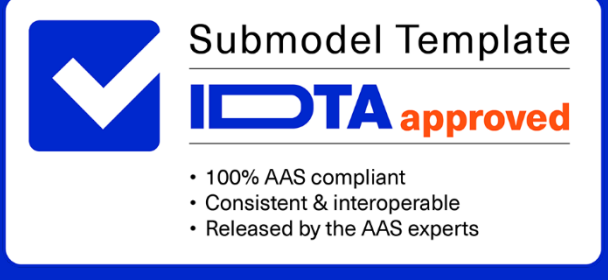
IDTA 02011-1-0 Hierarchical Structures enabling Bills of Material

April 2023

SPECIFICATION

Submodel Template of the Asset Administration Shell



Imprint

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

Version history

Date	Version	Comments
2023-04-17	1.0	Release of the official Submodel template published by IDTA.

Contents

Figures

Tables

1 General

1.1 About this document

This document is a part of a specification series. Each part specifies the contents of a Submodel Template for the Asset Administration Shell (AAS). The AAS is described in [1], [2], [3] and [6]. First exemplary Submodel contents were described in [4], while the actual format of this document was derived by the "Administration Shell in Practice" [5]. The format aims to be very concise, giving only minimal necessary information for applying a Submodel Template, while leaving deeper descriptions and specification of concepts, structures and mapping to the respective documents [1] to [6].

The target group of the specification are developers and editors of technical documentation and manufacturer information, which are describing assets in smart manufacturing by means of the Asset Administration Shell (AAS) and therefore need to create a Submodel instance with a hierarchy of SubmodelElements. This document especially details on the question, which SubmodelElements with which semantic identification shall be used for this purpose.

1.2 Scope of the Submodel

This Submodel Template aims to provide hierarchical structures applicable to industrial equipment in an interoperable manner. For this primarily Entities and Relationship Elements of the AAS Metamodel are used.

This industrial equipment, for example production lines, modules and sub systems, are provided by partners in the value chain, such as suppliers, equipment manufacturers and systems integrators and used in specific applications by industrial operators and end users, both in factory as also process automation. Industrial equipment can be composed of subsystems down to material and component level, can include produced products and can be described on type or instance level.

The AAS contains Submodels that cover aspects of the assets among their life cycle. Already in the design phase, assets are composed and aggregated into newly created hierarchical structures.

Typically, assets have their own AAS (Entity with entityType “SelfManagedEntity"), but it is possible that an Asset has no AAS and is represented by a co-managed Entity.

Since nesting of AAS and Submodels is forbidden by the metamodel, this Submodel is intended to provide a description of the internal structure of an asset. It shall allow the consumer of an AAS to identify assets and their corresponding entities and find their respective AAS if they exist. The Submodel serves as an index, pointing to Assets (described as co- or self-managed entities) and AAS in a distributed network capable of transcending the limits of a single organization.

Instances of this Submodel Template shall be the authoritative source for hierarchical structures within an AAS during all lifecycle phases. Complementing information about each asset and their own lifecycle phase is enabled to be discoverable into the n-th level of the hierarchy and across the whole supply chain depending on the design of the Submodel Instance.

1.3 Not in Scope

This Submodel only defines the structure for a generic asset hierarchy. Modeling specific types of assets is not in the scope of this base standard, neither is the differentiation toward different aspects of an asset (mechanical BoM, electrical BoM) or its lifecycle phases (engineering BoM, manufacturing BoM).

This Submodel Template is designed to be extensible by future versions of this, or similar Submodel Templates, e.g., version 1.1, 1.2, ….

1.4 Relevant standards for the Submodel Template

This Submodel Template for the Asset Administration Shell is specified as a Submodel according to [6]. It is designed to enable the interaction with distributed AAS as specified by [7].

Modeling the hierarchical structure of physical assets has been in scope of several industrial standards. AutomationML allows for the modelling of hierarchical structures via its “InternalElement”-mechanism. OPC UA has a variety of hierarchical references, e.g., the “HasComponent” ReferenceType is commonly used to denote a composition structure of assets, see chapter 14 of [11]. ECLASS has published a guide for expert groups on composition [8]. Many of the ontologies for manufacturing (mostly published in an academic context) also contain methods of nesting assets [9][10]. Up until V3RC01 of the metamodel, an Asset was assumed to link to a Submodel that includes a set of Entity SubmodelElements that serve as a *billOfMaterial*. However, there was never a standard detailing the structure of the Submodel – neither in the Specification of the Asset Administration Shell - Part 1: Metamodel [6] nor in a standardized Submodel Template. The possibility to leverage the RelationshipElement (and AnnotatedRelationshipElement) for this purpose realized neither as it is standardized only as metamodel-elements. Thus, the metamodel offers no mechanism to model hierarchy of assets.

1.5 Use cases, requirements and design decisions

Table 1: Use cases, requirements and design decisions

Use Case	Explanation
Traceability	An OEM assembles a product from suppliers’ subproducts. In order to preserve the data that spawns from processes at (sub-)suppliers, each part should be appended with an AAS. Upon assembly, the assets are aggregated physically, consequently the AAS must be merged as well. This enables the OEM to identify, locate and access the AAS of all delivered products that may contain data on i.e., the source of quality deviations, carbon footprint and arbitrary other data stored in the AAS of the subsystem.
Machine/Plant Design	In order to model the inherent hierarchy of a plant in the process industry, a planner uses this Submodel to integrate the data of supplied or self-designed subsystems.
Delivery of subsystems	In order to integrate potential subsystems into another system, the subsystem creator can model the subsystem in this Submodel, including its subcomponents. The modeled subsystem can be used to complete a system structure by a system engineer in a second step.
Production Order	In loosely coupled value chains, participants continuously monitor the market for suppliers, strengthening resilience and flexibility. Automating this search requires a common interface for a production order including a formal description of all incoming components’ hierarchy.

1.5.1 Design Decisions

As stated in Use-Case “Traceability”, Submodels built from this Template shall enable a data consumer to access data about an assembled asset that is stored in potentially distributed systems. As specified in the Specification of the Asset Administration Shell - Part 2: Application Programming Interfaces [7], Submodels and AAS can run on separate systems. The identity of Assets, AAS and Submodels must be discoverable via AAS mechanisms, e.g., a Registry. Only with this precondition can the references set by the *Entity* SubmodelElements be resolved in a distributed environment. For self-contained environments, e.g., an AASX file, the references noted in the Entities can be discovered via local methods.

1.5.1.1 Co-managed and Self-managed Entities

The Submodel utilizes the *Entity* SubmodelElement for each asset represented inside the hierarchical structure. Assets represented by Self-managed Entities have their own AAS. Co-managed Entities have no own AAS. Algorithms consuming the Submodel built in accordance with this Template will have to use the attribute *entityType* as differentiation.

1.5.1.2 Distributed and Centralized AAS

AAS and their Submodels can be distributed in different forms. The first form is a file-based approach where one or more AAS are serialized. The second form is a distributed format where one or more AAS servers may exist, hosting a variable number of AAS. The Assets described in the AAS can form a functional unit. To allow access to the distributed information in the second approach, relations between the Assets have to be modeled in the AAS. In this Submodel only logical relations are defined to allow the modeling of hierarchical structures.

1.5.1.3 Allowed modelling variants

This Submodel Template allow to model a hierarchy with different archetypes, see Figure 1 for a graphical representation:

- Full: This Submodel Template allows to model a full hierarchy (including sub assets) in a single Submodel as illustrated by the “Full” view below. This is useful if Entities representing Co-Managed Entities have to be expressed, as Co-Managed-Entities typically do not have an Asset Administration Shell of their own. In addition, full modeling also allows a version status to be kept centrally.
- One Down: The One Down archetype is useful for subsystem or component manufactures. For any given Asset in the hierarchy tree, an AAS corresponding to the Asset shall exists. The AAS shall contain a Submodel expressing the one down excerpt view starting with the Asset of the AAS. This type allows the modelling of a consistent stand-alone hierarchy in the engineering-phase of the subsystem. The integration is done by adding the subsystem in a top-level system via the given rules of this Submodel Template, e.g., with the *HasPart* Relation.
- One Up: The One Up relationship is suitable for describing the installation location of an asset. This enables the asset to provide information without external asset administration shells (e.g., in offline scenarios). In addition, the installation location can already be determined when the parent asset and its AAS are still in the planning stage.

The three archetypes may serve different Use-Cases and thus can be selected as needed.

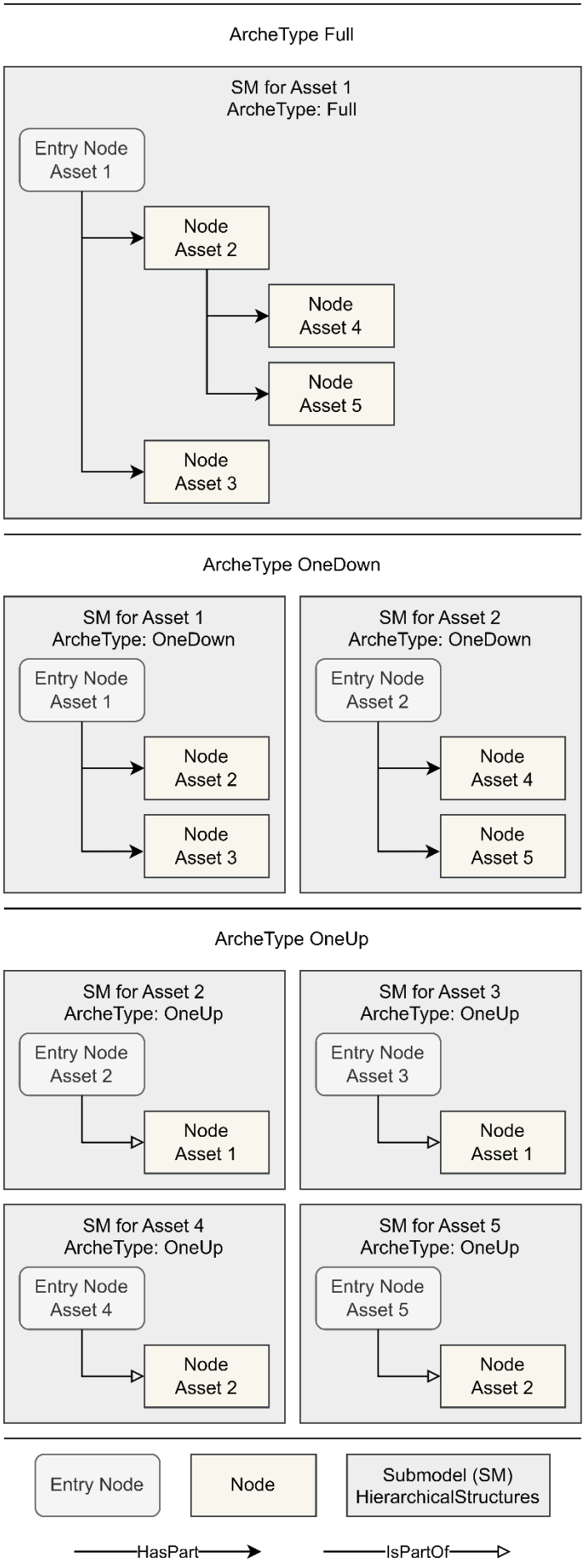


Figure 1: Hierarchy modelling archetypes

1.5.1.4 Implications of modelling types

When choosing the archetype(s), the characteristics of the use cases and the implications of the modelling must be considered. The table lists some of these implications.

Table 2: Implications of modelling types

	Full	One Down	One Up
Use case characteristics	• Low dynamics / fixed hierarchy • Simple version status / central proof management • Technically and organizationally the top AAS is in the lead • Listing of co-managed assets	• Dynamic environment / changing hierarchy • Technically and organizationally, the respective higher-level AAS is in the lead • Listing of co-managed assets	• Dynamic environment / changing hierarchy • Technically and organizationally, the respective subordinate AAS is in the lead
Implications	• Central management • Transparency on all assets down to "bottom" necessary • Complete BoM without query effort • May contain redundant information, as sub-assets can contain BoMs as well.	• Partially centrally controlled • Transparency up to each subordinate AAS necessary • A composite BoM requires high query effort	• Fully decentralized • Creation of a complete BoM not possible, since query effort is too high and from the subordinate asset only "a path upwards" is possible, but not to neighbouring assets of the same hierarchy level.

Use cases can be solved differently based on the modeling archetypes. In practice, there will be mixed forms, as shown in the following example:

Scenario 1: Machine builder creates a hierarchical structure of the machine.

The machine is divided into machine modules, purchased parts such as drive systems are integrated and even components such as the lubricant within the drive system are taken over from the supplier.

Solution approach:

Variant 1: Machine builder creates and maintains Full-BoM

Variant 2: Machine builder creates Full-BoM and reports the installation location of the components

to the supplier for his One-Up hierarchical structure.

Variant 3: Machine builder creates One-Down hierarchical structure of the machine and machine modules and receives online access to the One-Down-BoM of the supplier components.

Scenario 2: Hierarchy of a production line in different lifecycle phases

Solution approach:

Variant 1: During the planning phase, the One-Down hierarchical structure is sufficient, since only the workstations and machines used are to be recorded.

Variant 2: For asset management, the controllers (PLCs) and industrial PCs within the machines and workstations are also required. For this, relationships are created to the hierarchical structure of the workstations.

Variant 3: An edge management system has recorded all IPCs and PLCs. An IPC/PLC can only be managed simultaneously by one higher-level edge management system, so the IPC/PLC uses one-up modelling to avoid simultaneous multi usage.

1.5.1.5 Other design decisions

This Submodel defines several other design decisions, which are described in the following.

- The Submodel defines an “EntryNode” which allows a defined entry-point into the Submodels content. This “EntryNode” must only represent the asset administrated in the corresponding AAS in which this Submodel is registered.
- This Submodel defines relationship representation for parent → child (HasPart) and child → parent (IsPartOf). To allow this Submodel to be machine-readable, one of the two relationships must be modeled depending on the used ArcheType. The relationships must contain an EntryNode or Node as first and second attribute. The relationships shall only reference EntryNodes or Nodes in the same Submodel.
- An instance of this Submodel must only use one archetype. If more than one type shall be modeled a second or third Submodel with the specific archetype must be created next to the existing Submodel. For differentiation of the archetype, the “Archetype” Property of the Submodel can be used.
- For the purpose of modeling distributed hierarchical structures, the archetypes “Full” and “OneDown” can be mixed to create the hierarchy using several Submodels. The “OneUp” archetype cannot be used in a mixed (only Full and OneDown) form with the other two forms.
- The Submodel defines the “SameAs” relationship, the relationship is mainly intended for the two following use cases, but not limited to them. This relationship must contain an EntryNode or Node as first and second attribute.
  In the first use case a “Co-managed” Entity is used to model an asset which does not have an AAS on its own. In this case the “Co-managed” Entity is located in one of the Submodels and can describe the asset. As this Submodel is only intended for the modeling of the structure and not for a detailed description of an asset, the relationship can be used to connect the detailed representation with the representation in this Submodel.
  As this Submodel can be used in a distributed system, the handling of finding represented assets for an entity in this Submodel can take a variable time to resolve. To allow a quicker way of finding a connected entity and its Submodel Hierarchical Structures, the relationship can be used to connect two entities in different Submodels, e.g., a “Node” entity for an asset and an “EntryNode” for the same asset within the Hierarchical Structures Submodel of its own AAS.
- A “Node” Entity can contain the “BulkCount” property which allows the representations of a multiplicity of the Entity the Node is representing. If the “BulkCount” property is present in an Entity, the Entity can only represent an asset with kind type. This allows the integration of elements which are described in the nature of an asset with kind type, but not in the nature of one or more instance assets.
- Below the “EntryNode” an arbitrary number of nested levels of “Nodes” can be nested. Yet for a standardized interaction with the Submodel, the modelling implications described in 1.5.1.4 must be considered.
- This Submodel describes hierarchical structures of entities, yet the Submodel name states the name “Bill of Material” which shall only be understood as a well-known name for an easier understanding of where the Submodel is applicable.

1.6 Further standardization & outlook

This Submodel Template is detailed enough to be used in practice and broad enough to be extended upon for more detailed use cases. When more application-specific hierarchical structures are needed, the elements proposed here may be specified further to meet more specific requirements. Hierarchical structures used in Bills of Material may vary depending on the life cycle phase and the technological scope (see Chapter 1.3). Subsequent standardization activities may analyze the requirements of the community for Submodel Templates enabling interoperable exchange of these data structures that go beyond the basic hierarchy, this especially includes topologies e.g., nets of elements connected by relationships.

1.6.1 Level of Detail

With extensive use of the Submodel and possible future extensions, the Submodel content may become very extensive. To counteract this, levels of details may provide an approach to reflect only parts of the Submodels content. In the following the concept for a potential use in upcoming standards is presented. For the example, three Level-of-Detail (LOD) in the range 0 – 2 are used.

LoD-0 describes only logical references between two Entities, e.g., Component A is a part of Component B.

Figure 2: Level of Detail 0, logical modelling

LoD-1 splits the logical relationship into two separate relations and allows the introduction of an entity which can describe the aspects of the relation in more details.

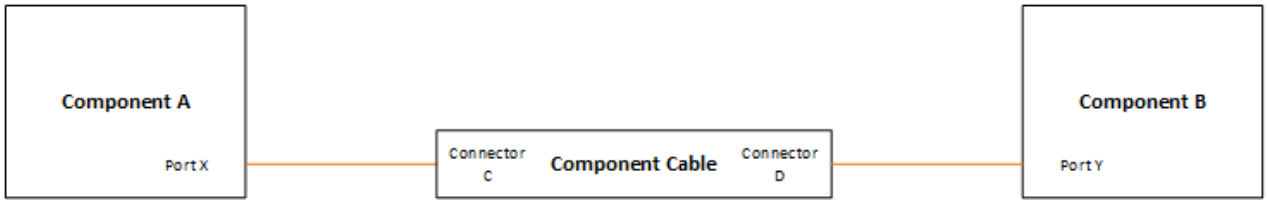


Figure 3: Level of Detail 1, modelling of physical connections

LoD-2 allows to split the relationships from LOD-1 to allow the modeling of more entities describing the relationship between the two components, e.g., Sockets and Plugs.

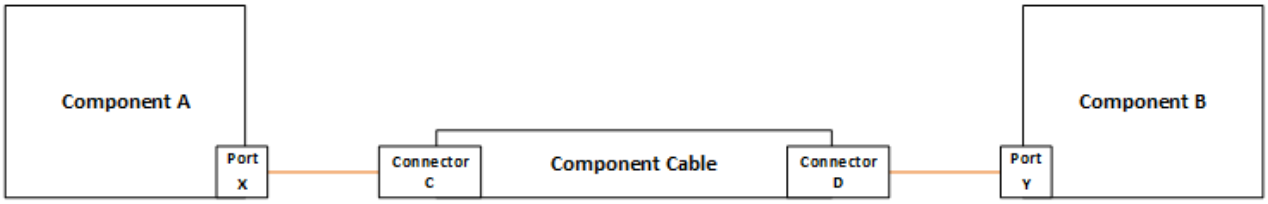


Figure 4: Level of Detail 2, modelling of physical connections and anchor points

2 Submodel hierarchical structures enabling Bills of Material

2.1 Approach

In this document, one Submodel is defined. The Submodel is usable in asset administration shells for type and instance assets.

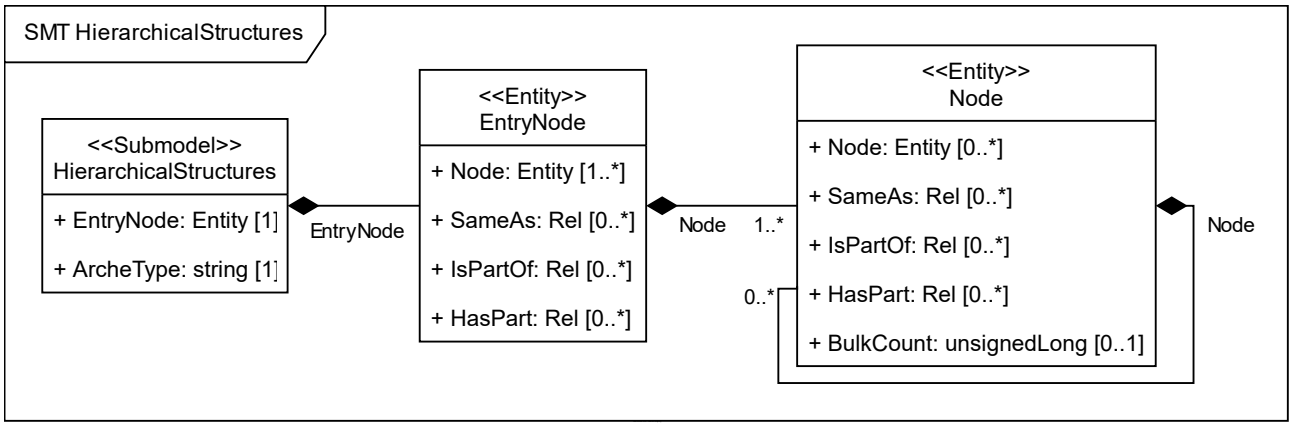


Figure 5: Submodel class diagram

2.2 Attributes of the Submodel

For the Submodel, these important attributes need to be set:

Table 3: Attribute of the Submodel

idShort:	HierarchicalStructures Note: The idShort can be chosen freely.		
Class:	Submodel – HierarchicalStructures		
semanticId:	[IRI] https://admin-shell.io/idta/HierarchicalStructures/1/0/Submodel		
Parent:	AAS		
Explanation:	Definition of the Submodel HierarchicalStructures identified by its semanticId. The Submodel idShort can be picked freely.		
[SME type]	**semanticId = [idType]value**	**[valueType]**	**card.**
idShort	**Description@en**	**example**	
[Entity] EntryNode	[IRI] https://admin-shell.io/idta/HierarchicalStructures/EntryNode/1/0 Base entry point for the Entity tree in this Submodel, this must be a Self-managed Entity reflecting the Assets administrated in the AAS this Submodel is part of.	[-] -	1
[Property] ArcheType	[IRI] https://admin-shell.io/idta/HierarchicalStructures/ArcheType/1/0 ArcheType of the Submodel, there are three allowed enumeration entries: 1. “Full”, 2. “OneDown” and 3. “OneUp”. These entries reflect the structure of the Submodel as defined in 1.5.1.3 & 1.5.1.4.	[string]	1

2.3 SubmodelElements of EntryNode

Table 4: SubmodelElements of EntryNode

idShort:	EntryNode Note: The idShort can be chosen freely.		
Class:	Entity - EntryNode		
semanticId:	[IRI] https://admin-shell.io/idta/HierarchicalStructures/EntryNode/1/0		
Parent:	Submodel HierarchicalStructures		
Explanation:	Base entry point for the Entity tree in this Submodel, this must be a Self-managed Entity reflecting the Assets administrated in the Asset Administration Shell this Submodel is part of. The idShort of the EntryNode can be picked freely and may reflect a name of the asset.		
[SME type]	**semanticId = [idType]value**	**[valueType]**	**card.**
idShort	**Description@en**	**example**	
[Entity] Node	[IRI] https://admin-shell.io/idta/HierarchicalStructures/Node/1/0 The Entity Node can be a co-managed or self-managed entity representing an asset in the hierarchical structure. Note: The idShort can be chosen freely. At least one nested Node shall be created as a SubmodelElement for the EntryNode. In relation to the ArcheType, either the Relationship IsPartOf or HasPart shall be created using this Node as Second attribute.	[-] -	1..*
[Rel] SameAs	[IRI] https://admin-shell.io/idta/HierarchicalStructures/SameAs/1/0 Reference between two Entities in the same Submodel or across Submodels. First and Second attributes must contain either an EntryNode or a Node. Note: The idShort can be chosen freely.	[-] ->	0..*
[Rel] IsPartOf	[IRI] https://admin-shell.io/idta/HierarchicalStructures/IsPartOf/1/0 Modeling of logical connections between asset and sub-asset. Either this or "HasPart" must be used, not both.	[-] ->	0..*

	First and Second attributes must contain either a EntryNode or a Node. The relationships shall only reference EntryNodes or Nodes in the same Submodel. Note: The idShort can be chosen freely.		
[Rel] HasPart	[IRI] https://admin-shell.io/idta/HierarchicalStructures/HasPart/1/0 Modeling of logical connections between components and sub-components. Either this or "IsPartOf" must be used, not both. First and Second attributes must contain either a EntryNode or a Node. The relationships shall only reference EntryNodes or Nodes in the same Submodel. Note: The idShort can be chosen freely.	[-] ->	0..*

2.4 SubmodelElements of Node

Table 5: SubmodelElements of Node

idShort:	Node Note: The idShort can be chosen freely.		
Class:	Entity - Node		
semanticId:	[IRI] https://admin-shell.io/idta/HierarchicalStructures/Node/1/0		
Parent:	EntryNode or Node		
Explanation:	Can be a Co-managed or Self-managed entity. A Node reflects an element in the hierarchical model is set into relation with one or more defined relations. The name of a node can be picked freely but it must be unique in its hierarchical (sub-)level.		
[SME type]	**semanticId = [idType]value**	**[valueType]**	**card.**
idShort	**Description@en**	**example**	
[Entity] Node	[IRI] https://admin-shell.io/idta/HierarchicalStructures/Node/1/0 The Entity Node can be a co-managed or self-managed entity representing an asset in the hierarchical structure. Note: The idShort can be chosen freely.	[-] -	0..*
[Rel] SameAs	[IRI] https://admin-shell.io/idta/HierarchicalStructures/SameAs/1/0 Reference between two Entities in the same Submodel or across Submodels. First attribute must contain either an EntryNode or a Node. The Second attribute may contain an Entity element in a different Submodel, including Submodels of a different specification. Note: The idShort can be chosen freely.	[-] ->	0..*
[Rel] IsPartOf	[IRI] https://admin-shell.io/idta/HierarchicalStructures/IsPartOf/1/0 Modeling of logical connections between components and sub-components. Either this or "HasPart" must be used, not both.	[-] ->	0..*

	First and Second attributes must contain either a EntryNode or a Node. The relationships shall only reference EntryNodes or Nodes in the same Submodel. Note: The idShort can be chosen freely.		
[Rel] HasPart	[IRI] https://admin-shell.io/idta/HierarchicalStructures/HasPart/1/0 Modeling of logical connections between components and sub-components. Either this or "IsPartOf" must be used, not both. First and Second attributes must contain either a EntryNode or a Node. The relationships shall only reference EntryNodes or Nodes in the same Submodel. Note: The idShort can be chosen freely.	[-] ->	0..*
[Property] BulkCount	[IRI] https://admin-shell.io/idta/HierarchicalStructures/BulkCount/1/0 To be used if bulk components are referenced, e.g., a 10x M4x30 screw. Additional constraint: With bulk count only a reference to an asset with kind type is allowed, e.g., the M4x30 type asset.	[unsignedLong]	0..1

Annex A. Explanations on used table formats

1. General

The used tables in this document try to outline information as concise as possible. They do not convey all information on Submodels and SubmodelElements. For this purpose, the definitive definitions are given by a separate file in form of an AASX file of the Submodel Template and its elements.

2. Tables on Submodels and SubmodelElements

For clarity and brevity, a set of rules is used for the tables for describing Submodels and SubmodelElements.

- The tables follow in principle the same conventions as in [5].
- The table heads abbreviate 'cardinality' with 'card'.
- The tables often place two informations in different rows of the same table cell. In this case, the first information is marked out by sharp brackets [] form the second information. A special case are the semanticIds, which are marked out by the format: (type)(local)[idType]value.
- The types of SubmodelElements are abbreviated:

SME type	SubmodelElement type
Property	Property
MLP	MultiLanguageProperty
Range	Range
File	File
Blob	Blob
Ref	ReferenceElement
Rel	RelationshipElement
SMC	SubmodelElementCollection

- If an idShort ends with '{00}', this indicates a suffix of the respective length (here: 2) of decimal digits, in order to make the idShort unique. A different idShort might be choosen, as long as it is unique in the parent's context.
- The Keys of semanticId in the main section feature only idType and value, such as: [IRI]https://admin-shell.io/vdi/2770/1/0/DocumentId/Id. The attributes "type" and "local" (typically "ConceptDescription" and "(local)" or "GlobalReference" and (no-local)") need to be set accordingly; see [6].
- If a table does not contain a column with "parent" heading, all represented attributes share the same parent. This parent is denoted in the head of the table.
- Multi-language strings are represented by the text value, followed by '@'-character and the ISO 639 language code: example@EN.
- The [valueType] is only given for Properties.

Bibliography

[1] "Recommendations for implementing the strategic initiative INDUSTRIE 4.0", acatech, April 2013. [Online]. Available https://www.acatech.de/Publikation/recommendations-for-implementing-the-strategic-initiative-industrie-4-0-final-report-of-the-industrie-4-0-working-group/

[2] "Implementation Strategy Industrie 4.0: Report on the results of the Industrie 4.0 Platform"; BITKOM e.V. / VDMA e.V., /ZVEI e.V., April 2015. [Online]. Available: https://www.bitkom.org/noindex/Publikationen/2016/Sonstiges/Implementation-Strategy-Industrie-40/2016-01-Implementation-Strategy-Industrie40.pdf

[3] "The Structure of the Administration Shell: TRILATERAL PERSPECTIVES from France, Italy and Germany", March 2018, [Online]. Available: https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/hm-2018-trilaterale-coop.html

[4] "Beispiele zur Verwaltungsschale der Industrie 4.0-Komponente – Basisteil (German)"; ZVEI e.V., Whitepaper, November 2016. [Online]. Available: https://www.zvei.org/presse-medien/publikationen/beispiele-zur-verwaltungsschale-der-industrie-40-komponente-basisteil/

[5] "Verwaltungsschale in der Praxis. Wie definiere ich Teilmodelle, beispielhafte Teilmodelle und Interaktion zwischen Verwaltungsschalen (in German)", Version 1.0, April 2019, Plattform Industrie 4.0 in Kooperation mit VDE GMA Fachausschuss 7.20, Federal Ministry for Economic Affairs and Energy (BMWi), Available: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/2019-verwaltungsschale-in-der-praxis.html

[6] "Details of the Asset Administration Shell; Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 2.0.1)", May 2020, [Online]. Available: https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V2.html

[7] "Details of the Asset Administration Shell; Part 2 – Interoperability at Runtime (Version 1.0RC02)", November 2021, [Online]. Available: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part2_V1.html

[8] "Composition of Items – ECLASS Guideline 7", December 2020, [Online]. Available: https://eclass.eu/fileadmin/static/documents/wiki/Technical_Specifications/ECLASS_Guideline_7_Composition-of-Items_v_1.1.pdf

[9] Borgo, Stefano & Masolo, Claudio. Ontological foundations of DOLCE. (2010). 10.1007/978-90-481-8847-5_13.

[10] Smith, Barry & Ameri, Farhad & Cheong, Hyunmin & Kiritsis, Dimitris & Sormaz, Dusan & Will, Chris & Otte, Neil. A First-Order Logic Formalization of the Industrial Ontologies Foundry Signature Using Basic Formal Ontology. (2019).

[11] "OPC 10000-110 OPC Unified Architecture Part 110: Asset Management Basics", Release 1.01.0, November 2022, [Online] https://opcfoundation.org/documents/10000-110/ [PDF], https://reference.opcfoundation.org/AMB/v101/docs/ [HTML]

www.industrialdigitaltwin.org