IDTA 2001-1-0 Inclusion of Module Type Package (MTP) Data into Asset Administration Shell

22 August 2022

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	Comment
2022-08-22	1.0	Release of the offical Submodel template published by IDTA

Contents

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 audience of the specification are developers and editors of MTP related products 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.

The aim of the document is to define an embedding of MTP in an AAS Submodel and establish relations to further Submodels.

1.2 Scope of the Submodel

Within the modular production, module instances are called Process Equipment Assembly (PEA). Module Type Package (MTP) is the integration technology for production modules within the modular production. MTP defines the communication interface towards the PEA and the representation of these interfaces within an MTP file. Per definition this file exclusively contains type-specific module information. [7]

The Submodels defined in this document describe the integration of PEA (instance) and MTP (type) information into an AAS. The models do not intend to cover asset aspects addressed by further Submodel definitions like technical data [8] or digital nameplate [9]. Therefore, introduced models should be used along with mentioned ones to complete the AAS of the respective asset.

1.3 Relevant standards for the Submodel template

VDI/VDE/NAMUR 2658 (NAMUR-MTP)

VDI 2770

MD 2006/42/EC

1.4 Use cases, requirements and design decisions

1.4.1 Use Case 1: Type and instance modelling of a module

As-is situation

As stated in 1.1. MTP-files include only type specific information. During the PEA integration into the Process Orchestraion Layer (POL) additional instance specific information, like the OPC UA IP-address, is required for operation. This requires separated initialization procedures for type and instance information. Furthermore, this instance information can be changed during the lifecycle of a PEA. This information cannot be retrieved in a standardized manner.

To-be situation

Classification of type and instance information as well as their referenciation during the lifecycles is one of the cornerstones of Industry 4.0 architecture. This manifests in IEC 62830 and the RAMI 4.0 model. The meta model of the AAS follows this differentiation and can be directly used to model module-specific instance and type data. The type-specific information included in the MTP file should be available as a Submodel in the AAS. This Submodel allows access to the MTP-file in order to associate the type specific information with other Submodels of the AAS. In this way the components included in a module can be e.g. referenced with component-specific documentation.

Advantages

- Embedding of MTP- and PEA-specific data into Industrie 4.0 ecosystem with further potential extensions (cf. next use case).
- Clear conceptual foundation for PEA-data and a clear separation of those from type-information of a module.
- Interoperable exchange of module-data between POLs of different vendors.
- Interoperable exchange of module-data between POL and module vendors as well as further OT management systems.

Actors

Module-vendor, POL, plant owner/operator

Sequence of events for a minimal use case

1) Module-vendor delivers one AAS for module type and module instance information each to the plant owner. PEA-specific AAS contains constant data of the module e.g. its serial number. The type specific AAS allows access to at least the MTP file of the module.
2) Plant operator enters additional PEA-specific data into AAS, e.g. PEA's OPC UA endpoint.
3) Plant operator imports both AAS into POL.
4) During the engineering and operation, the POL can change/add instance-specific data of the module.
5) POL saves the dynamic PEA-specific data into the PEA-AAS.

1.4.2 Use Case 2: Supplying Documentation for Module Types and Instances

As-is situation

The documentation of a module and its components is essential for the successful commissioning. Additionally, the documents have to be available during the operation of the module according to MD 2006/42/EC. The number of documents makes it challenging to find the correct component related files. The MTP concept does not provide an explicit possibility to include documentation. Documentation-related Submodels are currently being developed by the Industry 4.0 community. Those models are based on VDI 2770 [10]. Following the implementation of Use Case 1, a module provides instance- and type- information in separate AASs.

To-be situation

The MTP file and type specific AAS Submodel provides visualization and operation aid of a module. The documentation of the module can be divided into type and instance specific parts. Those parts contain module descriptions as well as manuals for components. Module type specific documentation such as manuals are stored in the type specific AAS whereas instance specific document like the site map of the operation location in the instance specific AAS. The documentation aspects of the AAS should provide links towards the corresponding components included in the MTP.

Additional Submodels can be easily added to the PEA AAS. The relations between those aspects and the elements inside the MTP can be represented in the AAS. This use-case focuses on the relations towards the documentation Submodel.

Advantages

- Availability of type- (e.g. module technical specs) and instance-specific documentation (e.g. commissioning protocols).
- Re-use of existing tooling like the AASXPackageExplorer to view and edit documentation data.
- MTP file stays unchanged, existing MTP tooling can be reused.

Actors

PEA vendor, POL, plant owner/operator

Sequence of events for a minimal use case

1) PEA-vendor supplies the PEA-AAS to plant operator.
2) The PEA-AAS includes references an AAS containing MTP and documentation references. Alternatively, PEA-AAS may include PEA-specific documentation within its documentation Submodel.
3) Operator imports AAS into POL.
4) Operator uses module-documentation of the module type to get semantics of module's operation.
5) Operator uses PEA-documentation to check manufacturing date of built-in component of the PEA.

1.4.3 Requirements

R1 (from UC 1): Embedding one MTP file into an AAS with kind=Type.

R2 (from UC 1): Definition and embedding of PEA-specific data in an AAS with kind=Instance. This data includes embedding constants and variables into PEA-specific AAS like serial number (constant) or OPC UA endpoint (variable).

R3 (from UC 2): Possibility to re-use further AAS-Submodels, e.g. nameplate or documentation Submodel.

R4 (from UC 2): Possibility to reference single MTP elements from defined Submodels. Example: attaching documentation from documentation Submodels to certain elements included in the MTP file.

1.4.4 Design Decisions

DD1: Embedding of MTP-file content into AAS Submodel.

Alternatives:

1) Re-modeling single MTP-contents in the AAS-Submodel or multiple Submodels. Therefore, the extraction of MTP-defined concepts and translation into the AAS meta-model is required.

2) Embedding the MTP-file as an "opaque" SubmodelElement of type "File" into the Submodel.

Decision: Alternative 2.

Advantages are:

- Existing MTP-tools can be adopted and used to import and export AASX packages. In most simple case, an AASX package needs to be extracted and the MTP file can be imported into existing tools.
- No synchronization of redundant content between MTP and AAS is needed.

Additional re-modeling of MTP-content with the help of AAS meta-model is still possible, in case further aspects of MTP need to be modeled as AAS-elements.

1.5 Approach

In the following, we assume the existence of the following two AAS:

- “AAS Type” uses module type as asset. It embeds MTP file by providing a ModuleTypePackage Submodel defined in Section 2.
- “AAS Instance” uses PEA as asset. It embeds ProcessEquipmentAssembly Submodel defined in Section 2.

To create a link between PEA and its MTP file, a “derivedFrom” reference between “AAS Instance” and “AAS Type” should be used. In case when using two AAS is infeasible for any reason, ModuleTypePackage Submodel can also be embedded directly in the “AAS Instance” to include MTP information (this approach is not recommended, due to limitation in distinguishing between type and instance information).

Furthermore, the defined Submodels included into “AAS Type” and “AAS Instance” should be used along with further Submodels covering at least the aspects:

- Identification: Properties to describe the type or instance of the process module. Possible candidate for PEA can be the nameplate model.
- Documentation: Use case 2 foresees a need for documentation embedding. The described Submodel needs to provide cross-link documentation elements with equipment that is described within MTP. Possible candidate is the documentation Submodel developed based on VDI 2770 [10].
-

1.6 Cross-AAS Relations

A “derivedFrom” reference between “AAS Instance” (embedding ProcessEquipmentAssembly Submodel defined in Section 3) and “AAS Type” (embedding ModuleTypePackage Submodel defined in Section 2).

1.7 Semantic IDs

Throughout this document, https://admin-shell.io/vdi/2658/1/0 is the generic prefix for semantic IDs used in this version of the Submodel specification. The series of guidelines VDI 2658[1] is covering all parts of MTP specification.

Under this namespace, Submodels and shared concepts like “documentationRelation” are defined. Furthermore, we systematically re-use parts of the AutomationML system unit class library of MTP definition “MTPSUCLib”.

For two specific PEA properties relating to OPC UA concepts we use https://admin-shell.io/idta/opcua-server-datasheet/1/0 proposed by IDTA work group “OPC UA Server Datasheet”.

[1] https://www.vdi.de/2658

2 Submodel for MTP Module Types

2.1 Approach

In this document, two Submodels are defined – one Submodel for module type, i.e. representing MTP, and one for module instance, i.e. representing a specific PEA.

2.2 Attributes of the Submodel instance

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

Table 1: Attributes of the Submodel instance

idShort:	ModuleTypePackage Note: The above idShort shall always be as stated.		
Class:	Submodel		
semanticId:	[IRI]https://admin-shell.io/vdi/2658/1/0/MTPSubmodel		
Kind	Instance		
Version	1		
Revision	0		
Parent:	Asset Administration Shell with module type as asset		
Explanation:	The submodel defines an entrypoint to a MTP environment containing an embedded MTP file as SubmodelElement		
[SME type]	**semanticId = [idType]value**	**[valueType]**	**card.**
idShort	**Description@en**	**example**	
[File] MTPFile	[IRI]https://admin-shell.io/vdi/2658/1/0/MTPSUCLib/ModuleTypePackage ModuleTypePackage file included as a zipped package with ending “.mtp”	MimeType = application/mtp Value = /aasx/mtp/package.mtp	1
[SMC] MTPReferences or BOMReferences or DocumentationReferences	[IRI]https://admin-shell.io/vdi/2658/1/0/MTPReferences Collection containing references to documentation documents which are associated with TagNames within the MTP file	n/a	0..*

2.3 SubmodelElements of MTPReferences

The Submodel instance this attribute needs to be set:

Table 2: SubmodelElements of MTPReferences

idShort:	MTPReferences Note, that the idShort can be chosen freely to match the needs of included MTPRefernces e.g. “DocumentationReferences” or “BOMReferenes”		
Class:	SubmodelElementCollection (SMC)		
semanticId:	[IRI]https://admin-shell.io/vdi/2658/1/0/MTPReferences		
Parent:	Submodel with idShort = ModuleTypePackage and respective semanticId or Submodel with idShort = ModuleInstance and respective semanticId		
Explanation:	This SubmodelElementCollection holds references to elements from other Submodels, e.g. included into VDI 2770 documentation Submodel		
[SME type]	**semanticId = [idType]value**	**[valueType]**	**card.**
idShort	**Description@en**	**example**	
[RelationshipElement] {arbitrary}	[IRI]https://admin-shell.io/vdi/2658/1/0/MTPReference Reference between (first) an opaque TagName within the MTP file and (second) a documentation element within a documentation Submodel In this example we link a Tag Name “M0013” from the MTP file with a documentation element “Document01” from another Submodel	first: (Submodel)(local)[IdShort]ModuleTypePackage (File)(local)[idShort]MTPFile (FragmentReference)[Custom] CAEX@ModuleTypePackage/BPXX_Freelance/CommunicationSet/InstanceList/M0013 second: (Submodel)(local)[IRI] http://example.com/id/instance/9992020020616052900001 2810 (SubmodelElementCollection)(local)[idShort]Document01	0..*

MTPReferences are used to connect elements of other Submodels with internal elements within the AML file. We propose to use four formats for the FragmentReference Key’s value to reference CAEX elements:

- CAEX@ID=’14c32ff2-f58f-45dc-b228-66a2091393dd’ – the content of the MTP file is interpreted as CAEX and the fragment path is used to locate an element with a particular ID. This will allow to connect documentation attribute to almost any elements within the MTP file.
- CAEX@ModuleTypePackage/BPXX_Freelance/CommunicationSet/InstanceList/M0013 – the content of MTP file is interpreted as CAEX and internal AML hierarchy is used to point to an element with Name “M0013”.

- MTP@TagName=’M0013’ the content of the MTP file is interpreted as CAEX and a global search for an element having an attribute with name “TagName” and value “M0013”. In case of usage of multi-language tag names this, tag name is valid only if a single language is used
- MTP@TagName=’M0013’@Language=’EN’ a variant of the above format to use in conjunction with multi-language tag names. Multiple language should be modeled using multiple references.

3 Submodel for Module Instance (Process Equipment Assembly)

3.1 Attributes of the Submodel instance

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

Table 3: Attributes of the Submodel instance

idShort:	ProcessEquipmentAssembly Note: The above idShort shall always be as stated.		
Class:	Submodel		
semanticId:	[IRI]https://admin-shell.io/vdi/2658/1/0/PEASubmodel		
Kind:	Instance		
Version:	1		
Revision:	0		
Parent:	Asset Administration Shell with module instance as asset		
Explanation:	The Submodel defines a set of PEA-properties specific to module instance Furthermore, we assume that the AAS of the PEA is referencing the AAS of module type, s.t. the relevant MTP file can be accessed by the tools. In exception cases where no AAS of MTP is available, this Submodel can also contain the MTPFile directly as defined in Section 0. In this case the MTPFile can be accessed two times, the MTP file of the Submodel instance shadows the MTPFile contained in ModuleTypePackage Submodel of referenced AAS.		
[SME type]	**semanticId = [idType]value**	**[valueType]**	**card.**
idShort	**Description@en**	**example**	
[File] MTPFile	[IRI]https://admin-shell.io/vdi/2658/1/0/MTPSUCLib/ModuleTypePackage ModuleTypePackage file included as a zipped package with ending “.zip” or “.mtp” (.mtp is preferred)	MimeType = application/mtp Value = /aasx/mtp/package.mtp	0..1
[SMC] DocumentationReferences	[IRI] https://admin-shell.io/vdi/2658/1/0/MTPReferences Collection containing references to documentation documents which are associtated with TagNames within the MTP file (defined in Section 0)	n/a	0..1
[MLP] DisplayName	[IRI]https://admin-shell.io/vdi/2658/1/0/peaSubmodel/DisplayName Operator-specific module name	[string] en, Module 42	0..1
[MLP] Description	[IRI]https://admin-shell.io/vdi/2658/1/0/PEASubmodel/Description Operator-specific module description	[string] en, Stirrer module used for process D	0..1
[SMC] SourceList	[IRI] https://admin-shell.io/vdi/2658/1/0/MTPSUCLib/CommunicationSet/SourceList	n/a	0..1

3.2 SubmodelElements of SourceList Collection

Table 4: SubmodelElements of SourceList Collection

idShort:	SourceList		
Class:	SubmodelElementCollection (SMC)		
semanticId:	[IRI]https://admin-shell.io/vdi/2658/1/0/MTPSUCLib/CommunicationSet/SourceList		
Parent:	Submodel with idShort ProcessEquipmentAssembly and respective semanticId		
Explanation:	This SMC contains descriptions to OPC UA servers of process equipment assembly. The idShort of the contained SMC could correspond to the respective InternalElement of RefBaseSystemUnitPath="MTPCommunicationSUCLib/ServerAssembly/OPCUAServer" within the MTP file.		
[SME type]	**semanticId = [idType]value**	**[valueType]**	**card.**
idShort	**Description@en**	**example**	
[SMC] {arbitrary} Example for idShort could be "FreelanceOPC UA"	[IRI]https://admin-shell.io/vdi/2658/1/0/ MTPCommunicationSUCLib/ServerAssembly/OPCUAServer	n/a	1..*

3.3 SubmodelElements of OPCUAServer-type Collection

Table 5: SubmodelElements of OPCUAServer-type Collection

idShort:	{arbitrary}		
Class:	SubmodelElementCollection (SMC)		
semanticId:	[IRI]https://admin-shell.io/vdi/2658/1/0/MTPCommunicationSUCLib/ServerAssembly/OPCUAServer		
Parent:	SMC with SourceList idshort and respective semanticId		
Explanation:	This SMC contains discovery endpoints of OPC UA servers. Note that the DiscoveryUrl is used here instead of the “Endpoint” used in MTP specification to allow a flexible OPC UA endpoint selection by OPC UA client (e.g. different OPC UA security modes). Additionally, an optional ApplcaitonUri can be included to allow OPC UA clients to select a suitable OPC UA endpoint returned by endpoint discovery.		
[SME type]	**semanticId = [idType]value**	**[valueType]**	**card.**
idShort	**Description@en**	**example**	
[Property] DiscoveryUrl{00} Example for idShort could be “DiscoveryUrl01“	[IRI] https://admin-shell.io/idta/opcua-server-datasheet/1/0/discovery-url	[string] opc.tcp://localhost:4800	1..*
[Property] ApplicationUri{00} Example for idShort could be “ApplicationUrl01”	[IRI] https://admin-shell.io/idta/opcua-server-datasheet/1/0/application-uri	[string] urn:org.com:PEA1:UA Server	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 chosen, 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] VDI/VDE/NAMUR 2658 Blatt 1: Automatisierungstechnisches Engineering modularer Anlagen in der Prozessindustrie - Allgemeines Konzept und Schnittstellen, Oktober 2019, Available: https://www.vdi.de/richtlinien/details/vdivdenamur-2658-blatt-1-automatisierungstechnisches-engineering-modularer-anlagen-in-der-prozessindustrie-allgemeines-konzept-und-schnittstellen

[8] „Generic Frame for Technical Data for Industrial Equipment in Manufacturing", Version 1.1, November 2020, Plattform Industrie 4.0 in cooperation with ZVEI, Federal Ministry for Economic Affairs and Energy (BMWi), Available: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Submodel_Templates-Asset_Administration_Shell-Technical_Data.html

[9] "ZVEI Digital Nameplate for industrial equipment", Version 1.0, November 2020, Plattform Industrie 4.0 in cooperation with ZVEI, Federal Ministry for Economic Affairs and Energy (BMWi), Available: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Submodel_Templates-Asset_Administration_Shell-digital_nameplate.html

[10] VDI 2770 Blatt 1: 2020-04 Betrieb verfahrenstechnischer Anlagen; Mindestanforderungen an digitale Herstellerinformationen für die Prozessindustrie; Grundlagen. Berlin: Beuth-

Verlag.
“Operation of process engineering plants - Minimum requirements for digital manufacturer information of process industry - Fundamentals (EN). Available: https://www.beuth.de/en/draft-technical-rule/vdi-2770-blatt-1/293855206

www.industrialdigitaltwin.org