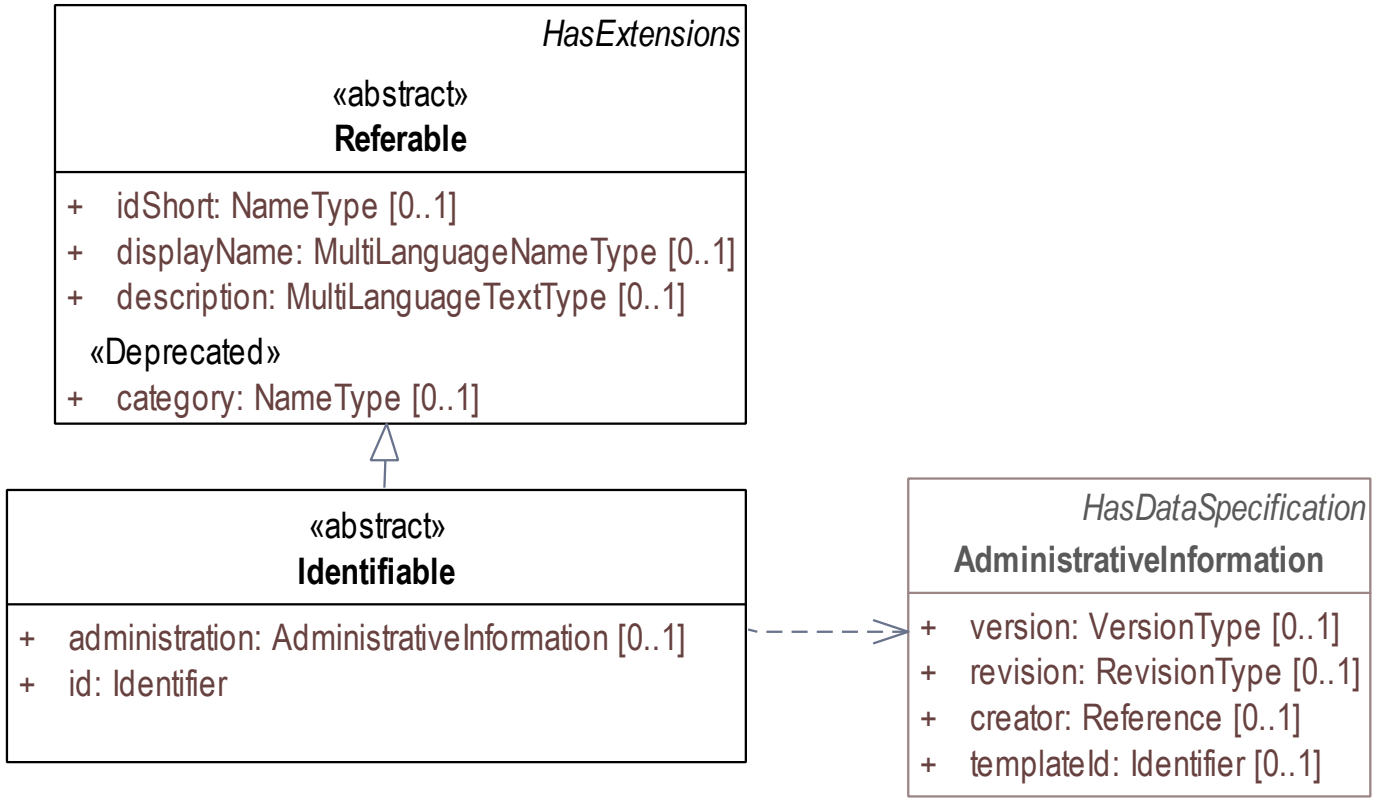
Common





Note: h0_ h1_ are just added for sorting. In diagrams alias are used without this prefix. Only in package overview and inheritance alias are not supported by the UML tooling used.

Common – Identifiables and Referables



Note for Experts 3.0: checksum handling in discussion, not part of model 3.0.

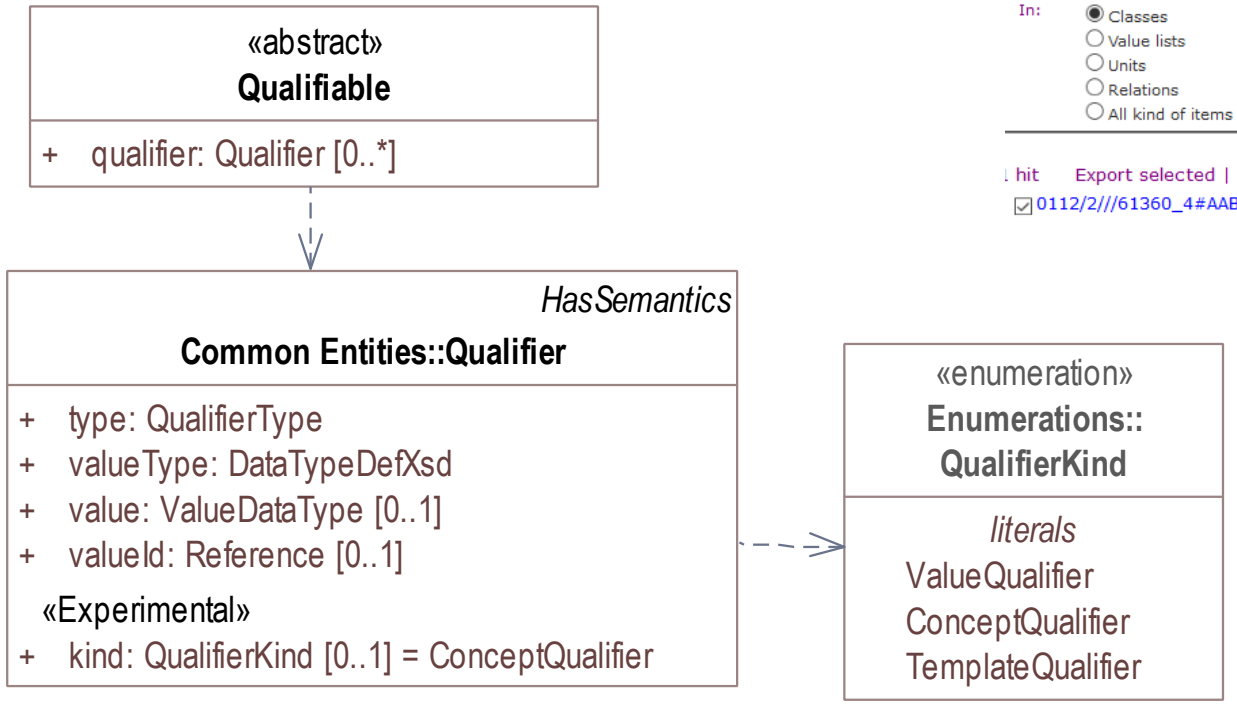
Note for Experts: Identifier in previous versions of the specification had two attributes: the ID itself and the ID type (IRI, IRDI, Custom). The ID type was removed from the model.

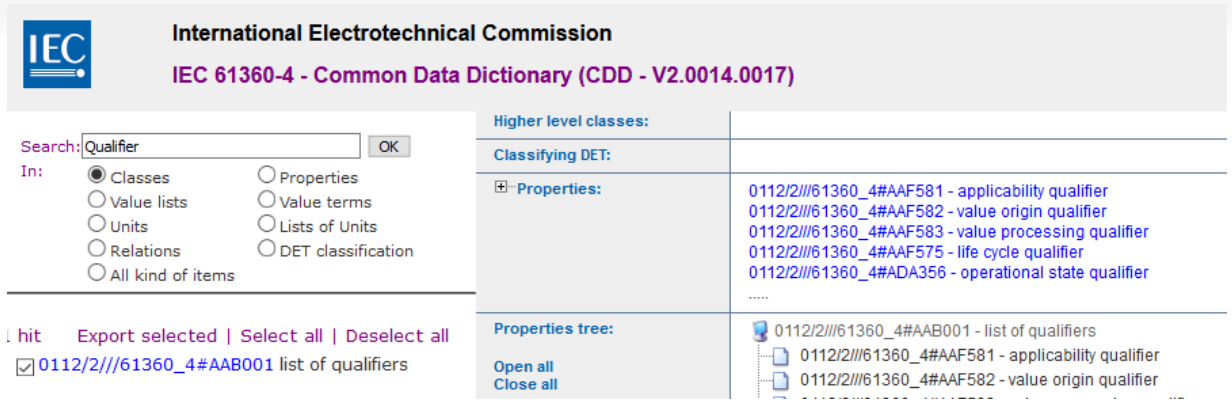
Note for Experts: idShort now optional but still required for non-identifiable referables. DisplayName introduced.

Note for Experts: category of referables set to deprecated.

Note for Experts: New attributes in administrative information: creator and template ID

Common - Qualifiables

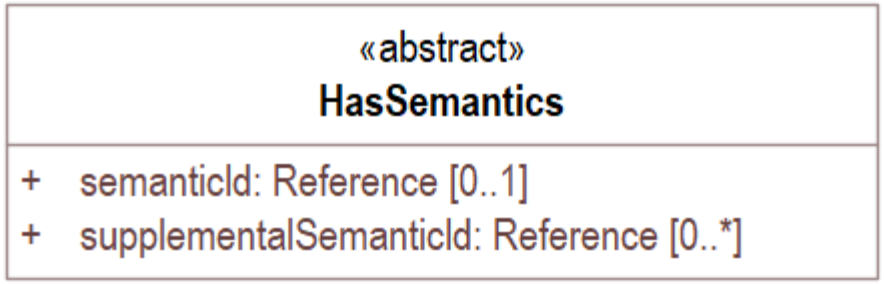




Note for Experts: No Formulas (or other Constraints) supported for Qualifiables any longer,

Note for Experts: Expertimental Qualifier kind introduced

Common - HasSemantics



The semanticId is the identifier of the semantic definition of the element.
Supplemental semantic IDs can be added.

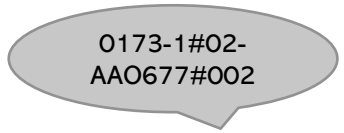
Note for Experts: Besides the semanticId supplemental semantic IDs are now possible to be added.

Note for Experts: semantic ID now optional but recommended

Matching Strategies for semantic IDs

Note: isCaseOf of concept descriptions and supplemental semantic IDs are not yet considered in the defined matching strategies.

Exact Matching (identicals semantic Ids) – DEFAULT

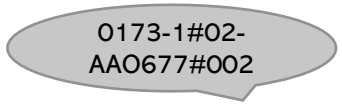


ManufacturerName
semanticId =
0173-1#02-AAO677#002

Herstellername
semanticId =
0173-1#02-AAO677#002

Intelligent Matching (compatible semantic Ids)

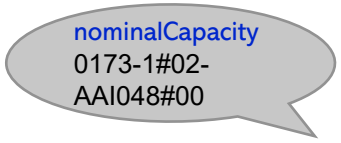
Ignore Versioning



ManufacturerName
semanticId =
0173-1#02-AAO677#002

Herstellername
semanticId =
0173-1#02-AAO677#003

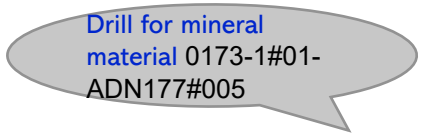
Consider Semantic Mappings



nominalCapacity
semanticId =
0112/2///61360_4#AAE530

nominalCapacity
semanticId =
0173-1#02-AAI048#00

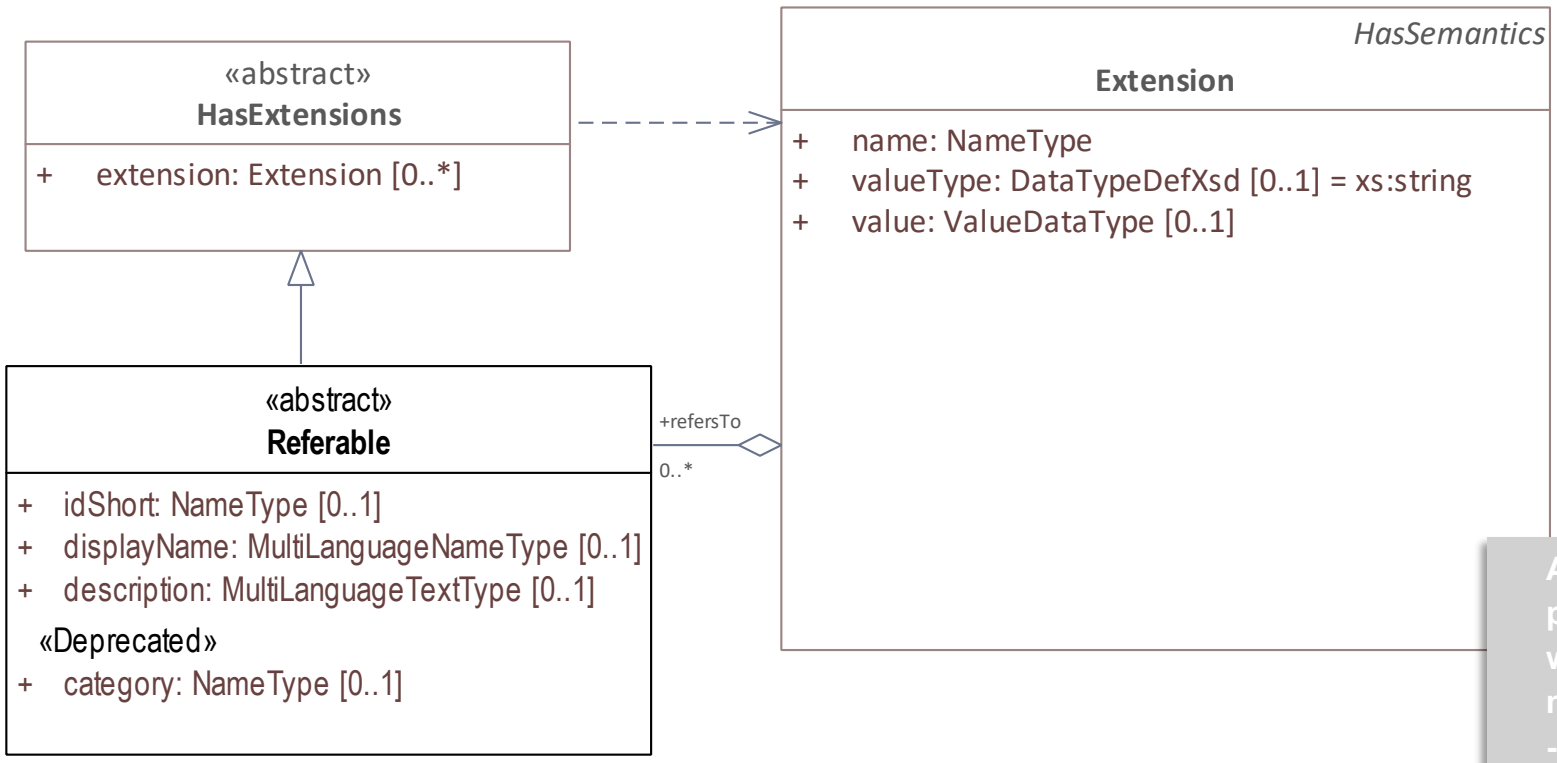
Consider Domain Knowledge



HammerDrill
semanticId =
0173-1#01-ADS698#010

PercussionDrill
semanticId =
0173-1#01-ADN177#005

Common - HasExtensions



Note for Experts 3.0:
refersTo now only supports references to Referables, not to external sources.

Note for Experts 2.0:
proprietary extensions now supported

Allows to annotate an object with proprietary add-ons (extensions) without need to (wait for) update the metamodel

-

<u>Be aware:</u> extensions do not support interoperability!

Common - HasDataSpecification

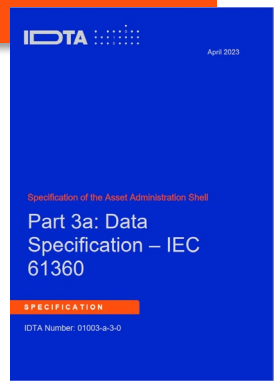
«abstract» **HasDataSpecification**
+ dataSpecification: Reference [0..*]

Allows to define standardized templates for data specification

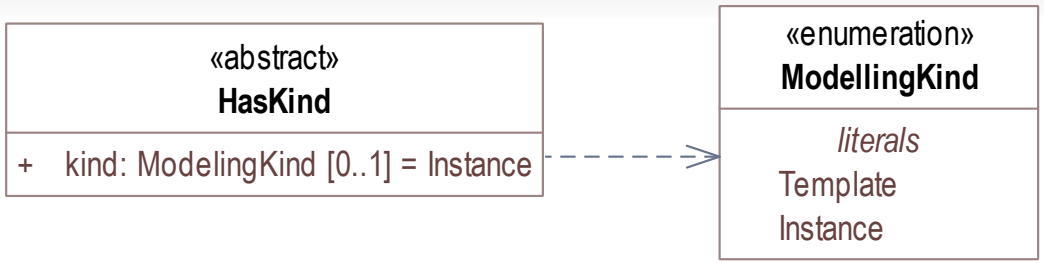
Data specification reference shall be globally unique and identifies which data specifications are used for an object

Attributes defined in template are added to the object

Note for Experts: Data Specifications are not part of Part 1 any longer: They are part of data specifications series Part 3



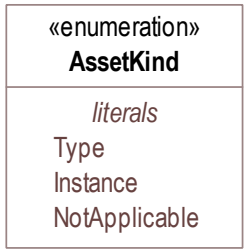
Common - HasKind



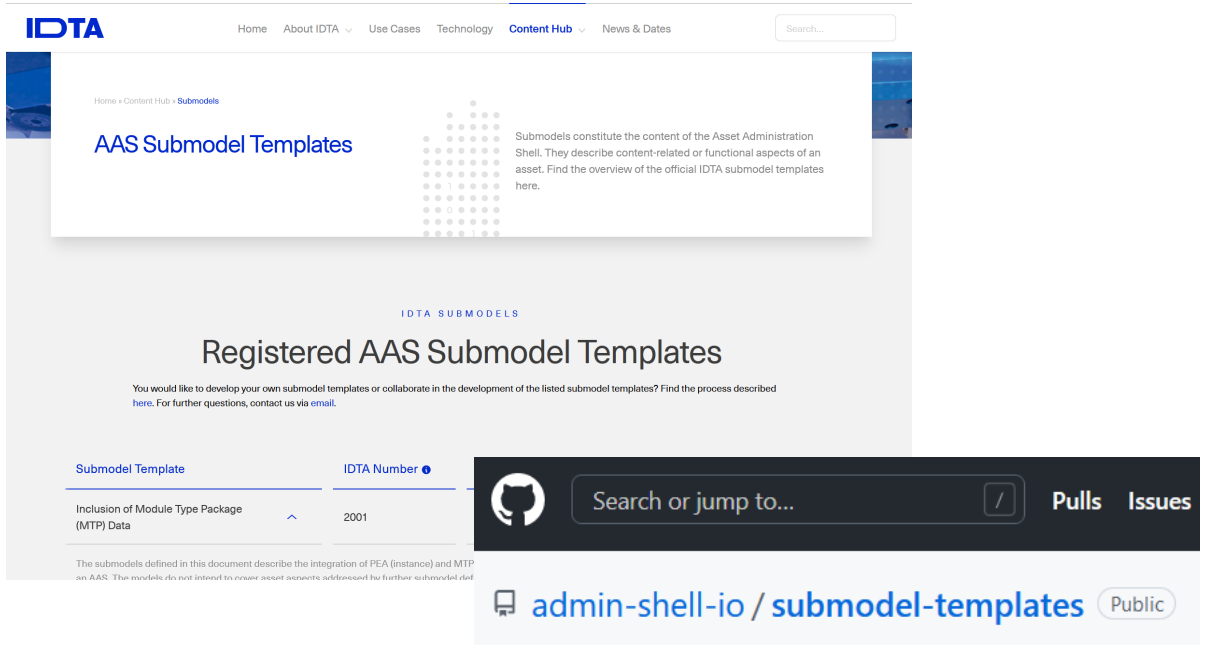
ModelingKind = Instance provides values for all data elements within a submodel

ModelingKind = Template is used to specify Submodel Templates, e.g. for the Digital Nameplate or Technical Data

Note for Experts: Only Submodels have the kind attribute, submodel elements do implicitly have the same modelling kind.



Note: Do not mix up with AssetKind.
AssetKind reflects the time in the life cycle of a product, e.g. in Engineering phase it is a product type, in production it is an product instance that is produced.

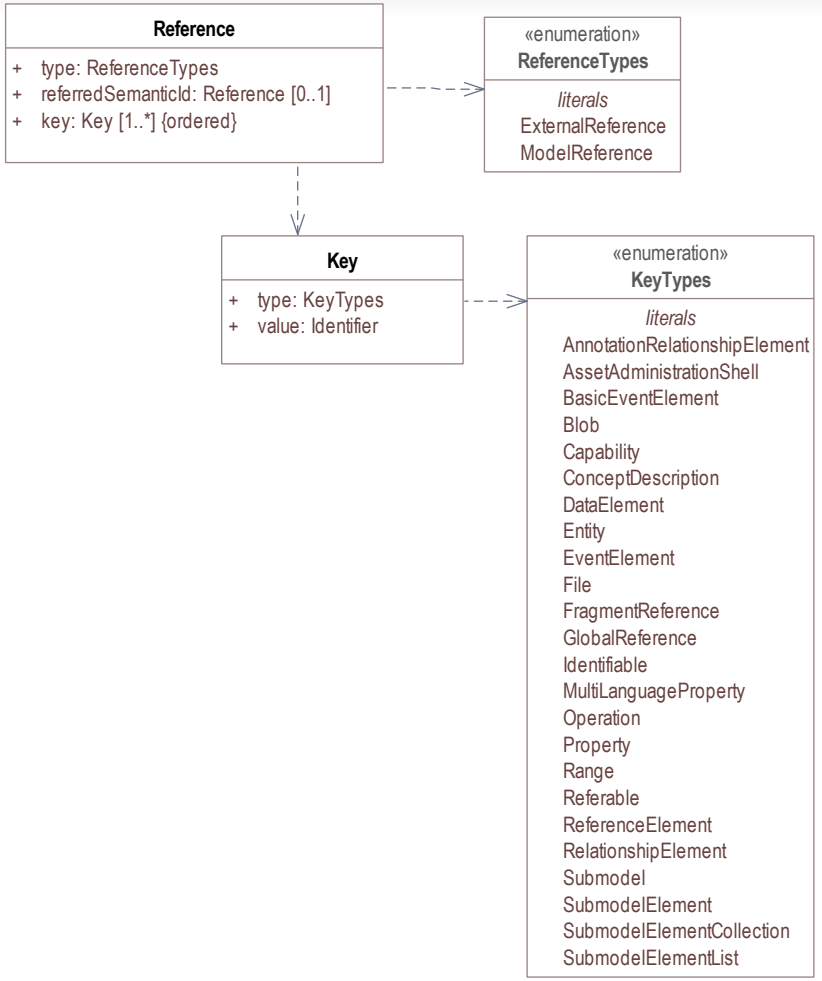


Referencing

External Global References (e.g. to ECLASS IRDI, manfuacturer Web-Site)

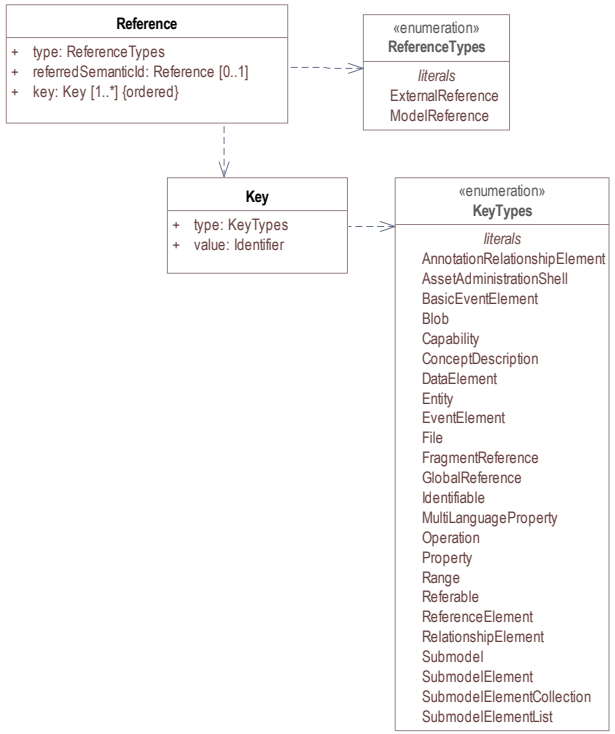
References into a File (Fragment)

Model References to any Referable in an AAS model (e.g. to define relationships between elements)



Note for Experts: Reference Concept updated

Referencing - Examples



7.2.3 Serialization of Values of Type "Reference"

Some mappings or serializations convert the type "Reference" into a single string. In this case, the following serialization is required:

Grammar:

```
<Reference> ::= ["[" <ReferenceType> [ "- " <referredSemanticId> " -" ]   "]" ]
<Key> {(", "  <Key> }*

<ReferenceType> ::= "ExternalRef" | "ModelRef"    value of AAS:Reference/type

<SemanticId> ::= ["[" <ReferenceType> "]"]  <Key> {(", "  <Key> }*
                            value of AAS:Reference/referredSemanticId

<Key> ::= "(" <KeyType> ")" <KeyValue>

<KeyType> ::= value of AAS:Key/type

<KeyValue> ::= value of AAS:Key/value
```

Note 1: an IRI may also contain special symbols like "(", "," and "[". A blank is added before the new key or value to distinguish beginning and end of a new key.

Note 2: *ReferenceType* is optional. It is clear from the first key in the key chain whether the reference is a global or a model reference. The examples in this document therefore do not use this prefix.

Valid Examples:

References:

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

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

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

Model References:

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

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

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

[ModelRef- (ConceptDescription)0173-1#02-BAA120#008
-](Submodel)https://example.com/aas/1/1/1234859590
```


i700E70
Serial number
231231
Inverter current
0.02
Now dive in

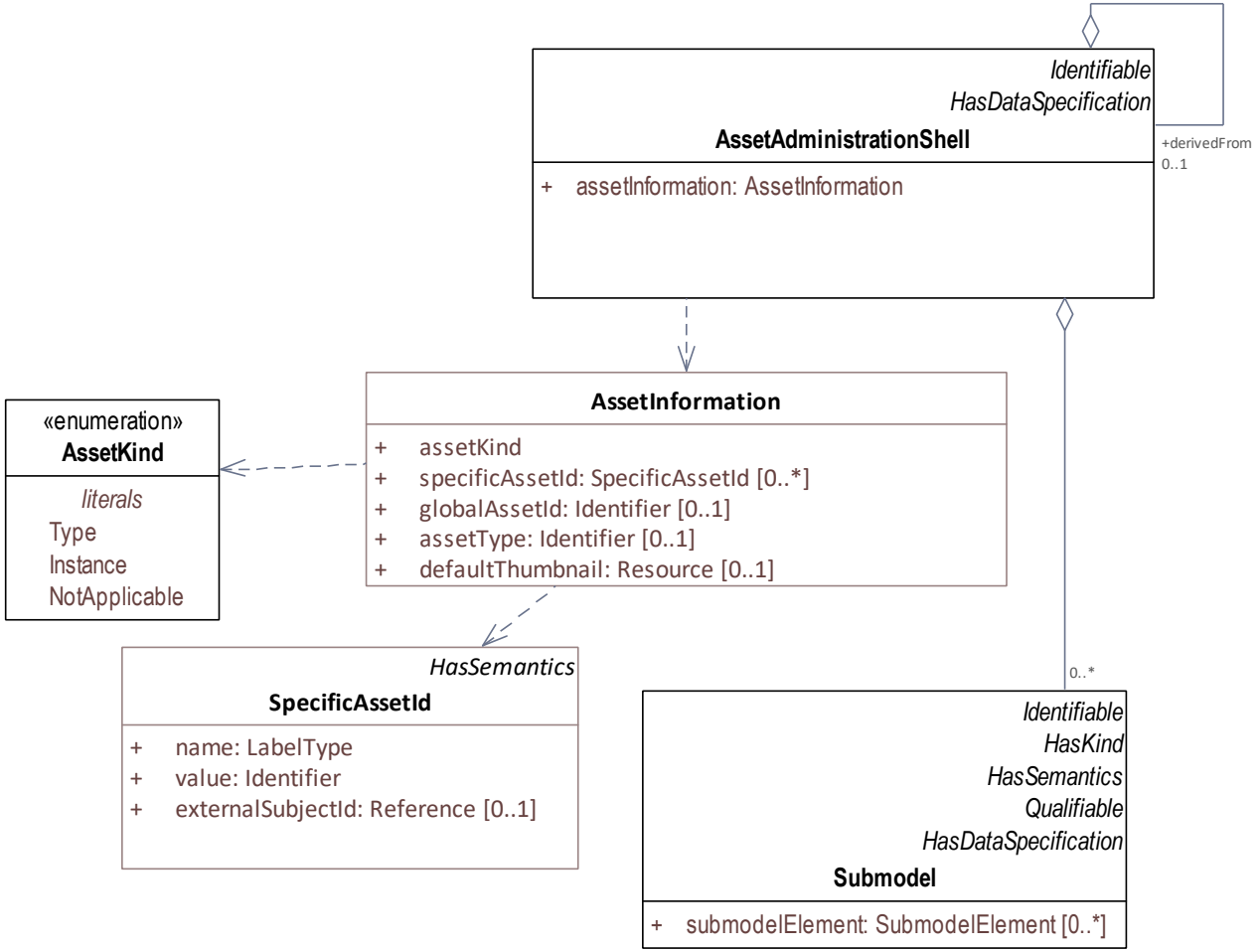
The Asset Administration Shell

Note for Experts: Security and Asset Administration Shell now loosely coupled only

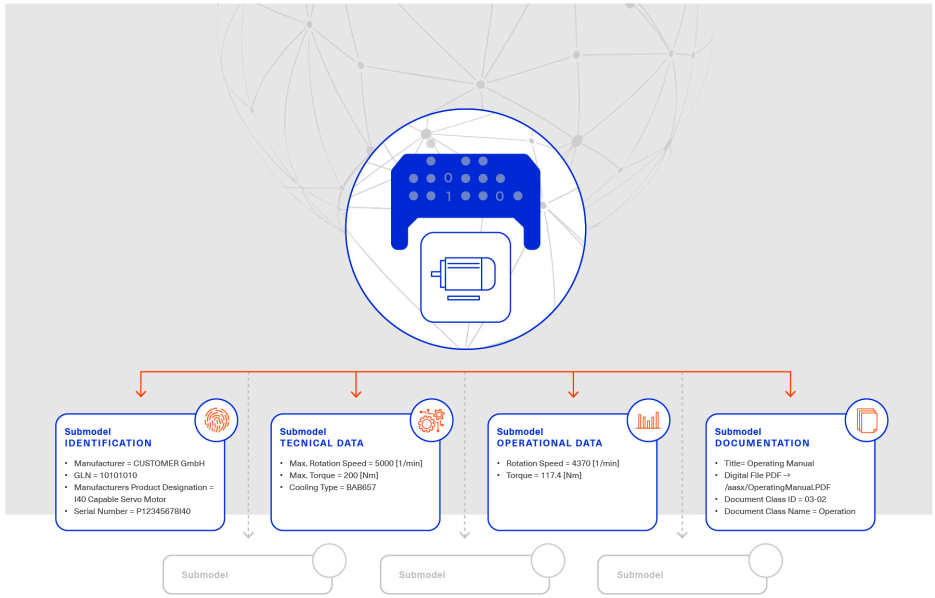
Note for Experts: Updated asset handling

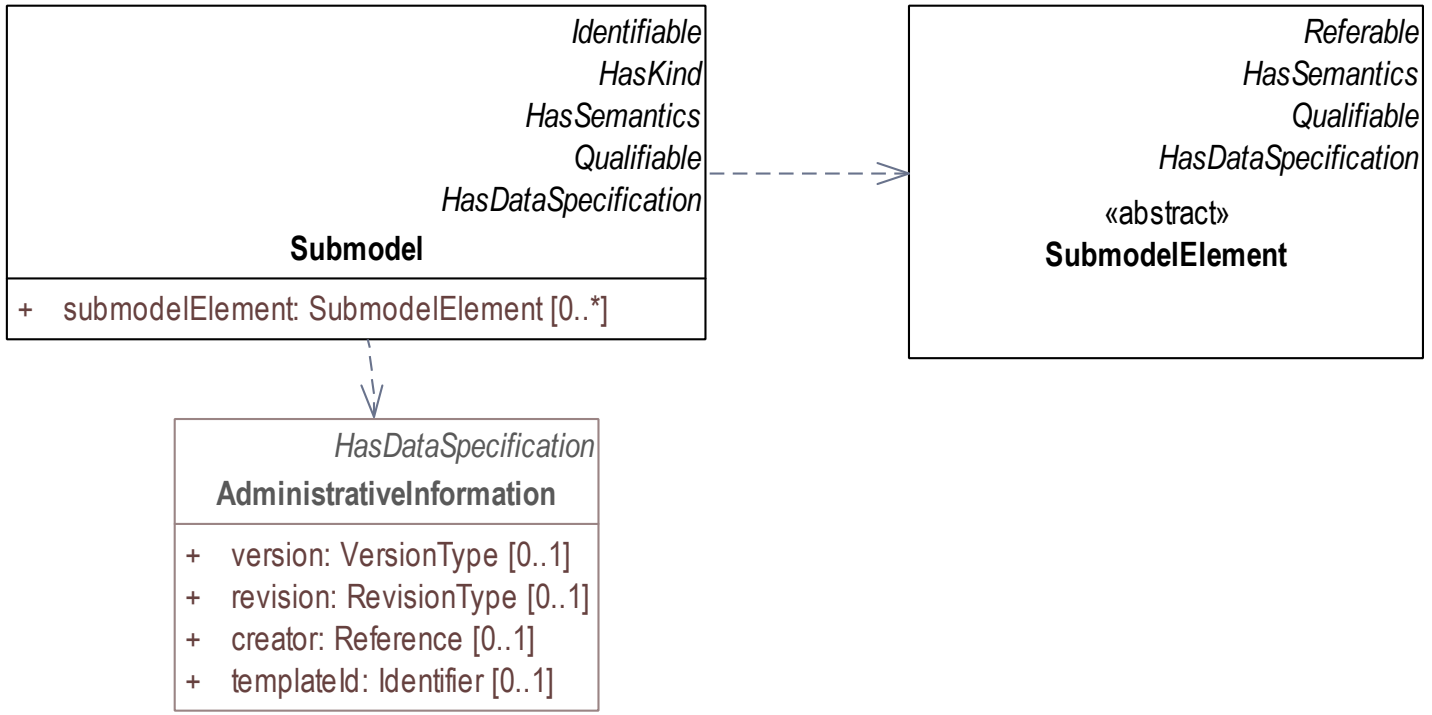
Note for Experts: new attribute „assetType“ for product type – product instance relationship

Note for Experts: New value NotApplicable for asset kind



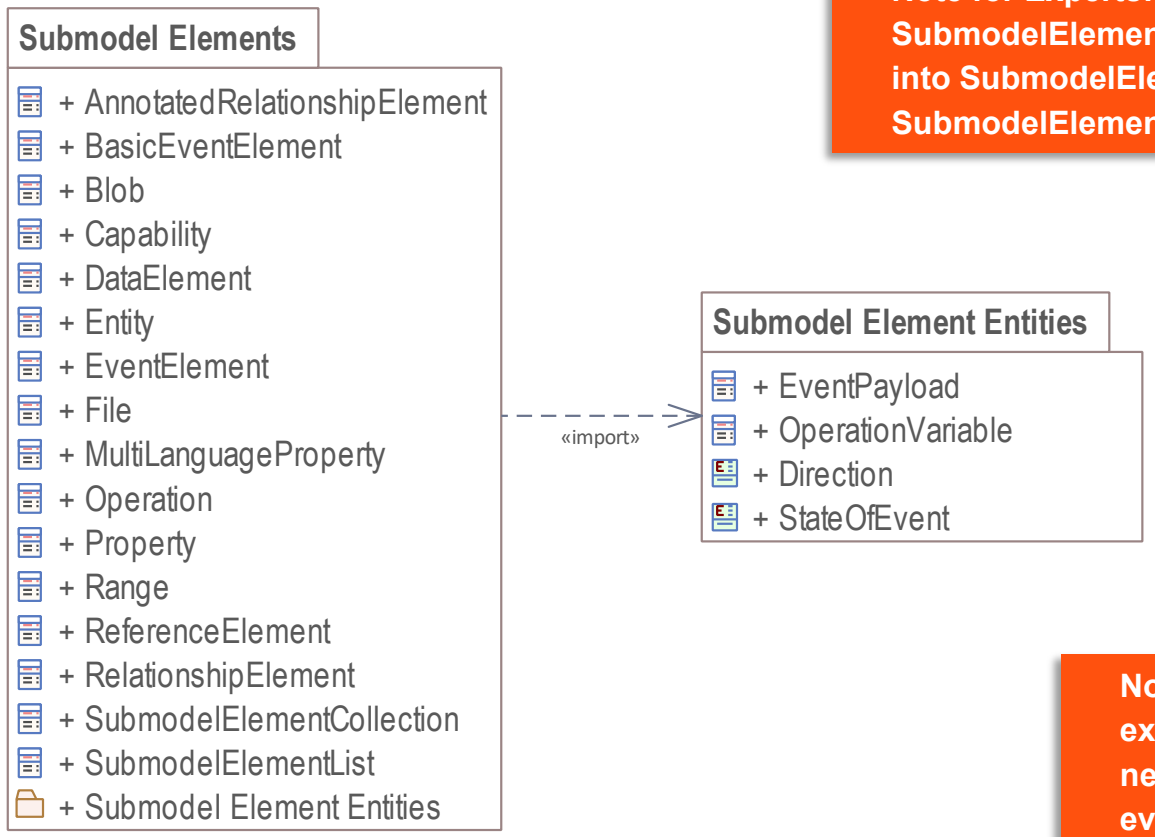
Submodel


Submodel
IDENTIFICATION
• Manufacturer = CUSTOMER GmbH
• GLN = 10101010
• Manufacturers Product Designation = I40 Capable Servo Motor
• Serial Number = P1234567840
Submodel
TECNICAL DATA
• Max. Rotation Speed = 5000 [1/min]
• Max. Torque = 200 [Nm]
• Cooling Type = BAB857
Submodel
OPERATIONAL DATA
• Rotation Speed = 4370 [1/min]
• Torque = 117,4 [Nm]
Submodel
DOCUMENTATION
• Title= Operating Manual
• Digital File PDF --> /aasx/OperatingManual.PDF
• Document Class ID = 03-02
• Document Class Name = Operation
Submodel
Submodel
Submodel


Identifiable
HasKind
HasSemantics
Qualifiable
HasDataSpecification
Submodel
+ submodelElement: SubmodelElement [0..*]
Referable
HasSemantics
Qualifiable
HasDataSpecification
«abstract»
SubmodelElement
HasDataSpecification
AdministrativeInformation
+ version: VersionType [0..1]
+ revision: RevisionType [0..1]
+ creator: Reference [0..1]
+ templateId: Identifier [0..1]

Overview – Package Submodel Elements

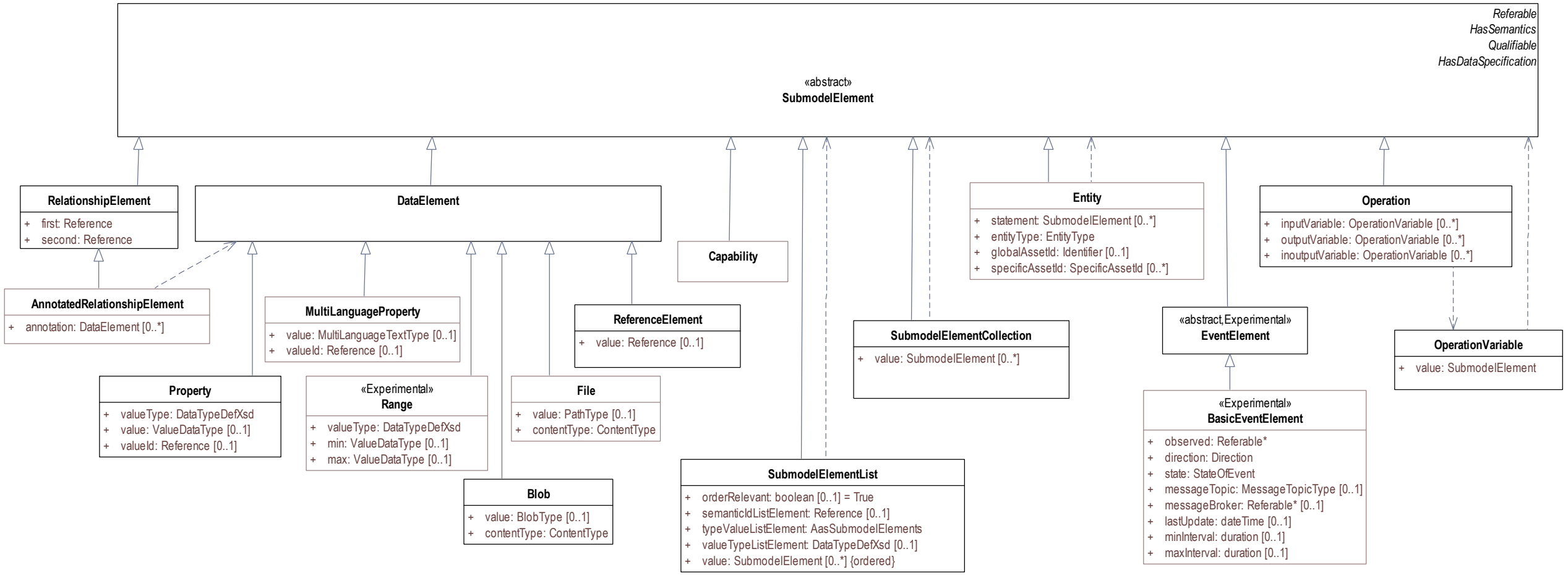
- 5.3.7 Overview of Submodel Element Types
  - 5.3.7.1 General
  - 5.3.7.2 Annotated Relationship Element Attributes
  - 5.3.7.3 Basic Event Element Attributes
  - 5.3.7.4 Blob Attributes
  - 5.3.7.5 Capability Attributes
  - 5.3.7.6 Data Element and Overview of Data Element Types
  - 5.3.7.7 Entity Attributes
  - 5.3.7.8 Event Attributes
  - 5.3.7.9 File Attributes
  - 5.3.7.10 Multi Language Property Attributes
  - 5.3.7.11 Operation Attributes
  - 5.3.7.12 Property Attributes
  - 5.3.7.13 Range Attributes
  - 5.3.7.14 Reference Element Attributes
  - 5.3.7.15 Relationship Element Attributes
  - 5.3.7.16 Submodel Element Collection Attributes
  - 5.3.7.17 Submodel Element List Attributes



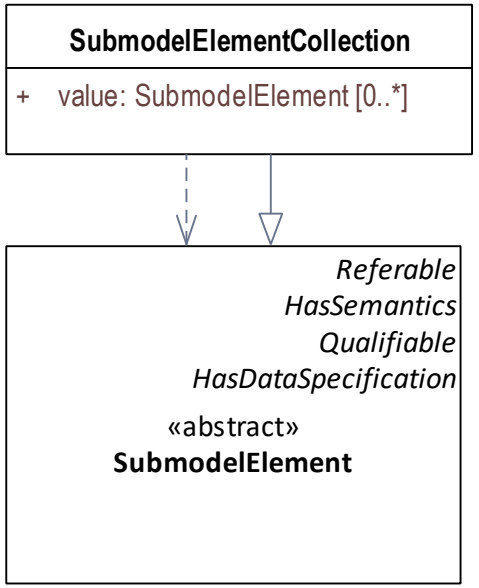
Note for Experts:
SubmodelElementCollection splitted into SubmodelElementCollection and SubmodelElementList

Note for Experts:
experimental new/updated elements for events

Submodel Element Subtypes



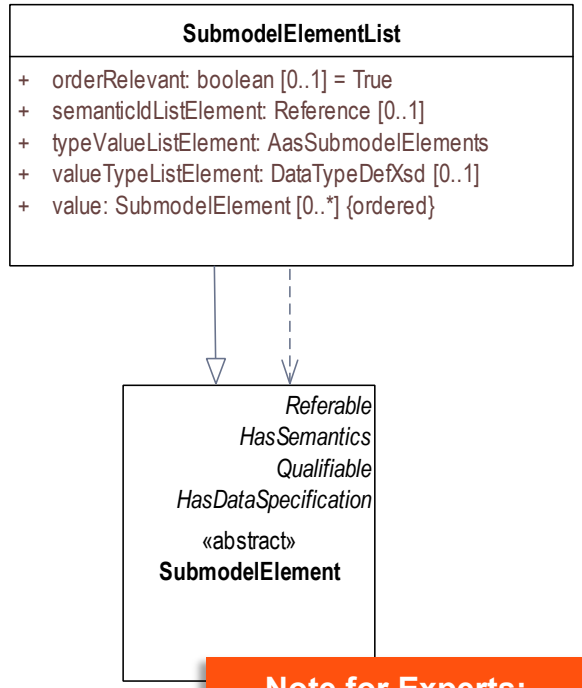
Submodel Element Collections and Lists




```
{
    "NamesOfFamilyMembers": {
        "NameOfMother": "Martha ExampleFamily",
        "NameOfFather": "Jonathan ExampleFamily",
        "NameOfSon": "Clark ExampleFamily"
    }
}
```

Difference in serialization in ValueOnly Format

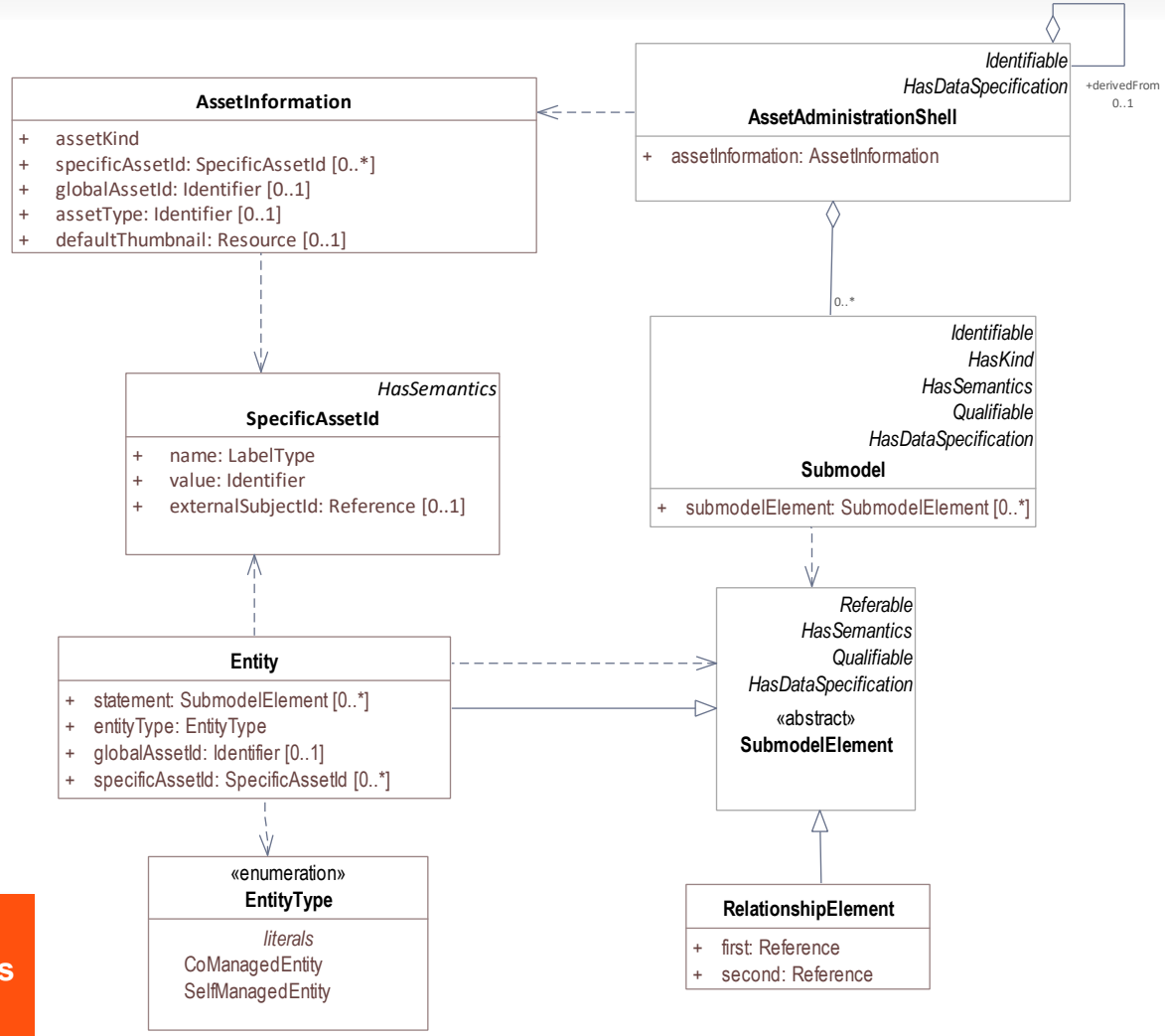
```
{
    "NamesOfFamilyMembers": [
        "Martha ExampleFamily",
        "Jonathan ExampleFamily",
        "Clark ExampleFamily"
    ]
}
```



Note for Experts: Elements in submodel elements list do not have an idShort but are numbered

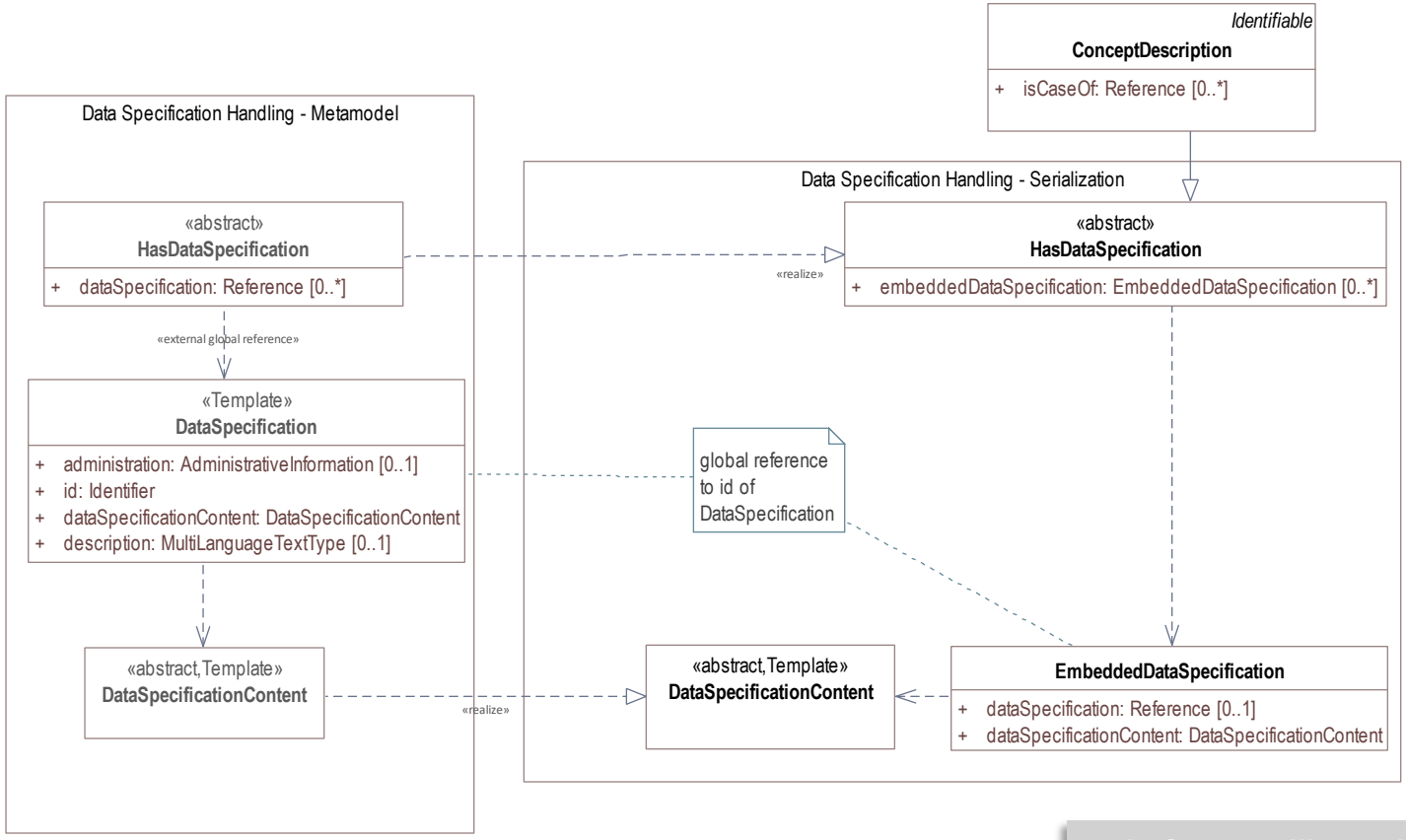
Composite I4.0 Components

- There is no composite Asset Administration Shell, only a composite asset
- Add a bill of material submodel to the AAS of the composite asset: this submodel contains asset IDs to identify its parts
- Via the asset IDs of its parts the AASs of the parts can be found – in case of self-managed entities
- In case of co-managed entity the part is described in the AAS of the composite asset itself
- Any kind of relationship between parts of the composite asset can be expressed



Note for Experts: no bill of materials any longer as part of assetInformation

Embedded Data Specifications



Property	0173-1#02-BAA120#008 Max. rotation speed
Data type	INTEGER_MEASURE
Unit of measure	1/min
Definition	Greatest possible rotation speed with wich the motor or feeding unit may be operated

DataSpecificationContent

«Template»

DataSpecificationIec61360

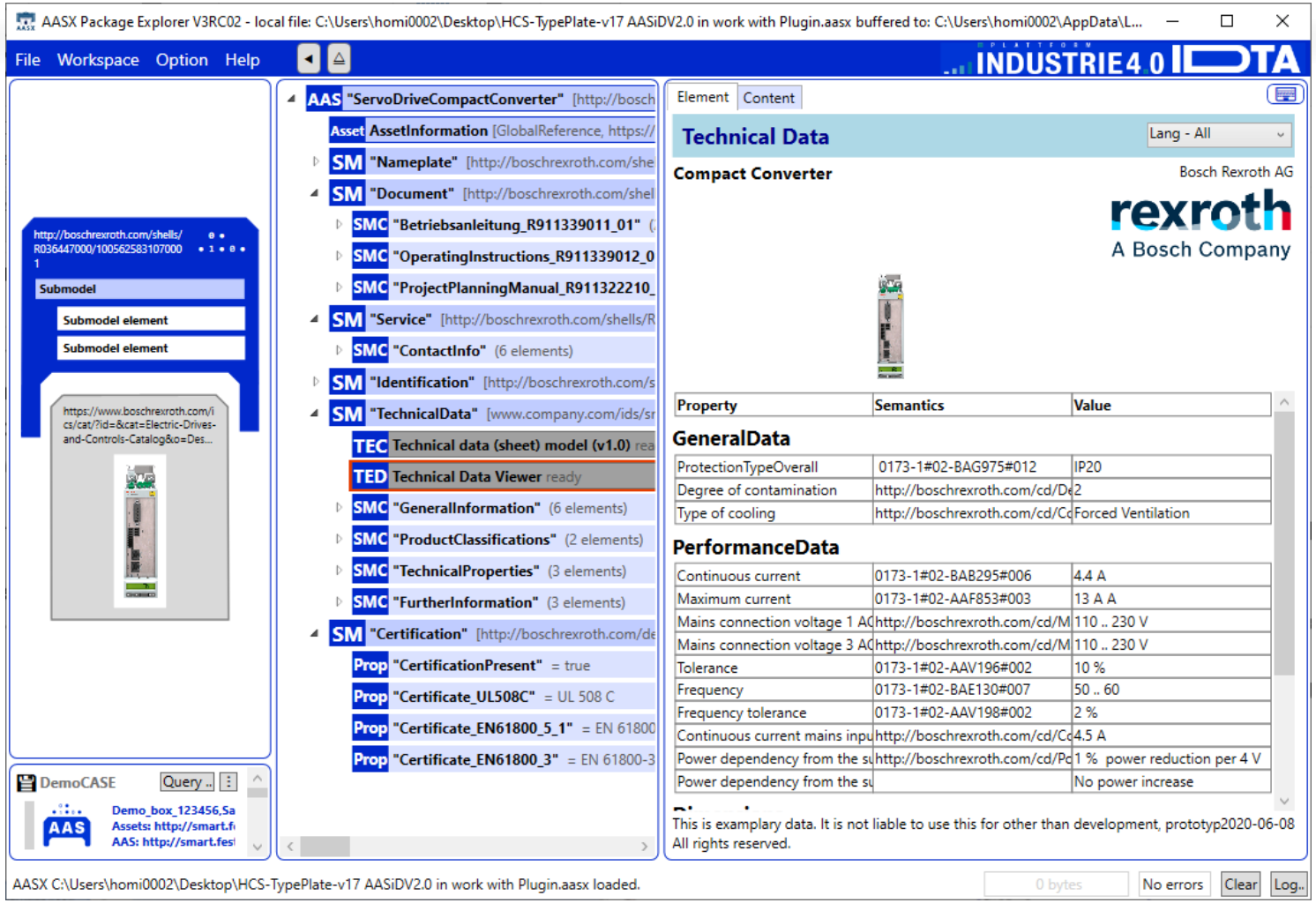
+ preferredName: PreferredNameTypeIec61360
+ shortName: ShortNameTypeIec61360 [0..1]
+ unit: string [0..1]
+ unitId: Reference [0..1]
+ sourceOfDefinition: string [0..1]
+ symbol: string [0..1]
+ dataType: DataTypeIec61360 [0..1]
+ definition: DefinitionTypeIec61360 [0..1]
+ valueFormat: ValueFormatTypeIec61360 [0..1]
+ valueList: ValueList [0..1]
+ value: ValueTypeIec61360 [0..1]
+ levelType: LevelType [0..1]

In formats like xml, JSON, rdf the embedded data specification approach is implemented

Data Specifications in Part 3 – data specification template IEC 61360 just for illustration

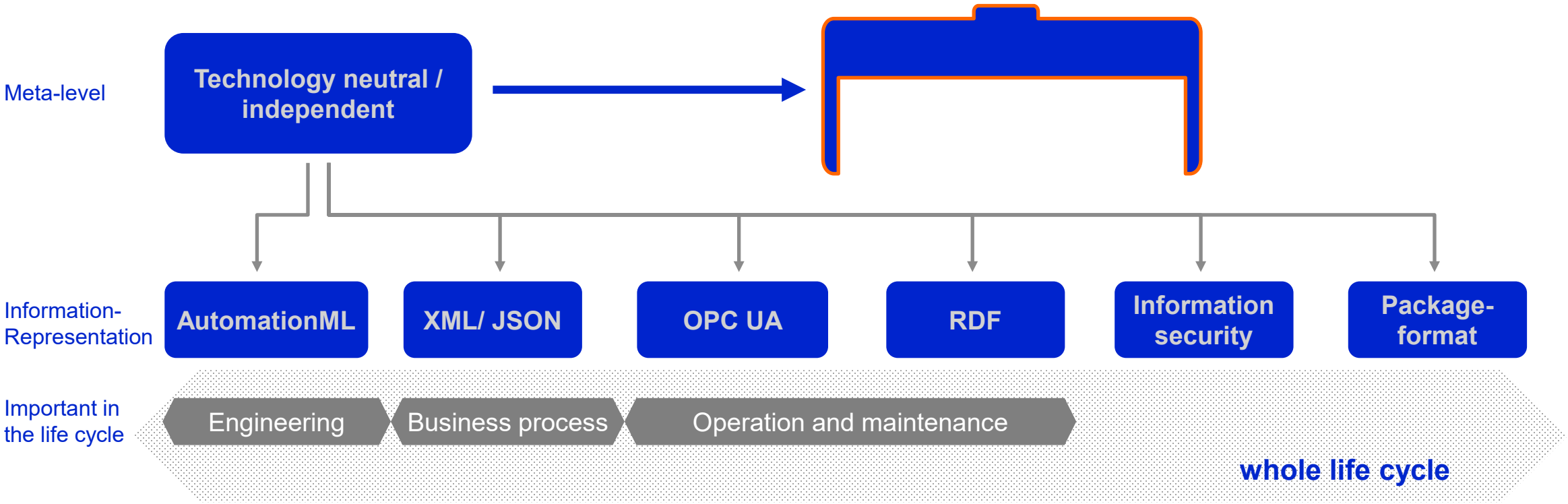

Create your first digital twin

AASX Package Explorer

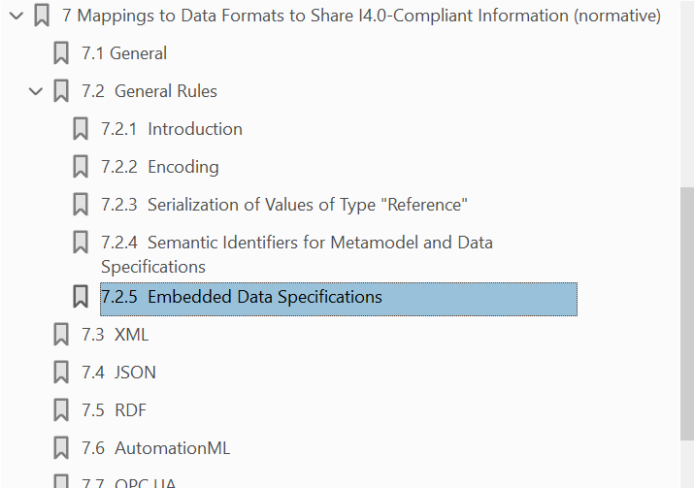


https://github.com/admin-shell-io/aasx-package-explorer

Life Cycle Phases and Formats



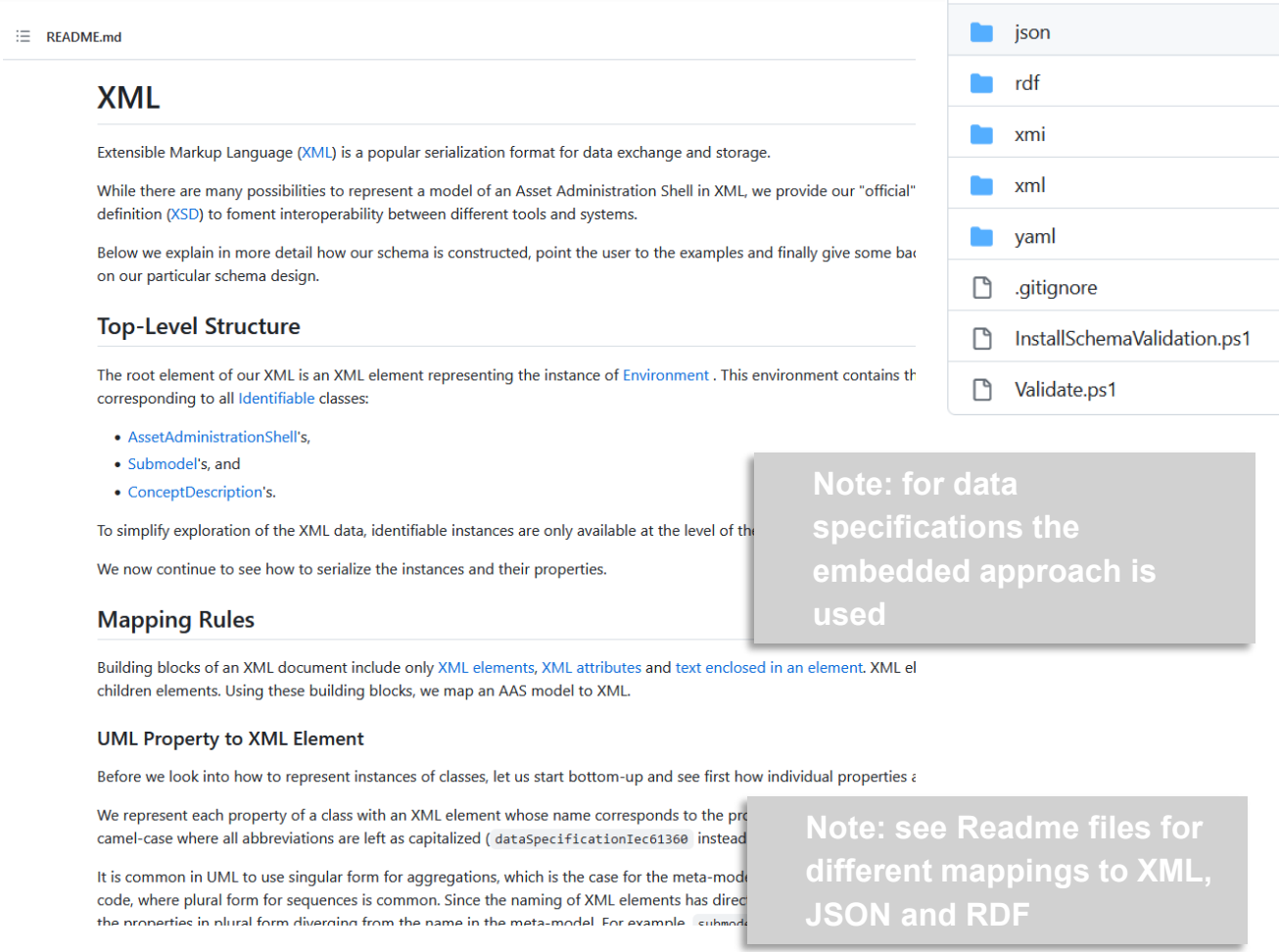
Serializations/Formats



https://github.com/admin-shell-io/aas-specs/tree/master/schemas

Note for Experts: Mapping Rules and Schema for xml, JSON and rdf as well as examples not part of specification any longer ➔ now part of open source project admin-shell-io/aas-specs

Note for Experts: Formats like OPC UA or AutomationML are maintained in OPC Foundation and Automation e.V.



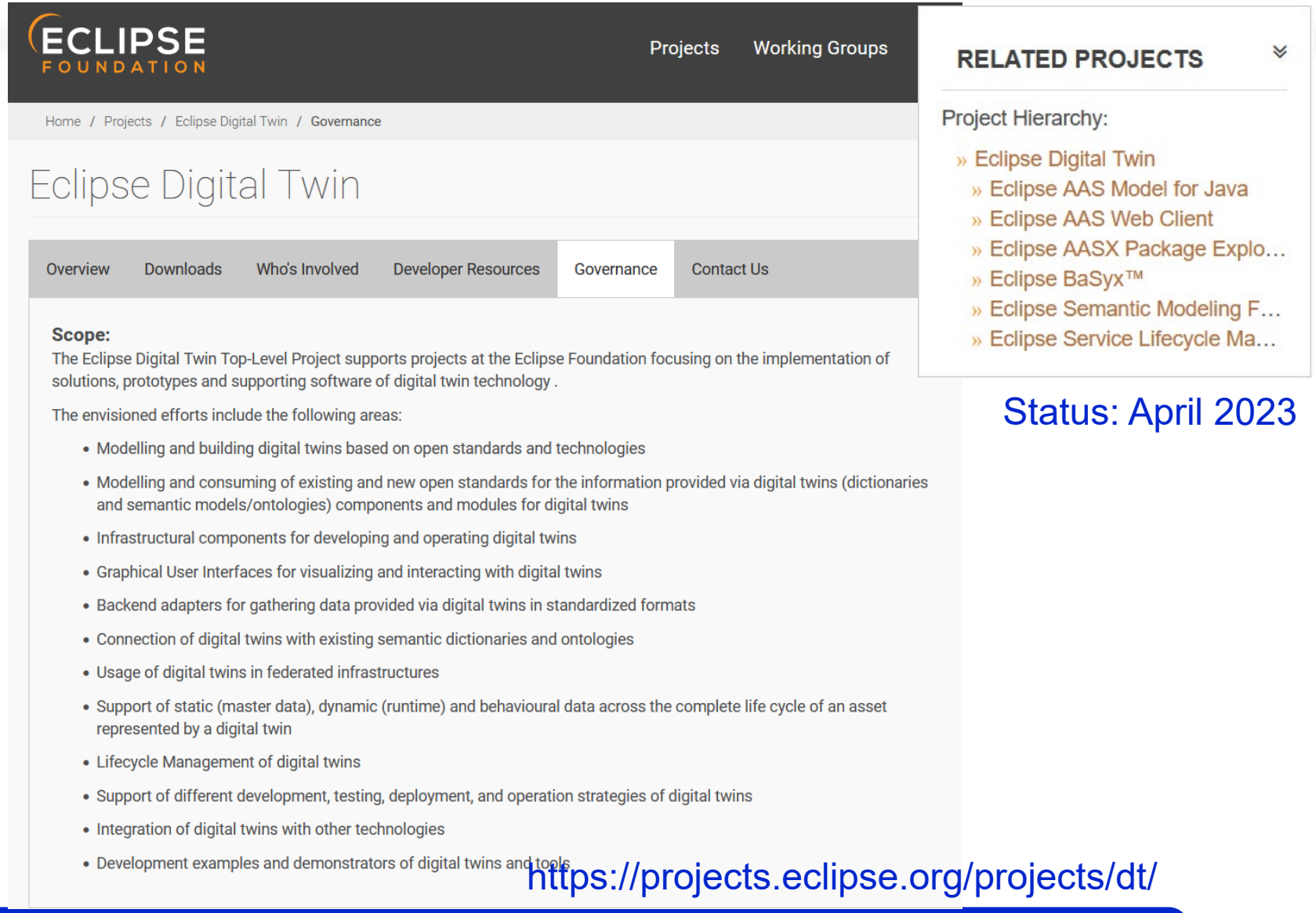
Note: for data specifications the embedded approach is used

Note: see Readme files for different mappings to XML, JSON and RDF

Open Source Support

https://github.com/orgs/admin-shell-io/

Note: specifications maintained in admin-shell-io

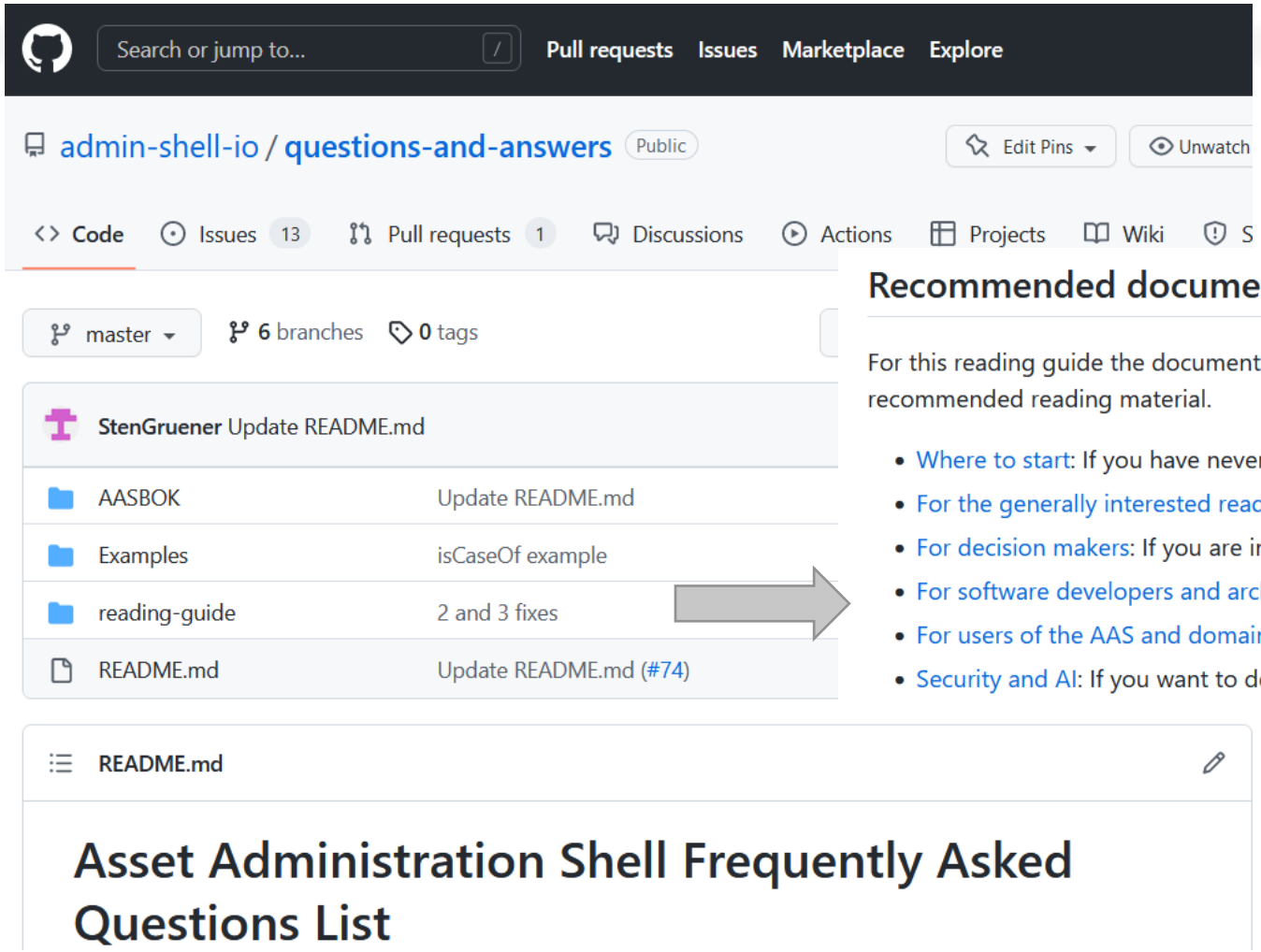


Status: April 2023

https://projects.eclipse.org/projects/dt/


Still Questions?

Questions and Answers



Recommended documents

For this reading guide the documents have been sorted by interest groups rather than topics. In some cases, only specific pages or sections ar recommended reading material.

- Where to start: If you have never heard of the AAS
- For the generally interested reader: If you want to learn more about the subject
- For decision makers: If you are interested in the business side of I4.0
- For software developers and architects: If you want to know how to create software for the AAS
- For users of the AAS and domain experts: If you are interested in using the AAS for specific tasks
- Security and AI: If you want to deep dive into these special topics.

https://github.com/admin-shell-io/questions-and-answers

AAS Academy

www.u4i.io/IDTA

Industrial Digital Twin Association

Let´s go!

Dr. Birgit Boss

Robert Bosch GmbH, Bosch Connected Industry

- Board member of the Industrial Digital Twin Association (IDTA)
- Chair of the Working Group “Open Technology” and its Working Stream “Specifications of the Asset Administration Shell”
- Chair of the Working Group “Semantic Layer including Digital Twins” of Catena-X
- PMC member of the Eclipse Digital Twin Top Level Project

Connect on
www.linkedin.com/in/birgit-boss/

Industrial Digital Twin Association

www.industrialdigitaltwin.org