Digitale Interoperabilität in kollaborativen Wertschöpfungsnetzwerken der Industrie 4.0

SPECIFICATION

Submodel

Software Bill of Materials

Version 1.0

20 June 2023

Submodel Template of the

Asset Adminstration Shell

Imprint

Publisher

Steinbeis Innovation gGmbH
Adornostr. 8
70599 Stuttgart
Germany

Source for Specification Document

Plattform Industrie 4.0
Bertolt-Brecht-Platz 3
10117 Berlin
Germany

Authors

Frank Scherenschlich
Sophia Machaidze
Christian Albert

Version history

2023-08-28	1.0	Submission to the InterOpera Consortium

Content

List of Figures

List of Tables

Foreword

This document is the result of the InterOpera project “Software Bill of Material (SBOM)”. The development of the submodel was coordinated by the company Class.Ing as project manager. People from different companies took part in the conception in different roles, which considered many of the market requirements. Participants of the workshops and the elaboration were:

Frank Scherenschlich, Class.Ing Ingenieur-Partnerschaft

Sophia Machaidze, Class.Ing Ingenieur-Partnerschaft

Christian Albert, Class.Ing Ingenieur-Partnerschaft

Andre Bröring, Technische Hochschule Ostwestfalen-Lippe

Andreas Harner, VDE Verband der Elektrotechnik Elektronik Informationstechnik e. V.

Dr. Markus Schoisswohl, Hegla New Technology Gmbh & Co KG

Stefan Ullmann, ifm electronic GmbH

Kai Schwermann, bill-X GmbH

Yannis Vierkötter, bill-X GmbH

We thank for the commitment

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 [1-3] 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 [1-6].

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 at interoperable provision of information describing Software Bill of Material (SBOM) in regard to the asset of the respective Asset Administration Shell. Central element is the provision of properties [7], ideally interoperable by the means of dictionaries such as ECLASS and IEC CDD (Common Data Dictionary). The purpose of this document is to make selected specifications of Submodels in such manner that information about assets can be exchanged in a meaningful way between partners in a value creation network. It targets equipment for process industry and factory automation by defining standardized meta data.

The intended use-case is the provision of a standardized property structure for Software Bill of Material, which enables the interoperability of Software Bill of Material from different manufacturers. The more detailed description can be found in chapter 2.

This concept can serve as a basis for standardizing the respective Submodel. The conception is based on existing norms, studies of common practices at enterprises, directives and standards so that a far-reaching acceptance can be achieved.

Beside standardized Submodel this template also introduces standardized SubmodelElementCollections (SMC) in order to improve the interoperability while modelling aspects of Software Bill of Material within other Submodels.

1.3 Relevant standards and sources of concepts for the Submodel template

According to [3], interoperable properties might be defined by standards, consortium specifications or manufacturer specifications. Useful standards providing sources of concepts are: List of examplary standards defining interoperable properties

Table 1: List of examplary standards defining interoperable properties

DIN EN 62559-2	Use case methodology Part 2: Definition of the template for use cases, actor list and requirements list
ISO/IEC 5962:2021	Information technology — SPDX® Specification V2.2.1
ISO/IEC 19770-2:2015	Information technology — IT asset management — Part 2: Software identification tag

So called property dictionaries are used identify information elements (see Terms and Definitions of [6]). Such property dictionaries include:

- ECLASS, see: https://www.eclasscontent.com/
- IEC CDD, see: https://cdd.iec.ch/cdd/iec61987/iec61987.nsf and https://cdd.iec.ch/cdd/iec62683/cdddev.nsf

In this document, properties are aimed to be described by ECLASS.

2 Information set for Submodel Contact Information

2.1 Brief description

Both in software development itself and in software-using industrial components and systems, upstream elements are used as part of the value chain. For the interoperable provision of hierarchical units and relationships of software components, the concept of software bill of materials (SBOM) is used. The software bill of materials is mapped as a role-dependent submodel in the administration shell (AAS).

The AAS submodel "Software Bill of Materials" represents a formal, machine-readable inventory of software components and their dependencies as well as information about these components and their hierarchical relationships. The currently existing standardised formats for the interoperable exchange of SBOM data are taken into account.

2.2 Description of the „Software Bill of Materials“ submodel

In addition to the description and referencing of the individual versions of the software, the "SBOM" submodel also contains a description of each element of the software parts list. In the SBOM environment, standard formats such as SPDX and CycloneDX already exist, which specify corresponding description fields and a standardised format for the description.

Derived from this, meta-data for each SBOM are provided as features for description and clear identification. The characteristics were identified as an important intersection of the standard formats and are differentiated according to optional and mandatory characteristics. The actual description is done on referenced or in the AAS contained source files in the standard format. In addition, for each description, the hash value can also be transferred according to standard hash formation procedures...

2.3 Focus

The submodel "SBOM" focuses on the mapping of the software parts list with versioning and the description of the individual elements via standardised characteristics. The created software itself is described via the IDTA submodel 02007 "Software Nameplate".

Other submodels that also provide specifications depending on the version and / or element (e.g. hardware BOM, vulnerability) are addressed via references. Thus, a clear separation of the contents exists.

This procedure is also used for the special case described below. In the case of software-using hardware, parameterisation files are often transferred by the INTEGRATOR at instance level for the use of the product. These files are not part of the submodel "SBOM", but are provided in other submodels such as "Handover Documentation".

2.4 Process representation and role consideration

The following consideration is based on an analysis of the roles, their respective tasks and the interaction between the roles .

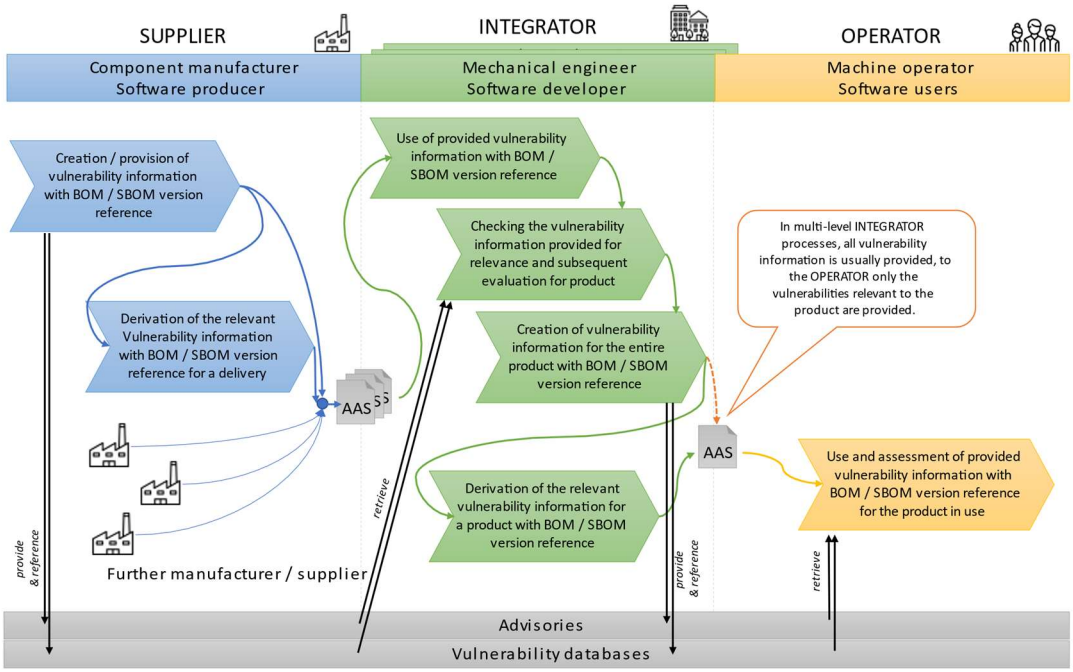

Figure 1: Process illustration SBOM

2.5 Actors

Table 2: Actors

File name	Actor type	Description Actor	more info
SUPPLIER	Business	Manufacturers of software components such as modules, libraries, etc., as well as manufacturers of technical products that use software for operation.	**Tasks** • Version-dependent description of the developed software components • Use of standard descriptions such as SPDX or CycloneDX for the description of the elements • Creation of references to the hardware used in the context of the bill of material (IDTA submodel 02011 "Hierarchical Structures enabling Bills of Material") as well as to detected vulnerabilities in the software itself or in the combination of hardware and software used (IDTA submodel 02037 "Vulnerability Management ")

			• Instance derivation of SBOM information for delivered products with the relevant information • Provision of SBOM information either in typed form (all versions) or in instance form (per version) In order for this concept to work optimally and efficiently, all SUPPLIERS make their SBOM available to the INTEGRATOR in the form of management trays.
INTEGRATOR	Business	Software developers who use software components from SUPPLIER to create their products and who make their own software available to the market as part of the value chain	**Tasks** • Use of provided component SBOM of different SUPPLIER and integration into the documentation of the own software development depending on the version • Specification of component SBOM in the same version-dependent manner if the SUPPLIER does not provide corresponding machine-readable data. • Creation of an SBOM for the own software developed within the framework of the value chain ○ where a versioning of the provided software versions is also carried out • Interaction with hardware in use in the context of BOM • Instance-based derivation of relevant information for delivered version statuses
OPERATOR	Business	Users of the software who are provided with and use information on the software used by INTEGRATOR.	**Tasks** • Use of the instance information provided by the INTEGRATOR • Evaluation of new components, e.g. for carrying out the software updates provided by the INTEGRATOR.

2.6 Interaction of the roles

The roles represent independent actors according to the process view. This means that data is exchanged between these actors in a standardised form:

- SUPPLIER provide machine-readable data for the INTEGRATOR
- The INTEGRATOR provides machine-readable data for the OPERATOR

2.7 Benefits

The consistent use of the same machine-readable formats and the further use of the data along the value chain creates a great advantage. This is done using already existing and established standard formats, which are integrated accordingly.

2.8 Technical description

In the technical description, the procedural requirements are transferred to the technology used for the administration shell. The submodels used are discussed as well as the type of administration shell, differentiated according to type and instance as well as "As-Built" and "As-IS".

2.9 Administration shells and submodels

Representation for the derivation of the management scales

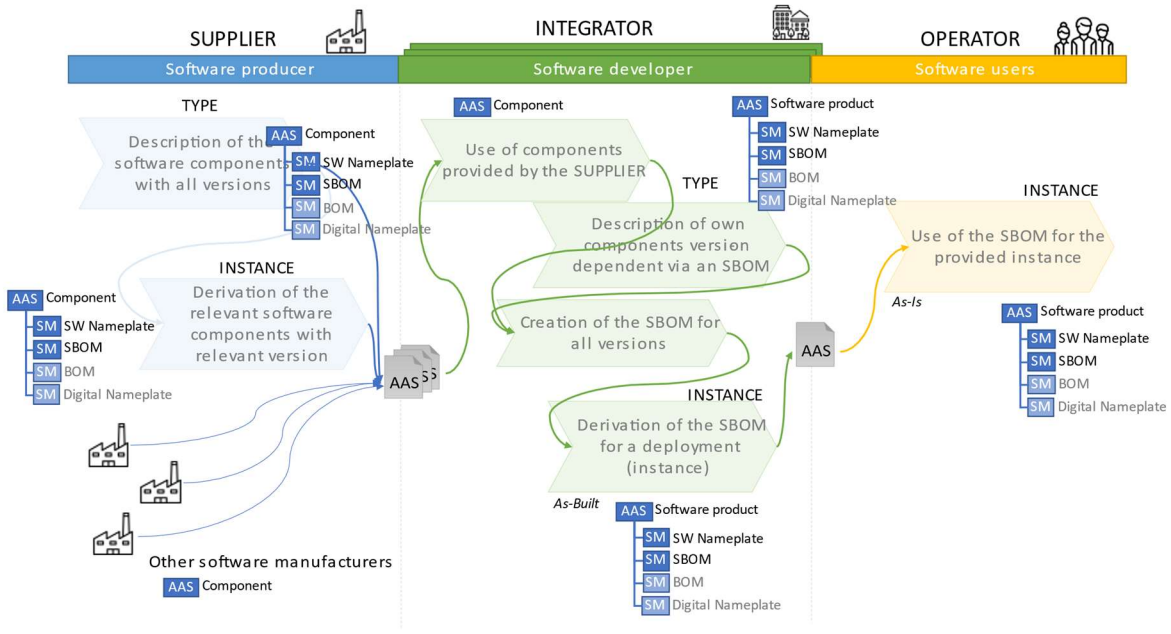

Figure 2: Technical description with management tray

SUPPLIER

The "software manufacturer" also includes suppliers and dealers who distribute software and who do not carry out any further "refinement" of the software.

Each software component is described via a separate administration shell (AAS). The IDTA submodel 02007 "Software Nameplate" is used for the basic description. The provided version statuses of the respective software parts list are defined via the submodel "SBOM", whereby all relevant software statuses are represented in the type description and only the respective software version is represented in a possible instance description.

In the case of interaction with hardware, as often occurs in the industrial environment, referencing to the hardware BOM takes place, which is mapped in IDTA submodel 02011 "Hierarchical Structures enabling Bills of Material". If vulnerabilities have been identified and information on them is to be provided, these are described and referenced with IDTA submodel 02037 "Vulnerability Management" (version 1.0).

INTEGRATOR

The INTEGRATOR uses the administration shells of the SUPPLIER within the scope of its development, if these are provided. For components used where the SUPPLIER does not provide an AAS, the INTEGRATOR itself creates a description as its own administration shell and references it.

On this basis, the INTEGRATOR also creates an administration shell for its own created software, which describes precisely this software in the IDTA submodel 02007 "Software Nameplate" (version 1.0). For own elements of the software, the submodel "Software Bill of Materials" is also used here. And here, too, integration and referencing to the IDTA submodels 02011 "Hierarchical Structures enabling Bills of Material" for connected hardware and "Vulnerability Management" for vulnerabilities connected with the software or the software-hardware combination takes place if necessary.

For the transfer to the OPERATOR, i.e. the user of the software, the INTEGRATOR creates an instance administration shell that contains only the partial model contents that are also contained in the transferred version of the software or software-hardware combination. This is an "As-Built" image, i.e. an image as created and provided by the INTEGRATOR (whereby the INTEGRATOR has no (or only limited) knowledge of its use by the OPERATOR).

OPERATOR

The OPERATOR receives the "As-Built" instance administration shell from the INTEGRATOR, which initially corresponds to the state of delivery and is subsequently updated accordingly via updates. With the transfer and use at the OPERATOR, it becomes an "As-IS" administration shell, since it is considered according to the intended use at the OPERATOR. The OPERATOR is independently responsible, for example, for applying updates for the software about which he is informed by the INTEGRATOR in the form of updates of the "As-Built" administration shell in his environment and also to document this in his "As-IS" administration shell. The same applies to the IDTA submodels 02011 "Hierarchical Structures enabling Bills of Material" and "Vulnerability Management", whereby in the latter case, for example, an assessment of vulnerabilities is carried out by the OPERATOR and these can then be assessed and documented in the AAS in the form of VEX entries.

2.10 Differentiation according to type and instance

The distinctions between type and instance of the administration shell have already been given in the previous sections. Type administration shells describe a created software across the board with all versions and thus also with all components used. For versions of software that have been provided, the provision focuses on the concrete provision, i.e. which components etc. are contained in which version.

2.11 Interactions with other systems

The "SBOM" submodel is usually considered separately from other systems, as there are currently no central locations for SBOM data records. Various SUPPLIERS or partly also INTEGRATORS provide version-dependent information with SBOM information via their own solutions. These sources can be referenced via the submodel "SBOM".

The situation is different in the area of vulnerabilities (IDTA submodel 02037 "Vulnerability Management") - here, relevant databases (CSAF Aggegratoren, CERT@VDE, ICS-CERT, etc.) with standardised formats are available as well as SUPPLIER or INTEGRATOR specific advisories. These can also be referenced starting from the submodel used.

2.12 Sector relevance

The submodel "SBOM" was considered from different industry perspectives in the InterOpera project of the same name and thus includes coverage of all these requirements. Considered in this context were:

- Software manufacturer (SUPPLIER)
  - >> Provision of software components
- Component manufacturer with hardware and software (SUPPLIER)
  - >> Provision of hardware-software products
- Software Developer (INTEGRATOR)
  - >> Development company for software products of various kinds
- Mechanical engineer (INTEGRATOR)
  - >> Use of pure software components or hardware-software components as part of the construction of machines
  - >> Provision of own SBOM information for the created plant
- Software user (OPERATOR)
  - >> Use of provided software products in the context of the operation of a software or a system

3 Submodel and Collections

3.1 Approach

Submodel defined in this document is usable in asset management shells for type and instance assets.

The submodel "Software Bill of Material" can be used in the "type" AAS (usually internal maintenance of item-related information) and as an "instance" AAS (transfer of item-related data for an item or use in operation). It is divided into the following three areas at the top level of the submodel:

- "**Deployments**" stores information about which internal and external software versions are possible on a target system or a production environment, i.e., can be installed or run, e.g., the version of a library. For a description it refers to …
- "**Deployment Parts**" that provide a list of all meta-level SBOM descriptions with attributes and references. Examples of this are: status, supplier, file, author, list of references to components used such as the operating system, etc. The deployment parts either refer to their own AAS or to ...
- "**Component Parts**", which contains at least a brief documentation of the component.

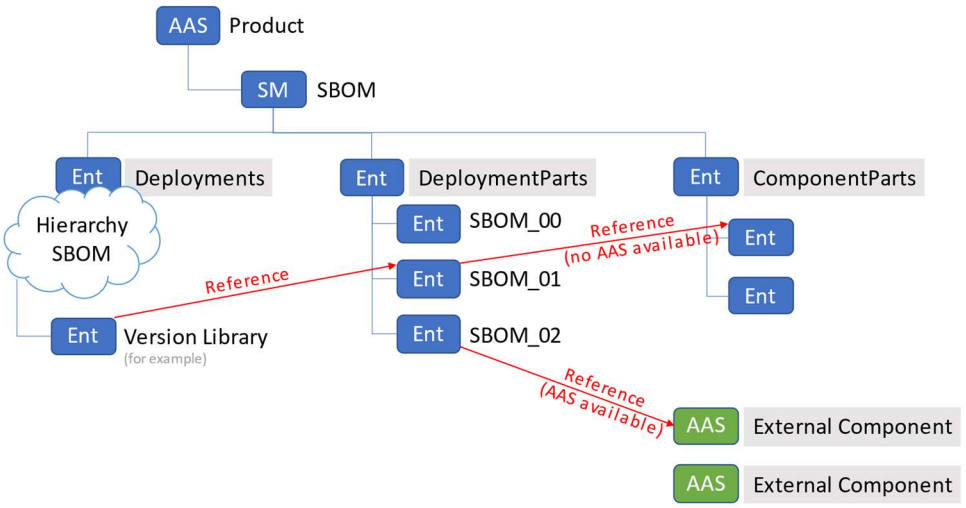

Figure 3: Areas of the SBOM

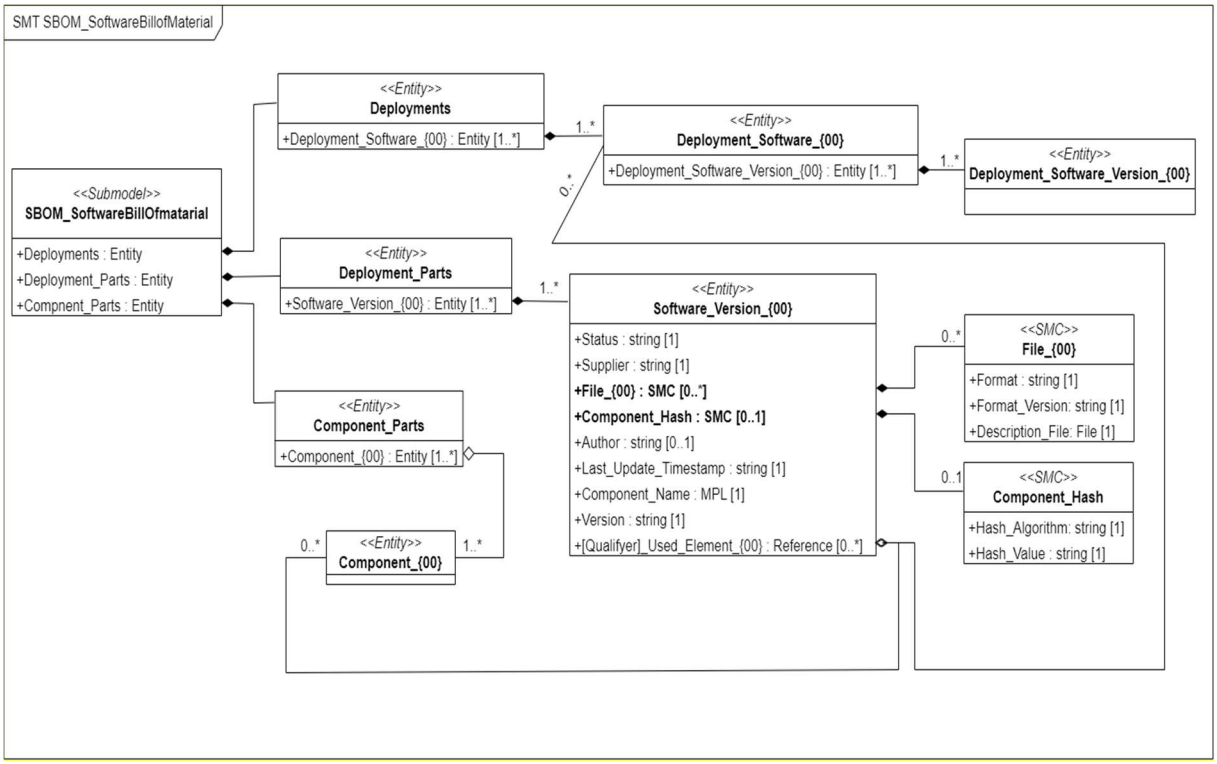

Figure 4: UML-Diagram for Submodel "Software Bill of Materials"

3.2 Properties of the Submodel “Software Bill of Material”

Table 3: Submodel “Software Bill of Material"

idShort:	**Software Bill of Material**		
Class:	**Submodel - SBOM**		
semanticId:	**[IRI] https://admin-shell.io/SBOM/1/0/Submodel/SBOM_machine**		
Parent:	**AAS**		
Explanation:	**Software Bill of Material for the machine**		
[SME type] idShort	**semanticId = [idType]value Description@en**	**[valueType] example**	**card.**
[Entity] Deployments	[IRI]https://admin-shell.io/SBOM/1/0/Deployments Process of installing and running a developed application or software on a target system or production environment	[-] -	
[Entity] Deployment_Parts	[IRI]https://admin-shell.io/SBOM/1/0/Deployments Bill of materials of a software provided by the development environment for a target environment	[-] -	
[Entity] Component_Parts	[IRI]https://admin-shell.io/SBOM/1/0/ComponentParts A listing or documentation of all components included in a software application, such as various modules, libraries, frameworks, and other components	[-] -	

3.3 Properties of the Submodel Element “Deployments”

Table 4: Submodel Element “Deployments”

idShort:	Deployments		
Class:	Entity		
semanticId:	[IRI] https://admin-shell.io/SBOM/1/0/Deployments		
Parent:	Submodel - SBOM		
Explanation:	Process of installing and running a developed application or software on a target system or production environment		
[SME type] idShort	**semanticId = [idType]value Description@en**	**[valueType] example**	**card.**
[Entity] Deployment_Software_00	[IRDI]0173-1#02-AAU368#003 Software that is installed and executed on a target system or production environment	[-] -	1..*

3.4 Properties of the Submodel Element “Deployment Software”

Table 5: Submodel Element “Deployment Software”

idShort:	Deployment_Software_00		
Class:	Entity		
semanticId:	[IRDI] 0173-1#02-AAU368#003		
Parent:	Deployments		
Explanation:	Software that is installed and executed on a target system or production environment		
[SME type] idShort	**semanticId = [idType]value Description@en**	**[valueType] example**	**card.**
[Entity] Deployment_Software_Version_00	[IRDI]0173-1#02-AAM737#002 Version of the software used by the device	[-] -	0..*

3.5 Properties of the Submodel Element “Deployment Parts”

Table 6: Submodel Element “Deployment Parts”

idShort:	Deployment_Parts		
Class:	Entity		
semanticId:	[IRI] https://admin-shell.io/SBOM/1/0/Deployments		
Parent:	Submodel - SBOM		
Explanation:	Bill of materials of a software provided by the development environment for a target environment		
[SME type] idShort	**semanticId = [idType]value Description@en**	**[valueType] example**	**card.**
[Entity] Software_Version_00	[IRDI]0173-1#02-AAM737#002 Version of the software used by the device	[-] -	1..*

3.6 Properties of the Submodel Element “Software Version”

Table 7: Submodel Element “Software Version”

idShort:	Software_Version_00		
Class:	Entity		
semanticId:	[IRDI] 0173-1#02-AAM737#002		
Parent:	Deployment_Parts		
Explanation:	Version of the software used by the device		
[SME type] idShort	**semanticId = [idType]value Description@en**	**[valueType] example**	**card.**
[Property] Status	[IRI]https://admin-shell.io/SBOM/1/0/Status Current state or situation of an object or a system	[string] active	1
[Property] Supplier	[IRDI]0173-1#02-AAO735#003 Name of supplier which provides the customer with a product or a service	[string] Sample_Manufacturer	1
[SMC] File_00	[IRI]https://admin-shell.io/SBOM/1/0/File A named collection of information stored on a computer or other electronic storage medium	[-] 3 elements	0..*
[SMC] Component_Hash	[IRI]https://admin-shell.io/SBOM/1/0/ComponentHasch A mathematical function that converts an input (usually a data set of any length) into a fixed, usually shorter value	[-] 2 elements	0..1
[Property] Author	[IRDI]0173-1#02-AAO104#002 Name of the person who generates an item	[string] Max Mustermann	0..1
[Property] Last_Update_Timestamp	[IRDI]0173-1#02-AAT465#002 Date and time of creation of the document	[string] 2023-05-17T18:00:05+01:00	1
[MLP] Component_Name	[IRI] https://admin-shell.io/SBOM/1/0/Component_Name Contains all describing information of a component	[string] Software@en	1
[Property] Version	[IRDI] 0173-1#02-AAS381#001 Stage of development of a software product at time of delivery	[string] V5.0.10	1
[Ref] Used_Element_00	[IRI]https://admin-shell.io/SBOM/1/0/UsedElement A reference to the all used operating systems, libraries, submodels etc	[-] -	0..*

3.7 Properties of the Submodel Element Collection “File”

Table 8: Submodel Element Collection “File”

idShort:	**File_00**		
Class:	**SubmodelElementCollection**		
semanticId:	**[IRI] https://admin-shell.io/SBOM/1/0/File**		
Parent:	**Software_Version_00**		
Explanation:	**A named collection of information stored on a computer or other electronic storage medium**		
[SME type] idShort	**semanticId = [idType]value Description@en**	**[valueType] example**	**card.**
[Property] Format	[IRDI]0173-1#02-AAK646#001 The way data is organized, structured, or coded to make it readable or interpretable	[string] CycloneDX	1
[Property] Format_Version	[IRI]https://admin-shell.io/SBOM/1/0/File/ForamtVersion Version of the format with which a file or data structure was created or encoded	[string] 1.4	1
[File] Description_File	[IRDI]0173-1#02-ABI504#001 MIME-Type, file name and file contents given by the file SubmodelElement	[-] /aasx/beispiel.json	1

3.8 Properties of the Submodel Element Collection “Component Hash”

Table 9: Submodel Element Collection “Component Hash”

idShort:	**Component_Hash**		
Class:	**SubmodelElementCollection**		
semanticId:	**[IRI] https://admin-shell.io/SBOM/1/0/ComponentHasch**		
Parent:	**Software_Version_00**		
Explanation:	**A mathematical function that converts an input (usually a data set of any length) into a fixed, usually shorter value**		
[SME type] idShort	**semanticId = [idType]value Description@en**	**[valueType] example**	**card.**
[Property] Hash_Algorithm	[IRDI]0173-1#02-ABH451#001 Technique or method used by the hashing algorithm	[string] SHA-256	1

[Property] Hash_Value	[IRI]https://admin-shell.io/SBOM/1/0/ComponentHasch/HashValue The result of applying a hash function to an input or set of data	[string] 98cd312d4dfc104a0a66d65023d0aade423f08951f2a9e0215e703f0c81c4f274d8e11a2db1abc30a558b820d65860c4	1

3.9 Properties of the Submodel Element “Component Parts”

Table 10: Submodel Element “Component Parts”

idShort:	**Component_Parts**		
Class:	**Entity**		
semanticId:	**[IRI] https://admin-shell.io/SBOM/1/0/ComponentParts**		
Parent:	**Submodel - SBOM**		
Explanation:	**A listing or documentation of all components included in a software application, such as various modules, libraries, frameworks, and other components**		
[SME type] idShort	**semanticId = [idType]value Description@en**	**[valueType] example**	**card.**
[Entity] Component_00	[IRDI]0173-1#02-AAN575#004 Contains the designation of the component adaptor	[-] -	1..*

4 Description of the example

2 SUPPLIER provide an AAS for their components: a temperature sensor (as Type) and a pressure sensor (as Instance). These are available as individual AAS with description of the hardware (BOM) and software (SBOM) used and are used by INTEGRATOR.

The INTEGRATOR builds a machine consisting of the following components:

- Temperature sensor with hardware and software (purchase component)
- Pressure sensor with hardware and software (purchase component)
- Human Machine Interface (HMI) with hardware and software (own development)

The machine created from the components has the aforementioned components as hardware and also has software itself that brings all the components together and regulates the output and control of the sensors via the HMI. No other hardware components are used for this; these are available via the intelligent HMI.

The INTEGRATOR first provides a type AAS with all versions of the machine. When the machine is transferred to the OPERATOR, the INTEGRATOR also creates an instance of the AAS which represents the "As-Built" state and is also transferred.

The OPERATOR receives the instance AAS (As-Built) of the machine and continues to use it internally. This makes the AAS an "As-IS" AAS that is the responsibility of the OPERATOR to maintain.

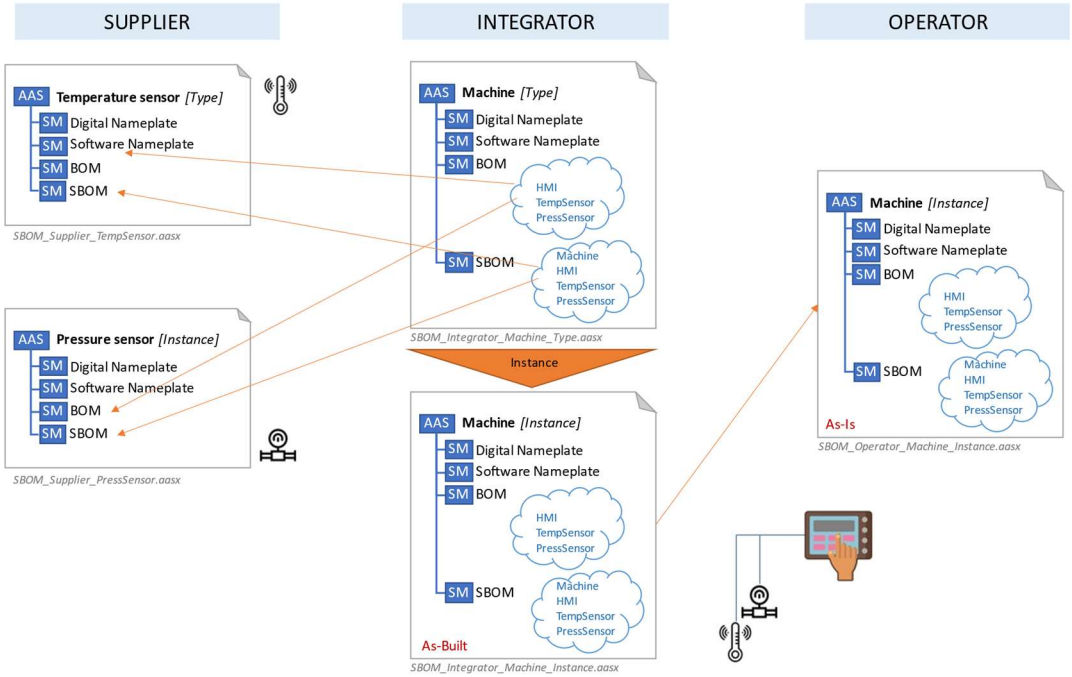

Figure 5: Big Picture of Example description

Annex A: Explanations on used table formats

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.

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

SME type Submodel	Element type
Prop	Property
MLP	MultiLanguageProperty
Range	Range
File	File
Blob	Blob
Ref	ReferenceElement
Rel	RelationshipElement
SMC	SubmodelElementCollection
Ent	Entity

- 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@de.
- 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] “Semantic interoperability: challenges in the digital transformation age”; IEC, International Electronical Commission; 2019. [Online]. Available:https://basecamp.iec.ch/download/iec-white-paper-semantic-nteroperability-challenges-in-the-digital-transformation-age-en/

[8] „Types of Software Bill of Materials (SBOM)”, April 21, 2023 Online]. Available: https://www.cisa.gov/resources-tools/resources/types-software-bill-materials-sbom