Building joint forces for the digital twin
From Concept to Specification

April 2023
V3.0

2018
V1.0
This Tutorial is about Part 1
For whom is this tutorial?

- You should have an idea of the benefits of introducing 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 disappointed 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 IIoT
- 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.
Download Specification

https://industrialdigitaltwin.org/content-hub/
Import XMI to your UML tooling

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.>

V3.0RC02
https://www.youtube.com/watch?v=QR-nOl6cuOI
1. Get (re-)familiar with general UML modeling rules
2. Get familiar with specific graphical representation of UML in the specification (partly tool specific)
UML Generalization/Inheritance

A) Class1
   + class2: Class2

   Class3

B) Class1
   Class3

C) Class1
   ::Class1
   + class2: Class2

   Class3

Hint: Graphical representation tool specific
UML Shared Aggregation

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.

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

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

**Template for Classes:**

<table>
<thead>
<tr>
<th>Class:</th>
<th><code>&lt;Class Name&gt; [&lt;&lt;abstract&gt;&gt;] [&quot;&lt;&lt;Experimental&gt;&gt;&quot;] [&quot;&lt;&lt;Depreciated&gt;&gt;&quot;] [&quot;&lt;&lt;Template&gt;&gt;&quot;]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation:</td>
<td><code>&lt;Explanatory text&gt;</code></td>
</tr>
<tr>
<td>Inherits from:</td>
<td>`{&lt;Class Name&gt; &quot;;&quot; }+</td>
</tr>
<tr>
<td>Attribute</td>
<td><code>&lt;attribute or association name&gt; [&quot;&lt;&lt;ordered&gt;&gt;&quot;] [&quot;&lt;&lt;Experimental&gt;&gt;&quot;] [&quot;&lt;&lt;Depreciated&gt;&gt;&quot;]</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Explanation</strong></th>
<th><strong>Type</strong></th>
<th><strong>Card.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Explanatory text&gt;</code></td>
<td><code>&lt;Type&gt;</code></td>
<td><code>&lt;Card&gt;</code></td>
</tr>
</tbody>
</table>

**Note for experts:**
- ModelReference<SubmodelElement> is equal to former notation SubmodelElement*
- No kind column any longer, instead different notation for Type

---

The Asset Administration Shell Metamodel
Example for Class Specification

5.3.7.12 Property Attributes

**DataElement**

<table>
<thead>
<tr>
<th>Property</th>
<th>valueType: DataTypeDefXsd</th>
<th>value: ValueDataType [0..1]</th>
<th>valueId: Reference [0..1]</th>
</tr>
</thead>
</table>

![Figure 39 Metamodel of Properties](image)

<table>
<thead>
<tr>
<th>Class:</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation:</td>
<td>A property is a data element that has a single value.</td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>Inherits from:</td>
<td>DataElement</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Explanation</th>
<th>Type</th>
<th>Card.</th>
</tr>
</thead>
<tbody>
<tr>
<td>valueType</td>
<td>Data type of the value attribute</td>
<td>DataTypeDefXsd</td>
<td>1</td>
</tr>
<tr>
<td>value</td>
<td>The value of the property instance</td>
<td>ValueDataType</td>
<td>0..1</td>
</tr>
<tr>
<td>valueId</td>
<td>Reference to the global unique ID of a coded value</td>
<td>Reference</td>
<td>0..1</td>
</tr>
</tbody>
</table>

Note: it is recommended to use an external reference.
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
The Asset Administration Shell Metamodel

5.3.2 Common Attributes

- 5.3.2.1 General
- 5.3.2.2 Administrative Information Attributes
- 5.3.2.3 Has Data Specification Attributes
- 5.3.2.4 Extensions Attributes
- 5.3.2.5 Has Kind Attributes
- 5.3.2.6 Has Semantics Attributes
- 5.3.2.7 Identifiable Attributes
- 5.3.2.8 Qualifiable Attributes
- 5.3.2.9 Referable Attributes
- 5.3.2.10 Referable Attributes

Common

+ AdministrativeInformation
+ Extension
+ h0_HasExtensions
+ h1_Referable
+ h2_Identifiable
+ h3_HasKind
+ h4_HasSemantics
+ h5_Qualifiable
+ h6_HasDataSpecification
+ Common Entities

Common Entities

+ Qualifier

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:
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.

Note for Experts 3.0:
Checksum handling in discussion, not part of model 3.0.
Common - Qualifiiables

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

Note for Experts: Experimental Qualifier kind introduced
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

**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

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

### HasExtensions

- **extension**: Extension [0..*]

### Referable

- **idShort**: NameType [0..1]
- **displayName**: MultiLanguageNameType [0..1]
- **description**: MultiLanguageTextType [0..1]
- **category**: NameType [0..1]

### Extension

- **name**: NameType
- **valueType**: DataTypeDefXsd [0..1] = xs:string
- **value**: ValueDataType [0..1]

**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.

- **Be aware:** 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
The Asset Administration Shell Metamodel

Common - HasKind

<table>
<thead>
<tr>
<th>«abstract»</th>
<th>ModellingKind</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind: ModellingKind [0..1] = Instance</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>«enumeration»</th>
<th>AssetKind</th>
</tr>
</thead>
<tbody>
<tr>
<td>literals</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td></td>
</tr>
<tr>
<td>Instance</td>
<td></td>
</tr>
<tr>
<td>NotApplicable</td>
<td></td>
</tr>
</tbody>
</table>

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.

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

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.
Referencing

External Global References (e.g. to ECLASS IRDI, manufacturer 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>]

<Key> ::= ["," <KeyType>]

<KeyType> ::= value of AAS:KeyType

<KeyValue> ::= value of AAS:KeyValuePair/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-BA112004008
[ExternalRef] (GlobalReference)0173-1#02-BA112004008
(Submodel) https://example.com/aas/1/1/1234859590, (SubmodelElementList) Documents,
(SubmodelElementCollection)0, (MultiLanguageProperty) Title

Model References:

[ConceptDescription]0173-1#02-BA112004008
[ModelRef] [ConceptDescription]0173-1#02-BA112004008
(Submodel) https://example.com/aas/1/1/1234859590, (Property) Temperature
[ModelRef] [ConceptDescription]0173-1#02-BA112004008
-} (Submodel) https://example.com/aas/1/1/1234859590
```
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

The Asset Administration Shell Metamodel

- **AssetAdministrationShell**
  - + assetInformation: AssetInformation

- **AssetInformation**
  - assetKind
  - specificAssetId: SpecificAssetId [0..*]
  - globalAssetId: Identifier [0..1]
  - assetType: Identifier [0..1]
  - defaultThumbnail: Resource [0..1]

- **SpecificAssetId**
  - name: LabelType
  - value: Identifier
  - externalSubjectId: Reference [0..1]

- **HasSemantics**
  - **AssetKind**
    - literals
    - Type
    - Instance
    - NotApplicable

- **Submodel**
  - + submodelElement: SubmodelElement [0..*]
Submodel

The Asset Administration Shell Metamodel

Submodel

+ submodelElement: SubmodelElement [0..*]

HasDataSpecification

AdministrativeInformation

+ version: VersionType [0..1]
+ revision: RevisionType [0..1]
+ creator: Reference [0..1]
+ templateId: Identifier [0..1]
Overview – Package Submodel Elements

Submodel Elements
- AnnotatedRelationshipElement
- BasicEventElement
- Blob
- Capability
- DataElement
- Entity
- EventElement
- File
- MultiLanguageProperty
- Operation
- Property
- Range
- ReferenceElement
- RelationshipElement
- SubmodelElementCollection
- SubmodelElementList
- Submodel Element Entities

Note for Experts:
SubmodelElementCollection splitted into SubmodelElementCollection and SubmodelElementList

Note for Experts:
experimental new/updated elements for events
Submodel Element Collections and Lists

SubmodelElementCollection
- value: SubmodelElement [0..*]

Referable
HasSemantics
Qualifiable
HasDataSpecification

«abstract»
SubmodelElement

SubmodelElementList
- orderRelevant: boolean [0..1] = True
- semanticIdListElement: Reference [0..1]
- typeValueListElement: AasSubmodelElements
- valueTypeListElement: DataTypeDefXsd [0..1]
- value: SubmodelElement [0..*] {ordered}

SubmodelElement
- Referable
- HasSemantics
- Qualifiable
- HasDataSpecification
- «abstract»

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

Difference in serialization in ValueOnly Format

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

{  "NamesOfFamilyMembers": {    "NameOfMother": "Martha ExampleFamily",    "NameOfFather": "Jonathan ExampleFamily",    "NameOfSon": "Clark ExampleFamily"  }}
- 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
The Asset Administration Shell Metamodel

Embedded Data Specifications

Data Specification Handling - Metamodel

Data Specification Handling - Serialization

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

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

Data Specification Content

- 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]
Create your first digital twin
AASX Package Explorer

https://github.com/admin-shell-io/aasx-package-explorer
Life Cycle Phases and Formats

- Technology neutral / independent
- AutomationML
- XML/JSON
- OPC UA
- RDF
- Information security
- Package-format

Important in the life cycle:
- Engineering
- Business process
- Operation and maintenance

Whole life cycle
Serializations/Formats

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.

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

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

Note: for data specifications the embedded approach is used
Open Source Support

The Asset Administration Shell Metamodel

Status: April 2023

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

Note: specifications maintained in admin-shell-io

https://github.com/orgs/admin-shell-io/
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 are 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 IIoT
- 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
Who we are and what we do...

www.u4i.io/IDTA
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