IDTA 02016-1-0 Control Component Instance

April 2023

SPECIFICATION

Submodel Template of the Asset Administration Shell

Submodel Template

IDTA approved

- 100% AAS compliant
- Consistent & interoperable
- Released by the AAS experts

Imprint

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

Version history

Date	Version	Comment
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 Control Component (CC) concept [7] supports the engineering, deployment and orchestration of service-based and component-oriented industry 4.0 automation solutions. The term component in this document refers to a “modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces” (IEV 741-01-11). In general, a physical device, machine or module can be considered as a component as well as software components, e.g. a function block. Both, the control and the component aspect in this document, are meant to be understood in an abstract manner and from a cybernetical viewpoint of the orchestration (process control) of industrial production processes.

There are two specifications defining how control component information should be modelled as Submodels:

- The “Control Component **Type**” (CCType, IDTA 02015) Submodel to describe control component types and
- the “Control Component **Instance**” (CCInstance, IDTA 02016) Submodel to describe their instances.

This document extends the CCType Submodel specification with a template for the CCInstance Submodel. Common use cases and aspects as well as Submodel elements for this Submodel can be found in the CCType Submodel specification (IDTA 02015) to reduce duplication.

1.2 Scope of the Submodel

The scope of this Submodel is the definition of instance-specific information of a Control Component (CC) into an AAS. Together with its counterpart, the CCType Submodel (IDTA 02015), both Submodels aim to establish templates to ensure a uniform structure. The use of these templates allows the development of manufacturer- and domain-independent control concepts and facilitates the exchange of process information with other Submodels. Additionally, it allows the use of standardized call and monitoring sequences, as well as standardized description of its services, endpoints, error-codes, etc.

1.3 Relevant standards for the Submodel template

Relevant standards that have been taken into account during the specification of both, the CCInstance and also the CCType Submodel are listed in the specification of the CCType Submodel (IDTA 02015),

1.4 Use cases, requirements and design decisions

In-depth discussion on use cases, requirements and design decisions is given in the specification of the CCType Submodel (IDTA 02015), as a CCInstance has to be seen in conjunction with a respective CCType Submodel,

2 Submodel and Collections

2.1 Approach

Most Submodel elements are modelled as mandatory, so that an application only needs to check their values, without the need to additionally check if the element is present beforehand. For example, an empty collection of skills indicates that no skills are available in the type or instance, but the collection itself is mandatory.

2.2 Elements of the Submodel "ControlComponentInstance"

Figure 1 shows the UML-diagram defining the relevant properties for the instance Submodel of a CC. Table 1 describes the details of the Submodel structure. Table 2 specifies the SMC "Endpoints". The structure of the SMC "Skills" of a CCInstance Submodel is identical to one in a CCType Submodel. It is described in section 2.3.1.

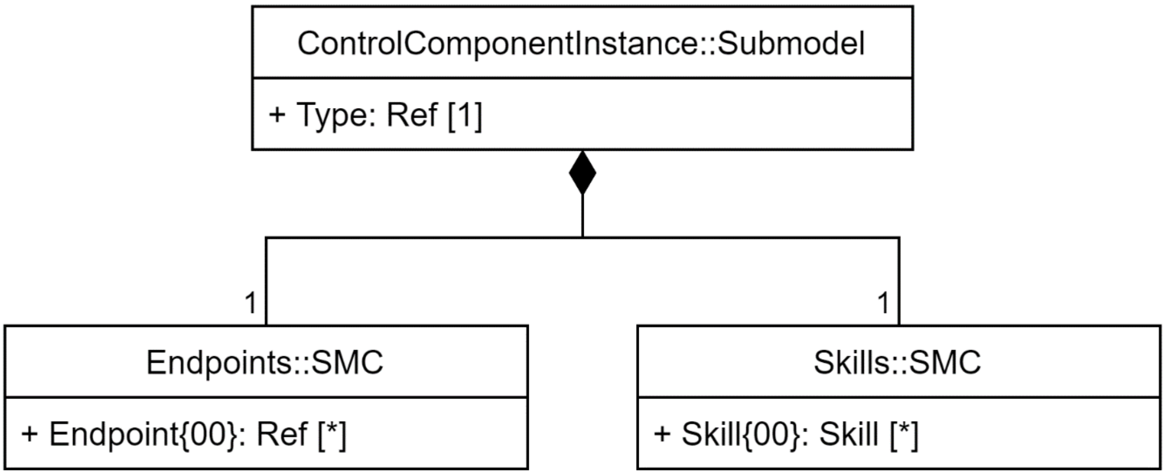

Figure 1: UML-Diagram for Submodel "ControlComponentInstance"

Table 1: Elements of Submodel ControlComponentInstance

idShort:	ControlComponentInstance Note: The above idShort shall always be as stated.		
Class:	Submodel		
semanticId:	[IRI] https://admin-shell.io/idta/ControlComponent/Instance/1/0		
Kind:	Instance		
Version	1		
Revision	0		
Parent:	Asset Administration Shell of component instance as asset		
Explanation:	Contains the instance information of a control component		
[SME type]	**semanticId = [idType]value**	**[valueType]**	**card.**
idShort	**Description@en**	**example**	
[SMC] Endpoints	[IRI] https://admin-shell.io/idta/ControlComponent/Instance/Endpoints/1/0 Collection of references to control endpoints supported by the instance of the component, e.g., to the Endpoint Metadata SMC of the Asset Interface Description Submodel (IDTA 02017), the MTP submodel (IDTA 02001) or OPC UA Server Datasheet submodel (IDTA 02009).	n/a	1
[SMC] Skills	[IRI] https://admin-shell.io/idta/controlcomponent/Skills/1/0 Collection of skills offered by the component instance Note: The skills of the control component can be either type-specific or instance-specific. This collection includes the skills offered specifically by the component instance	n/a	1
[Ref] Type	[IRI] https://admin-shell.io/idta/ControlComponent/Instance/Type/1/0 Reference between the component instance and its respective ControlComponentType Submodel.	n/a	1

Table 2: Elements of SMC Endpoints

idShort:	Endpoints		
Class:	SubmodelElementCollecttion		
semanticId:	[IRI] https://admin-shell.io/idta/ControlComponent/Instance/Endpoints/1/0		
Kind:	Instance		
Parent:	Submodel “ControlComponentInstance”		
Explanation:	Collection of endpoint references		
[SME type]	**semanticId = [idType]value**	**[valueType]**	**card.**
idShort	**Description@en**	**example**	
[Ref] Endpoint{00}	[IRI] https://admin-shell.io/idta/ControlComponent/Instance/Endpoint/1/0 A reference to a control endpoint	n/a	*

2.3 Common Submodel Elements for CCType and CCInstance Submodel

The CCInstance Submodel allows to overwrite or add instance specific skills. Hence, we define a common SubmodelElementCollection (SMC) for skills and their elements to be used in both, CCType and CCInstance Submodels.

2.3.1 Elements of the SMC “Skills”

Figure 2 shows the UML-diagram defining the relevant properties of the SubmodelElementCollection “Skill”. Table 3 describes the details of the SMC “Skill”. It contains a SMC “Modes” and “Parameters” whose contents are described in detail in Table 4 and Table 5.

Figure 2 UML-Diagram for SMC "Skill"

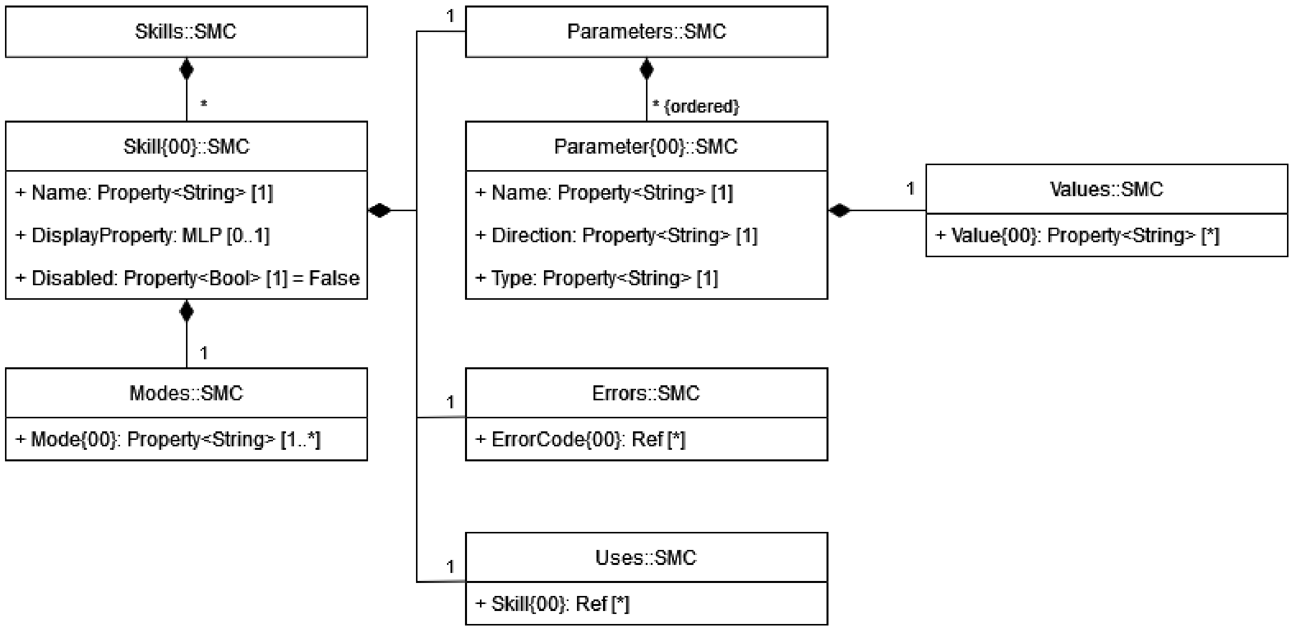
Skills::SMC
*
Skill{00}::SMC
+ Name: Property<String> [1]
+ DisplayProperty: MLP [0..1]
+ Disabled: Property<Bool> [1] = False
1
Modes::SMC
+ Mode{00}: Property<String> [1..*]
1
Parameters::SMC
* {ordered}
Parameter{00}::SMC
+ Name: Property<String> [1]
+ Direction: Property<String> [1]
+ Type: Property<String> [1]
1
Values::SMC
+ Value{00}: Property<String> [*]
1
Errors::SMC
+ ErrorCode{00}: Ref [*]
1
Uses::SMC
+ Skill{00}: Ref [*]

Table 3: Elements of SMC "Skill"

idShort:	Skill{00}		
Class:	SubmodelElementCollection		
semanticId:	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/1/0		
Kind:	Instance		
Parent:	SubmodelElementCollection “Skills”		
Explanation:	Contains the basic information to call (request the execution of) a skill, e.g. its signature		
[SME type]	**semanticId = [idType]value**	**[valueType]**	**card.**
idShort	**Description@en**	**example**	
[Property] Name	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/Name/1/0 Name used to select the skill via its interface	[String] “PICK”	1
[MLP] DisplayName	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/DisplayName/1/0 Name to display the skill, e.g. in an HMI or GUI	[LangStringSet] “en-us”, “Pick”	0..1
[Property] Disabled	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/Disabled/1/0 Boolean variable that defines if the skill is (currently) disabled, e.g. not licensed, tested, suitable, ….	[Boolean] False	1
[SMC] Modes	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/Modes/1/0 Collection of operation, operating, operational or execution modes (depending on the standard), in which the skill is available/allowed to execute.	n/a	1
[SMC] Parameters	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/Parameters/1/0 Collection of parameters used for the configuration of the skill.	n/a	1
[SMC] Errors	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/Errors/1/0 Collection of references to the error codes of the component that may be triggered by this skill.	n/a	1
[SMC] Uses	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/Uses/1/0 Collection of references to other skills, that this skill uses.	n/a	1

Table 4: Elements of SMC "Modes"

idShort:	Modes		
Class:	SubmodelElementCollection		
semanticId:	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/Modes/1/0		
Kind:	Instance		
Parent:	SubmodelElementCollection “Skill”		
Explanation:	Collection of operation, operating, operational or execution modes (depending on the standard), in which the skill is available/allowed to execute.		
[SME type]	**semanticId = [idType]value**	**[valueType]**	**card.**
idShort	**Description@en**	**example**	
[Property] Mode{00}	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/Mode/1/0 Name of the operation, operating, operational or execution modes (depending on the standard), in which the skill is available/allowed to execute.	[String] e.g. “AUTO”, “SEMIAUTO”, “MANUAL”, “SIMULATE”	1..*

Table 5: Elements of SMC "Parameter"

idShort:	Parameter{00}		
Class:	SubmodelElementCollection		
semanticId:	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/Parameter/1/0		
Kind:	Instance		
Parent:	SubmodelElementCollection “Parameters”		
Explanation:	Parameter used for the configuration of the skill		
[SME type]	**semanticId = [idType]value**	**[valueType]**	**card.**
idShort	**Description@en**	**example**	
[Property] Name	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/Parameter/Name/1/0 Name of the parameter.	[String] “Position”	1
[Property] Direction	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/Parameter/Direction/1/0 Indicates whether the parameter is an input (In) or an output (Out) of the skill. This also determines, whether the skill will read (In) or write (Out) the value. Hence, an InOut parameter can be set from outside and can also be changed from skill itself.	[String] “In”	1
[Property] Type	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/Parameter/Type/1/0 Data type as string used to interpret the parameter. Because the technology for implementing a CC is intenionally left open for the vendor, it is not possible to reference a specific type set. Especially the XML data type set or AAS-specific subsets are not sufficicent. Example: a skill could use a custom data type (IEC 61131 / OPC UA Structure, a Class in Java, C#, ...) as a parameter, e.g., a struct containing three float variables representing a 3D position.	[String] “Integer”	1
[SMC] Values	[IRI] https://admin-shell.io/idta/ControlComponent/Skill/Parameter/Values/1/0 Collection of properties of the accepted values that the parameter may take. Each entry of the collection may contain a semantic description of the meaning of the parameter value.	n/a	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 3.0RC01)”, November 2020, [Online]. Available: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Details-of-the-Asset-Administration-Shell-Part1.html

[7] Grothoff, J., Porta, D., Espen, D., Haque, A., Schnicke, F., and Kuhn, T., BaSyx ControlComponent, 2021. https://wiki.eclipse.org/BaSyx_/_Documentation_/_ControlComponent (accessed April 04, 2023).

www.industrialdigitaltwin.org