

Part 2: Application Programming Interfaces

April 2023

Specification of the Asset Administration Shell

**S P E C I F I C A T I O N**

IDTA Number: 01002-3-0

Imprint

**Publisher**

Industrial Digital Twin Association

Lyoner Strasse 18

60528 Frankfurt am Main

Germany

https://www.industrialdigitaltwin.org/

**Text and editing**

Industrial Digital Twin Association

Lyoner Strasse 18

60528 Frankfurt am Main

Germany

**Design and production**

Industrial Digital Twin Association, Frankfurt am Main

**Status**

April 2023

**Illustrations**

Plattform Industrie 4.0; Anna Salari, Publik. Agentur für Kommunikation GmbH, designed by Publik. Agentur für Kommunikation GmbH

Table of Contents

[1 Preamble 10](#_Toc132377507)

[1.1 Editorial Notes 10](#_Toc132377508)

[1.2 Metamodel Versions 10](#_Toc132377509)

[1.3 Scope of this Document 10](#_Toc132377510)

[1.4 Structure of the Document 10](#_Toc132377511)

[2 Terms, Definitions and Abbreviations 11](#_Toc132377512)

[2.1 Terms and Definitions 11](#_Toc132377513)

[2.2 Abbreviations 13](#_Toc132377514)

[3 Introduction 14](#_Toc132377515)

[4 General 15](#_Toc132377516)

[4.1 Services, Interfaces and Interface Operations 15](#_Toc132377517)

[4.2 Design Principles 16](#_Toc132377518)

[4.3 Semantic References for Operations 18](#_Toc132377519)

[4.4 References and Keys 19](#_Toc132377520)

[4.5 Special Parameters 19](#_Toc132377521)

[4.6 Relation of Interfaces 20](#_Toc132377522)

[5 Asset Administration Shell Interfaces 23](#_Toc132377523)

[5.1 General 23](#_Toc132377524)

[5.2 Asset Administration Shell Interface and Operations 23](#_Toc132377525)

[5.2.1 Asset Administration Shell Interface 23](#_Toc132377526)

[5.2.2 Operation GetAssetAdministrationShell 24](#_Toc132377527)

[5.2.3 Operation PutAssetAdministrationShell 24](#_Toc132377528)

[5.2.4 Operation GetAllSubmodelReferences 25](#_Toc132377529)

[5.2.5 Operation PostSubmodelReference 25](#_Toc132377530)

[5.2.6 Operation DeleteSubmodelReference 26](#_Toc132377531)

[5.2.7 Operation GetAssetInformation 26](#_Toc132377532)

[5.2.8 Operation PutAssetInformation 27](#_Toc132377533)

[5.2.9 Operation GetThumbnail 27](#_Toc132377534)

[5.2.10 Operation PutThumbnail 28](#_Toc132377535)

[5.2.11 Operation DeleteThumbnail 28](#_Toc132377536)

[5.3 Submodel Interface and Operations 29](#_Toc132377537)

[5.3.1 Submodel Interface 29](#_Toc132377538)

[5.3.2 Operation GetSubmodel 30](#_Toc132377539)

[5.3.3 Operation GetAllSubmodelElements 31](#_Toc132377540)

[5.3.4 Operation GetSubmodelElementByPath 31](#_Toc132377541)

[5.3.5 Operation GetFileByPath 32](#_Toc132377542)

[5.3.6 Operation PutFileByPath 32](#_Toc132377543)

[5.3.7 Operation DeleteFileByPath 33](#_Toc132377544)

[5.3.8 Operation PutSubmodel 33](#_Toc132377545)

[5.3.9 Operation PatchSubmodel 34](#_Toc132377546)

[5.3.10 Operation PostSubmodelElement 34](#_Toc132377547)

[5.3.11 Operation PostSubmodelElementByPath 35](#_Toc132377548)

[5.3.12 Operation PutSubmodelElementByPath 36](#_Toc132377549)

[5.3.13 Operation PatchSubmodelElementByPath 37](#_Toc132377550)

[5.3.14 Operation GetSubmodelElementValueByPath 37](#_Toc132377551)

[5.3.15 Operation PatchSubmodelElementValueByPath 38](#_Toc132377552)

[5.3.16 Operation DeleteSubmodelElementByPath 38](#_Toc132377553)

[5.3.17 Operation InvokeOperationSync 40](#_Toc132377554)

[5.3.18 Operation InvokeOperationAsync 41](#_Toc132377555)

[5.3.19 Operation GetOperationAsyncStatus 42](#_Toc132377556)

[5.3.20 Operation GetOperationAsyncResult 42](#_Toc132377557)

[5.4 Serialization Interface and Operations 43](#_Toc132377558)

[5.4.1 Serialization Interface 43](#_Toc132377559)

[5.4.2 Operation GenerateSerializationByIds 43](#_Toc132377560)

[5.5 AASX File Server Interface and Operations 44](#_Toc132377561)

[5.5.1 AASX File Server Interface 44](#_Toc132377562)

[5.5.2 Operation GetAllAASXPackageIds 45](#_Toc132377563)

[5.5.3 Operation GetAASXByPackageId 45](#_Toc132377564)

[5.5.4 Operation PostAASXPackage 46](#_Toc132377565)

[5.5.5 Operation PutAASXPackageById 47](#_Toc132377566)

[5.5.6 Operation DeleteAASXPackageById 47](#_Toc132377567)

[6 Registration Interfaces 48](#_Toc132377568)

[6.1 General 48](#_Toc132377569)

[6.2 Asset Administration Shell Registry Interface and Operations 48](#_Toc132377570)

[6.2.1 Asset Administration Shell Registry Interface 48](#_Toc132377571)

[6.2.2 Operation GetAllAssetAdministrationShellDescriptors 49](#_Toc132377572)

[6.2.3 Operation GetAssetAdministrationShellDescriptorById 49](#_Toc132377573)

[6.2.4 Operation PostAssetAdministrationShellDescriptor 50](#_Toc132377574)

[6.2.5 Operation PutAssetAdministrationShellDescriptorById 50](#_Toc132377575)

[6.2.6 Operation DeleteAssetAdministrationShellDescriptorById 51](#_Toc132377576)

[6.3 Submodel Registry Interface and Operations 51](#_Toc132377577)

[6.3.1 Submodel Registry Interface 51](#_Toc132377578)

[6.3.2 Operation GetAllSubmodelDescriptors 52](#_Toc132377579)

[6.3.3 Operation GetSubmodelDescriptorById 52](#_Toc132377580)

[6.3.4 Operation PostSubmodelDescriptor 53](#_Toc132377581)

[6.3.5 Operation PutSubmodelDescriptorById 53](#_Toc132377582)

[6.3.6 Operation DeleteSubmodelDescriptorById 54](#_Toc132377583)

[7 Repository Interfaces 55](#_Toc132377584)

[7.1 General 55](#_Toc132377585)

[7.2 Asset Administration Shell Repository Interface and Operations 55](#_Toc132377586)

[7.2.1 Asset Administration Shell Repository Interface 55](#_Toc132377587)

[7.2.2 Operation GetAllAssetAdministrationShells 56](#_Toc132377588)

[7.2.3 Operation GetAssetAdministrationShellById 57](#_Toc132377589)

[7.2.4 Operation GetAllAssetAdministrationShellsByAssetId 58](#_Toc132377590)

[7.2.5 Operation GetAllAssetAdministrationShellsByIdShort 59](#_Toc132377591)

[7.2.6 Operation PostAssetAdministrationShell 60](#_Toc132377592)

[7.2.7 Operation PutAssetAdministrationShellById 60](#_Toc132377593)

[7.2.8 Operation DeleteAssetAdministrationShellById 61](#_Toc132377594)

[7.3 Submodel Repository Interface and Operations 61](#_Toc132377595)

[7.3.1 Submodel Repository Interface 61](#_Toc132377596)

[7.3.2 Operation GetAllSubmodels 62](#_Toc132377597)

[7.3.3 Operation GetSubmodelById 62](#_Toc132377598)

[7.3.4 Operation GetAllSubmodelsBySemanticId 63](#_Toc132377599)

[7.3.5 Operation GetAllSubmodelsByIdShort 64](#_Toc132377600)

[7.3.6 Operation PostSubmodel 65](#_Toc132377601)

[7.3.7 Operation PutSubmodelById 65](#_Toc132377602)

[7.3.8 Operation PatchSubmodelById 66](#_Toc132377603)

[7.3.9 Operation DeleteSubmodelById 66](#_Toc132377604)

[7.4 Concept Description Repository Interface and Operations 67](#_Toc132377605)

[7.4.1 Concept Description Repository Interface 67](#_Toc132377606)

[7.4.2 Operation GetAllConceptDescriptions 68](#_Toc132377607)

[7.4.3 Operation GetConceptDescriptionById 68](#_Toc132377608)

[7.4.4 Operation GetAllConceptDescriptionsByIdShort 69](#_Toc132377609)

[7.4.5 Operation GetAllConceptDescriptionsByIsCaseOf 70](#_Toc132377610)

[7.4.6 Operation GetAllConceptDescriptionsByDataSpecificationReference 71](#_Toc132377611)

[7.4.7 Operation PostConceptDescription 72](#_Toc132377612)

[7.4.8 Operation PutConceptDescriptionById 72](#_Toc132377613)

[7.4.9 Operation DeleteConceptDescriptionById 73](#_Toc132377614)

[8 Publish and Discovery Interfaces 74](#_Toc132377615)

[8.1 General 74](#_Toc132377616)

[8.2 Asset Administration Shell Basic Discovery Interface and Operations 74](#_Toc132377617)

[8.2.1 Asset Administration Shell Basic Discovery Interface 74](#_Toc132377618)

[8.2.2 Operation GetAllAssetAdministrationShellIdsByAssetLink 75](#_Toc132377619)

[8.2.3 Operation GetAllAssetLinksById 76](#_Toc132377620)

[8.2.4 Operation PostAllAssetLinksById 77](#_Toc132377621)

[8.2.5 Operation DeleteAllAssetLinksById 77](#_Toc132377622)

[9 Description Interface 78](#_Toc132377623)

[9.1.1 Self-Description Interface 78](#_Toc132377624)

[9.1.2 Operation GetSelfDescription 78](#_Toc132377625)

[10 Data Types for Payload 79](#_Toc132377626)

[10.1 General 79](#_Toc132377627)

[10.2 Metamodel Specification Details 79](#_Toc132377628)

[10.2.1 Descriptor 79](#_Toc132377629)

[10.2.2 AssetAdministrationShellDescriptor 80](#_Toc132377630)

[10.2.3 SubmodelDescriptor 81](#_Toc132377631)

[10.2.4 Endpoint 81](#_Toc132377632)

[10.2.5 ProtocolInformation 83](#_Toc132377633)

[10.2.6 ServiceDescription 85](#_Toc132377634)

[10.2.7 Simple Data Types 87](#_Toc132377635)

[10.2.8 Primitive Data Types 87](#_Toc132377636)

[10.2.9 Status Code, Error Handling & Result Messages 88](#_Toc132377637)

[10.2.10 File Content 92](#_Toc132377638)

[11 Basic Operation Parameters 93](#_Toc132377639)

[11.1 General 93](#_Toc132377640)

[11.2 SerializationModifiers in Operations 93](#_Toc132377641)

[11.3 Applicability of SerializationModifiers 94](#_Toc132377642)

[11.4 Serialization in Specified Formats (SerializationModifier *Content*) 95](#_Toc132377643)

[11.4.1 General 95](#_Toc132377644)

[11.4.2 ValueOnly-Serialization in JSON 96](#_Toc132377645)

[11.4.3 JSON-Schema for the ValueOnly-Serialization 105](#_Toc132377646)

[11.4.4 IdShortPath Serialization 110](#_Toc132377647)

[12 HTTP/REST API 112](#_Toc132377648)

[12.1 General 112](#_Toc132377649)

[12.2 Design Decisions 113](#_Toc132377650)

[12.3 API Versioning 114](#_Toc132377651)

[12.4 Addressing Resources 115](#_Toc132377652)

[12.5 Metadata Objects 117](#_Toc132377653)

[12.6 Pagination 118](#_Toc132377654)

[12.7 Payload 120](#_Toc132377655)

[12.8 Modifier Constraints 120](#_Toc132377656)

[12.9 Mapping of Operations 120](#_Toc132377657)

[12.9.1 Asynchronous Invocation of the SubmodelElement “Operation” 126](#_Toc132377658)

[12.10 Mapping of Status Codes 127](#_Toc132377659)

[12.11 Additional Data Types for Payload for HTTP/REST 128](#_Toc132377660)

[12.11.1 PackageDescription 128](#_Toc132377661)

[12.12 Service Specifications and Profiles 129](#_Toc132377662)

[12.12.1 Profiles 130](#_Toc132377663)

[12.12.2 Asset Administration Shell Service Specification 130](#_Toc132377664)

[12.12.3 Submodel Service Specification 133](#_Toc132377665)

[12.12.4 AASX File Server Service Specification 136](#_Toc132377666)

[12.12.5 Asset Administration Shell Registry Service Specification 137](#_Toc132377667)

[12.12.6 Submodel Registry Service Specification 140](#_Toc132377668)

[12.12.7 Discovery Service Specification 142](#_Toc132377669)

[12.12.8 Asset Administration Shell Repository Service Specification 143](#_Toc132377670)

[12.12.9 Submodel Repository Service Specification 147](#_Toc132377671)

[12.12.10 Concept Description Repository Service Specification 152](#_Toc132377672)

[12.13 Interactions 152](#_Toc132377673)

[12.14 Security 155](#_Toc132377674)

[12.15 API Code Generation 157](#_Toc132377675)

[13 Summary and Outlook 158](#_Toc132377676)

[Annex A. Templates used for Specification 159](#_Toc132377677)

[Annex B. ValueOnly-Serialization Example 163](#_Toc132377678)

[Annex C. SerializationModifier Examples 166](#_Toc132377679)

[1. Description 166](#_Toc132377680)

[2. Examples for GET Operations 166](#_Toc132377681)

[3. Examples for PATCH Operations 170](#_Toc132377682)

[Annex D. Backus-Naur-Form 175](#_Toc132377683)

[Annex E. Bibliography 176](#_Toc132377684)

[Change Notes 177](#_Toc132377685)

[1. General 177](#_Toc132377686)

[2. Interface Changes w.r.t. V1.0RC03 to V3.0 177](#_Toc132377687)

[3. Operation Changes w.r.t. V1.0RC03 to V3.0 178](#_Toc132377688)

[4. Interface Changes w.r.t. V1.0RC02 to V1.0RC03 178](#_Toc132377689)

[5. Operation Changes w.r.t. V1.0RC02 to V1.0RC03 179](#_Toc132377690)

[6. Interface Changes w.r.t. V1.0RC01 to V1.0RC02 179](#_Toc132377691)

[7. Operation Changes w.r.t. V1.0RC01 to V1.0RC02 181](#_Toc132377692)

Table of Figures

[Figure 1 Types of Information Exchange via Asset Administration Shells 14](#_Toc132380113)

[Figure 2 Services, Interfaces & APIs and Operations 16](#_Toc132380114)

[Figure 3 Retrieval of Asset-related Information by AAS and Submodels 20](#_Toc132380115)

[Figure 4 - Generic URL Scheme for AAS API Versioning 114](#_Toc132380116)

[Figure 5 Example Hierarchy of Submodel Elements 116](#_Toc132380117)

[Figure 6 Sequence for asynchronous invocations of the SubmodelElement 'Operation' 126](#_Toc132380118)

[Figure 7 Interactions for Client Applications using AAS and Submodel Interfaces   
(independent Submodel Registry) 153](#_Toc132380119)

[Figure 8 Interaction for Client Application using AAS and Submodel Interfaces   
(included Submodel Registry) 154](#_Toc132380120)

[Figure 9 Interaction for Client Application using AAS and Submodels (for HTTP API Operations) 155](#_Toc132380121)

[Figure 10 The private\_key\_certchain\_jwt Method [...download service] 156](#_Toc132380122)

Table of Tables

[Table 1 Special Parameters 19](#_Toc132380092)

[Table 2 Simple Data Types used for API-specific Classes 87](#_Toc132380093)

[Table 3 Primitive Data Types used for the API-specific Classes 87](#_Toc132380094)

[Table 4 Status Codes 88](#_Toc132380095)

[Table 5 Level Parameters 93](#_Toc132380096)

[Table 6 Content Parameters 93](#_Toc132380097)

[Table 7 Extent Parameters 94](#_Toc132380098)

[Table 8 Applicability of SerializationModifiers 94](#_Toc132380099)

[Table 9 Mapping of Data Types in ValueOnly-Serialization 98](#_Toc132380100)

[Table 10 Children of certain objects 116](#_Toc132380101)

[Table 11 Metadata Attributes 117](#_Toc132380102)

[Table 12 AssetAdministrationShell JSON Serialization Example 118](#_Toc132380103)

[Table 13 AssetAdministrationShell Metadata JSON Serialization Example 118](#_Toc132380104)

[Table 14 Parameters for Pagination 119](#_Toc132380105)

[Table 15 Mapping of the generic Interface Operations to HTTP API Operations 122](#_Toc132380106)

[Table 16 Status Code Mapping for HTTP 127](#_Toc132380107)

[Table 17 Overview of Service Specifications and the Contained APIs 129](#_Toc132380108)

[Table 18 Interface Description 159](#_Toc132380109)

[Table 19 Operation Description 159](#_Toc132380110)

[Table 20 Data Types for Payload Description 161](#_Toc132380111)

[Table 21 Enumeration Description 162](#_Toc132380112)

# Preamble

## Editorial Notes

This document (version 3.0) was produced from November 2021 to May 2023 by the joint sub working group "Asset Administration Shell" of the working group "Reference Architectures, Standards and Norms" of the Plattform Industrie 4.0 and the working group "Open Technology" of the Industrial Digital Twin Association (IDTA). It is the first release published by the Industrial Digital Twin Association.

Earlier versions of this document were release candidates and used the version 1.0. It has been decided in the meantime that this first release will start with version 3.0, in line with the related release of the metamodel.

Version 1.0 RC02 of this document was developed from November 2020 to November 2021 by the joint working groups “Asset Administration Shell” and “Infrastructure of the Asset Administration Shell” of the Plattform Industrie 4.0 working group “Reference Architectures, Standards and Norms“.

Version 1.0 RC01 of this document was developed from December 2019 to November 2020 by the sub working groups “Asset Administration Shell” and “Infrastructure of the Asset Administration Shell” of the Plattform Industrie 4.0 working group “Reference Architectures, Standards and Norms“.

This document is Part 2 of the document series “Specification of the Asset Administration Shell”.

This specification is versioned using Semantic Versioning 2.0.0 and follows the semver specification [4].

## Metamodel Versions

This document (version 3.0) uses the following parts of the “Specification of the Asset Administration Shell” series:

* Part 1: Metamodel in version 3.0 [1]
* Part 3a: Data Specification – IEC 61360 in version 3.0 [2]
* Part 5: Package File Format (AASX) in version 3.0 [3]

## Scope of this Document

This document specifies the interfaces as well as the APIs in selected technologies for the Asset Administration Shells and its submodels.

## Structure of the Document

Clause 3 gives an introduction to the topic. General topics are discussed in Clause 4. The technology-neutral specification of the interfaces of the Asset Administration Shell can be found in Clauses 5 to 11.

Clause 12 defines the API specification for HTTP/REST. Annex B gives an example for the ValueOnly-serialization of the payload.

Clause 13 provides a summary and outlook.

The tables used to specify operations and interfaces are explained in the annex. Additionally, non-normative examples are given to illustrate in particular the different serialization alternatives.

# Terms, Definitions and Abbreviations

## Terms and Definitions

Please note: the definitions of terms are only valid in a certain context. This glossary applies only within the context of this document.

If available, definitions were taken from IEC 63278-1 DRAFT, July 2022.

API

specification of the set of operations and events that forms an API in a selected technology

API Operation

specification of the operations (procedures) that may be called through an API

Asset Administration Shell (AAS)

standardized digital representation of an asset

Note: Asset Administration Shell and Administration Shell are used synonymously.

* [SOURCE: IEC 63278-1, note added]

Interface

defined connection point of a functional unit which can be connected to other functional units

Note 1: “defined” means that the requirements and the assured properties of this connection point are described.

Note 2: the combination of interfaces of function units is also called an interface.

Note 3: in an information system, the defined exchange of information takes place at this point.

Note 4: an interface places certain requirements on the connection that is to be made.

Note 5: an interface demands certain features.

[Source: Glossary Industrie 4.0

DUDEN (modified)

ISO/IEC 13066-1:2011(en), 2.15 (modified)

DIN EN 60870-5-6:2009-11 (modified)

DIN IEC 60625-1:1981-05 (modified)]

Interface Operation

interface operations define interaction patterns via the specified interface

operation

executable realization of a function

Note 1: the term method is synonymous to operation.

Note 2: an operation has a name and a list of parameters [ISO 19119:2005, 4.1.3].

* [SOURCE: Glossary Industrie 4.0, editorial changes]

service

Demarcated scope of functionality which is offered by an [entity](https://www.plattform-i40.de/PI40/Redaktion/EN/Glossary/E/entity_glossary.html) or organization via [interfaces](https://www.plattform-i40.de/PI40/Redaktion/EN/Glossary/I/interface_glossary.html)

Note: one or multiple operations can be assigned to one service.

* [SOURCE: Glossary Industrie 4.0]

service specification

specification of a service according to the notation, architectural style and constraints of a selected technology

Note: one or multiple API Operations can be assigned to one service specification.

Submodel

representation of an aspect of an asset

* [SOURCE: IEC 63278-1]

SubmodelElement

element of a Submodel

* [SOURCE: IEC 63278-1]

## Abbreviations

| **Abbreviation** | **Description** |
| --- | --- |
| AAS | Asset Administration Shell |
| AASX | Package file format for the AAS |
| AML | AutomationML |
| API | Application Programming Interface |
| BITKOM | Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V. |
| BLOB | Binary Large Object |
| CDD | Common Data Dictionary |
| GUID | Globally unique identifier |
| ID | Identifier |
| IDTA | Industrial Digital Twin Association |
| IEC | International Electrotechnical Commission |
| IRDI | International Registration Data Identifier |
| ISO | International Organization for Standardization |
| JSON | JavaScript Object Notation |
| MIME | Multipurpose Internet Mail Extensions |
| OPC | Open Packaging Conventions (ECMA-376, ISO/IEC 29500-2) |
| OPCF | OPC Foundation |
| OPC UA | OPC Unified Architecture |
| PDF | Portable Document Format |
| RAMI4.0 | Reference Architecture Model Industrie 4.0 |
| RDF | Resource Description Framework |
| REST | Representational State Transfer |
| RFC | Request for Comment |
| ROA | Resource Oriented Architecture |
| SOA | Service Oriented Architecture |
| UML | Unified Modeling Language |
| URI, URL, URN | Uniform Resource Identifier, Locator, Name |
| VDE | Verband der Elektrotechnik Elektronik Informationstechnik e. V. |
| VDI | Verein Deutscher Ingenieure e.V. |
| VDMA | Verband Deutscher Maschinen- und Anlagenbau e.V. |
| W3C | World Wide Web Consortium |
| XML | eXtensible Markup Language |
| ZIP | archive file format that supports lossless data compression |
| ZVEI | Zentralverband Elektrotechnik- und Elektronikindustrie e. V. |

# Introduction

This document defines APIs for enabling the access to the information provided by an Asset Administration Shell. The underlying information model is as defined in [1].

Since an API can be specified in different technologies like HTTP/REST, MQTT and OPC UA, the specification offers a technology-neutral specification of the interfaces.

While Part 5 of the specification series of the Asset Administration Shell [3] mainly considered file exchange, this specification focuses on the API that allows online access to information provided by the AAS (see Figure 1).



Figure 1 Types of Information Exchange via Asset Administration Shells

# General

## Services, Interfaces and Interface Operations

This document uses the Industrie 4.0 Service Model illustrated in

Figure 2 for a uniform understanding and naming. It basically distinguishes between associated concepts on several levels (from left to right):

* technology-neutral level: concepts that are independent from selected technologies;
* technology-specific level: concepts that are instantiated for a given technology and/or architectural style (e.g. HTTP/REST, OPC UA, MQTT);
* implementation level: concepts that are related to an implementation architecture that comprises one or more technologies (e. g. C#, C++, Java, Python);
* runtime level: concepts that are related to identifiable components in an operational Industry 4.0 system.

This document deals with the concepts of the technology-neutral and technology-specific level. However, to avoid terminological and conceptual misunderstandings, the whole Industrie 4.0 Service Model is provided here.

The technology-neutral level comprises the following concepts:

* **Service**: a service describes a demarcated scope of functionality (including its informational and non-functional aspects), which is offered by an entity or organization via [interfaces](https://www.plattform-i40.de/PI40/Redaktion/EN/Glossary/I/interface_glossary.html).
* **Interface**: this is the most important concept as it is understood to be the unit of reusability across services and the unit of standardization when mapped to application programming interfaces (API) in the technology-specific level. One interface may be mapped to several APIs depending on the technology and architectural style used, e.g. HTTP/REST or OPC UA, whereby these API mappings also need to be standardized for the sake of interoperability.
* **Interface-Operation**: interface operations define interaction patterns via the specified interface.

The technology-specific level comprises the following concepts:

* **Service Specification**: specification of a service according to the notation, architectural style, and constraints of a selected technology. Among others, it comprises and refers to the list of APIs that forms this service specification. These may be I4.0-defined standard APIs but also other, proprietary APIs.

Note: such a technology-specific service specification may be but does not have to be derived from the “service” described in the technology-neutral form. It is up to the system architect and service engineer to tailor the technology-specific service according to the needs of the use cases.

* **API**: specification of the set of operations and events that forms an API in a selected technology. It is derived from the interface description on the technology-neutral level. Hence, if there are several selected technologies, one interface may be mapped to several APIs.
* **API-Operation**: specification of the operations (procedures) that may be called through an API. It is derived from the interface operation description on the technology-neutral level. When selecting technologies, one interface operation may be mapped to several API-operations; several interface operations may also be mapped to the same API-operation.

The implementation level comprises the following concepts:

* **Service-Implementation**: service realized in a selected implementation language following the specification in the Service Specification description on the technology-specific level.
* **API-Implementation**: set of operations realized in a selected implementation language following the specification in the API description on the technology-specific level.
* **API-Operation-Implementation**: concrete realization of an operation in a selected implementation language following the specification in the API-Operation description on the technology-specific level.

The runtime level comprises the following concepts:

* **Service-Instance**: instance of a Service-Implementation including its API-Instances for communication. Additionally, it has an identifier to be identifiable within a given context.
* **API-Instance**: instance of an API-Implementation which has an endpoint to get the information about this instance and the related operations.
* **API-Operation-Instance**: instance of an API-Operation-Implementation which has an endpoint to get invoked.

Figure 2 Services, Interfaces & APIs and Operations

One important message from the Industrie 4.0 Service Model is that it is the level of the interface (mapped to technology-specific APIs) that

* provides the unit of reusability,
* is the foundation for interoperable services, and
* provides the reference unit for compliance statements.

Therefore, this document defines the interfaces and operations which are needed for interaction regarding the elements of the Asset Administration Shell metamodel starting with Clause 5.

## Design Principles

The operations of the interfaces follow a resource-oriented approach which is close to general REST principles but not as strict in every situation. The approach consists of the three main agreements:

* Stateless: the API is stateless. Each operation is independent. The server is always consistent after each operation.
* Resources (nouns): each resource is a clearly defined noun. This means that it has a specific name and its relation to other nouns is defined. The nouns and the relationships between them are taken from the list of referable objects of “Specification of the Asset Administration Shell Part 1” and their relationships. Clause 10.2 gives an additional list of resources.
* Methods (verbs): a small set of standard REST methods (GET, POST, PUT, DELETE) is used to describe the semantic of the most common operations. There are only a few exceptions for situations where the standard methods do not fit (e.g. GETALL, SET, INVOKE).

The methods are:

* GET: a GET returns a single resource based on the resource identifier which is the identifier [1] for identifiables and the idShortPath for referables.
* GETALL: returns a list of resources based on optionally available parameters such as filters.
* POST: creates a new resource. The identifier of the resource is part of the resource description. This is necessary because the id of identifiables is globally unique and should be the identifier for the object in every system. This implies that the creation of an identifiable is idempotent. There shall never be more than one identifiable with the same ID in one system. For example, trying to post the same AAS object twice will not create two AAS resources.
* PUT: replaces an existing resource.
* PATCH: updates an existing resource. The content to be replaced will be defined by the given SerializationModifiers, e.g. content=value provides the ValueOnly-serialization to update all values in the existing resource. The structure of the existing resource on the server and of the content given by the PATCH must be the same.

Note: values remain unchanged with content=metadata.

* DELETE: deletes a resource based on a given identifier.
* SET: sets the value of an object, e.g. the value of a Property.
* INVOKE: invokes an operation at a specified path.

Note: these methods are intended for the naming of interfaces as described in

Figure 2. They shall not be interpreted as new protocol methods, e.g. on HTTP level.

Naming rules for operations:

The following rules shall apply for the operation names in Asset Administration Shell Interface, Submodel Interface, Shell Repository Interface, Submodel Repository Interface, Concept Description Repository Interface:

|  |  |  |
| --- | --- | --- |
| <Interface Operation> | ::= | <Method Verb><Model Element Name>[<Modifier>]  ["By"<By-Qualifier>] |
| <Method Verb> | ::= | "Get" | "GetAll" | "Put" | "Post" | "Patch" | "Delete" | "Set" | "Invoke" |
| <Model Element Name> | ::= | "AssetAdministrationShell"["s"] | "SubmodelReference" ["s"] | "AssetInformation" | "Submodel"["s"] | "SubmodelElement"["s"] | "ConceptDescription"["s"] |
| <Modifier> | ::= | "Value" | "IdShortPath" | "Reference" |
| <By-Qualifier> | ::= | "Id" | "SemanticId" | "ParentPathAndSemanticId" | "Path" | "AssetId" | "IdShort" | "IsCaseOf" | "DataSpecificationReference" |

Examples:

GetSubmodel has method verb “Get” and element name “Submodel”.

GetAllSubmodelElementsByPath has method verb “GetAll” and element name “SubmodelElements” plus a by-qualifier “Path”.

## Semantic References for Operations

The operations of this document need unique identifiers to reach a common understanding and allow all involved parties to reference the same things. These identifiers need to be globally unique and understandable by the community and implementing systems. Furthermore, the identifiers need to support a versioning scheme for future updates and extensions of the metamodel. The identifiers defined in this document are reused in related resources, for instance REST API operations or in self-descriptions of implementing services.

Internationalized Resource Identifiers (IRIs), Uniform Resource Identifiers (URIs) [5] in particular, and the requirements of DIN SPEC 91406 [6], serve as the basic format. Further design decisions include ‘https’ as the URI scheme, and the controlled domain name ‘admin-shell.io’ as the chosen authority. Both decisions guarantee the interoperability of the identifiers and their durability, since URIs are generally well-known and proven, while the domain is controlled and served through the Plattform Industrie 4.0. All identifiers included in the ‘admin-shell.io’ domain are described in a lightweight catalogue in the form of markdown documents; they are continuously maintained and updated[[1]](#footnote-2). The catalogue itself is structured in several sub-namespaces specified by the first path parameter. All URIs of this document reflect entities of the core metamodel, which are contained in the sub-namespace identified with the ‘/aas/API’ path.

The described identifiers appear mainly in the semanticId field of every class and operation. They are required since the class name is not necessarily constant over time. The respective semanticIds, however, guarantee the unique and certain relation between a reference and the referenced class or operation. The URIs are constructed as follows (compare to Clause Semantic Identifiers for Metamodel and Data Specifications in Part 1 [1]).

Note 1: version information is explicitly included in each identifier.

Note 2: even though the usage of the ‘https’ scheme might indicate URLs, all identifiers are regarded as URI look ups; dereferencing them cannot be expected.

The following grammar is used to create valid identifiers:

<Identifier> ::= <Namespace>"/aas/API/"<OperationName>"/"<Version>

<Namespace> ::= "https://admin-shell.io

<OperationName> ::= {<Character>}+

<Version> ::= {<Digit>}+"/"{<Digit>}+["/"{<Character>}+]

<Digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

<Character> ::= an unreserved character permitted by DIN SPEC 91406

? ::= zero or one

+ ::= one or more

Examples for valid identifiers:

* https://admin-shell.io/aas/API/GetSubmodel/1/23
* https://admin-shell.io/aas/API/GetAllSubmodelElements/1/0/RC03
* https://admin-shell.io/aas/API/GetAllSubmodelElements/3/0

Examples for invalid identifiers:

* http://admin-shell.io/API/GetSubmodel/1/0  
  The scheme is different to ‘https’, and the ‘aas’ path segment is missing
* https://admin-shell.io/aas/API/GetSubmodel  
  Version information is missing
* https://admin-shell.io/aas/API/GetSubmodel/1/0#0173-%20ABC#001  
  The URI includes DIN SPEC 91406-reserved (#) and impermissible (%) characters

## References and Keys

The concept of references is introduced in Part 1 of the series “ Specification of the Asset Administration Shell” [1].

When defining interfaces, a distinction is made between relative references and absolute references.

Absolute references require a global unique id as starting point of the reference to be resolvable. In this case the type “Reference” is used.

Relative references do not start with a global unique id. Instead, it is assumed that the context is given and unique. In this case, the key list only contains keys with *Key/type* that references a non-identifiable referable (e.g. a Property, a Range, a RelationshipElement, etc.).

## Special Parameters

The following table describes special parameters used for consistency throughout the document.

Table 1 Special Parameters

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| path | IdShort-Path via relative Reference/Keys to a submodel element |
| OperationHandle | The returned handle of an operation’s asynchronous invocation used to request the current state of the operation’s execution |
| OperationResult | The returned result of an operation’s invocation |
| SerializationModifier | Defines the format of the input or the output of an operation |
| SerializationFormat | Determines the format of serialization, i.e. JSON, XML, RDF, AML, etc. |
| ShellDescriptor | Object containing the Asset Administration Shell’s identification and endpoint information |
| SubmodelDescriptor | Object containing the Submodel’s identification and endpoint information |
| SpecificAssetId | The name of the specific asset identifier or the predefined name “*globalAssetId*” that would refer to the *AssetInformation/globalAssetId* |
| SemanticId | Identifier of the semantic definition |

## Relation of Interfaces

The following chapters define several interfaces, which work together as a system and support different deployment scenarios.

There are three major components of the overall system:

1. Repositories store the data of Asset Administration Shells, Submodels, and Concept Descriptions,
2. Registries are “directories” which store AAS-IDs and Submodel-IDs together with the related endpoints (typically a URL-path into a repository or to a single AAS/Submodel),
3. discovery (servers) supports a fast search and only store copies of essential information, i.e. key value pairs to find IDs by other IDs.

Figure 3 shows a typical sequence. Discovery finds the AAS-ID for a given Asset-ID. A Registry provides the endpoint for a given AAS-ID. Such an endpoint for an AAS and the related Submodel-IDs make the submodels with their submodelElements accessible.

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

Figure 3 Retrieval of Asset-related Information by AAS and Submodels

The Asset Administration Shell model is an asset-oriented model.

An Asset-ID may be retrieved e.g. by a QRCODE on the asset, by an RFID for the asset, from the firmware of the asset or from an asset database. IEC 61406 (formerly DIN SPEC 91406) defines the format of such Asset-IDs.

The “Administration Shell Basic Discovery Interface” may be used with an Asset-ID to get the related AAS-IDs (“GetAllAssetAdministrationShellIdsByAssetLink”).

The “Asset Administration Shell Registry Interface” may be used with an AAS-ID to retrieve the related descriptor for an AAS (“GetAssetAdministrationShellDescriptorById”). The retrieved AAS Descriptor includes the endpoint for the “Asset Administration Shell Interface”.

The “Asset Administration Shell Interface” makes the information about the AAS itself and the references to the related submodels available.

The related submodels of an AAS are retrieved by “GetAllSubmodelReferences”. Such a reference includes the SM-ID of a related submodel.

Similarly to the AAS above, the “Submodel Registry Interface” may be used to retrieve the related descriptor for a submodel (“GetSubmodelDescriptorById”) with a specific SM-ID. The retrieved Submodel Descriptor includes the endpoint for the “Submodel Interface”.

The “Submodel Interface” makes the information about the submodel itself and all its included submodel elements available.

Asset Administration Shells and submodels may be deployed on different endpoints in different ways.

One example is the deployment of an AAS on a device. In this case, the AAS might be fixed and might not be changed or deleted. In a cloud scenario, a single AAS may also be deployed as a single container (e.g. docker container).

Another example is the deployment of many Asset Administration Shells in an AAS Repository. In this case, the “Asset Administration Shell Repository Interface” may allow to create and manage multiple AAS in the repository.

The separate interfaces of the HTTP/REST API allow many ways to support different deployments.

For an AAS repository, the combination “Asset Administration Shell Repository Interface”, “Asset Administration Shell Interface”, “Submodel Interface”, “Serialization Interface”, and “Self-Description Interface” is proposed.

This will result in the following HTTP/REST paths as described in a combined OpenAPI file (<https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRepositoryServiceSpecification/V3.0_SSP-001>)[[2]](#footnote-3):

/shells

/shells/{aas-identifier}

/shells/{aas-identifier}/asset-information

/shells/{aas-identifier}/asset-information/thumbnail

/shells/{aas-identifier}/submodel-refs

/shells/{aas-identifier}/submodel-refs/{submodel-identifier}

/shells/{aas-identifier}/submodels/{submodel-identifier}

/shells/{aas-identifier}/submodels/{submodel-identifier}/submodel-elements

/shells/{aas-identifier}/submodels/{submodel-identifier}/submodel-elements/{idShortPath}

/shells/{aas-identifier}/submodels/{submodel-identifier}/submodel-elements/{idShortPath}/attachment

/shells/{aas-identifier}/submodels/{submodel-identifier}/submodel-elements/{idShortPath}/invoke

/shells/{aas-identifier}/submodels/{submodel-identifier}/submodel-elements/{idShortPath}/invoke-async

/shells/{aas-identifier}/submodels/{submodel-identifier}/submodel-elements/{idShortPath}/operation-status/{handleId}

/shells/{aas-identifier}/submodels/{submodel-identifier}/submodel-elements/{idShortPath}/operation-results/{handleId}

/serialization  
/description

If the repository also supports AASX Packages, it shall be extended by additionally supporting a “AASX File Server” Profile[[3]](#footnote-4).

The example of a device or container containing one AAS with its related submodels will result in the following HTTP/REST paths as described in the related OpenAPI file (<https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellServiceSpecification/V3.0_SSP-001>)2:

/aas

/aas/asset-information

/aas/asset-information/thumbnail

/aas/submodel-refs

/aas/submodel-refs/{submodel-identifier}

/aas/submodels/{submodel-identifier}

/aas/submodels/{submodel-identifier}/submodel-elements

/aas/submodels/{submodel-identifier}/submodel-elements/{idShortPath}

/aas/submodels/{submodel-identifier}/submodel-elements/{idShortPath}/attachment

/aas/submodels/{submodel-identifier}/submodel-elements/{idShortPath}/invoke

/aas/submodels/{submodel-identifier}/submodel-elements/{idShortPath}/invoke-async

/aas/submodels/{submodel-identifier}/submodel-elements/{idShortPath}/operation-status/{handleId}

/aas/submodels/{submodel-identifier}/submodel-elements/{idShortPath}/operation-results/{handleId}

/serialization  
/description

Note: identifiers are base64url-encoded in the API, i.e. {aas-identifier} and {submodel-identifier}. The {idShortPath} is URL-encoded in the API.

# Asset Administration Shell Interfaces

## General

These interfaces make it possible to access the elements of Asset Administration Shells or Submodels.

The AASX File Server Interface enables management of AASX packages on a server. A list of available packages can be retrieved. Each package in the list can be downloaded, uploaded, or deleted. New packages can also be added.

AASX packages are stored and managed independently from instantiated Asset Administration Shells or submodels on a server. The server documentation shall contain a description of when and how AASX packages are handled, e.g. if Asset Administration Shells or Submodels in AASX packages are instantiated at startup of the server and/or if they are also instantiated when an AASX package is changed by an API operation.

## Asset Administration Shell Interface and Operations

### Asset Administration Shell Interface

| Interface: Asset Administration Shell | |
| --- | --- |
| **Operation Name** | **Description** |
| GetAssetAdministrationShell | Returns the Asset Administration Shell |
| PutAssetAdministrationShell | Replaces the current Asset Administration Shell |
| GetAllSubmodelReferences | Returns all Submodel References |
| PostSubmodelReference | Creates a Submodel Reference at the Asset Administration Shell |
| DeleteSubmodelReference | Deletes a specific Submodel Reference from the Asset Administration Shell |
| GetAssetInformation | Returns the Asset Information |
| PutAssetInformation | Replaces the Asset Information |
| GetThumbnail | Returns the thumbnail file |
| PutThumbnail | Replaces the thumbnail file |
| DeleteThumbnail | Deletes the thumbnail |

### Operation GetAssetAdministrationShell

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetAssetAdministrationShell | | | |
| **Explanation** | Returns the Asset Administration Shell | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAssetAdministrationShell/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| serializationModifier | Defines the format of the response | no | SerializationModifier | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Asset Administration Shell | yes | AssetAdministrationShell | 1 |

### Operation PutAssetAdministrationShell

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | PutAssetAdministrationShell | | | |
| **Explanation** | Replaces the Asset Administration Shell | | | |
| **semanticId** | https://admin-shell.io/aas/API/PutAssetAdministrationShell/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| aas | AssetAdministrationShell | yes | Asset Administration Shell object | 1 |
| Output Parameter | | | | |
| statusCode | StatusCode | yes | Status code | 1 |
| payload | AssetAdministrationShell | yes | Replaced Asset Administration Shell | 1 |

### Operation GetAllSubmodelReferences

| **Operation Name** | GetAllSubmodelReferences | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns all Submodel References | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllSubmodelReferences/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Submodel References | yes | Reference | 0..\* |

### Operation PostSubmodelReference

| **Operation Name** | PostSubmodelReference | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Creates a Submodel Reference at the Asset Administration Shell | | | |
| **semanticId** | https://admin-shell.io/aas/API/PostSubmodelReference/3/0 | | | |
| **Name** | | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| submodelRef | | Reference to the Submodel | yes | Reference | 1 |
| Output Parameter | | | | |
| statusCode | | Status code | yes | StatusCode | 1 |
| payload | | Created Submodel Reference | yes | Reference | 1 |

### Operation DeleteSubmodelReference

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | DeleteSubmodelReference | | | |
| **Explanation** | Deletes the Submodel Reference from the Asset Administration Shell | | | |
| **semanticId** | https://admin-shell.io/aas/API/DeleteSubmodelReference/3/0 | | | |
| **Name** | | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| submodelId | | The unique id of the Submodel for the reference to be deleted | yes | Identifier | 1 |
| Output Parameter | | | | |  |  |
| statusCode | | Status code | yes | StatusCode | 1 |

### Operation GetAssetInformation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetAssetInformation | | | |
| **Explanation** | Returns the Asset Information | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAssetInformation/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Asset Information | yes | AssetInformation | 1 |

### Operation PutAssetInformation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | PutAssetInformation | | | |
| **Explanation** | Replaces the Asset Information | | | |
| **semanticId** | https://admin-shell.io/aas/API/PutAssetInformation/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| assetInfo | Asset Information object | yes | AssetInformation | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |

### Operation GetThumbnail

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetThumbnail | | | |
| **Explanation** | Returns the thumbnail file | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetThumbnail/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested thumbnail file | yes | File Content | 1 |

### Operation PutThumbnail

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | PutThumbnail | | | |
| **Explanation** | Replaces the thumbnail file | | | |
| **semanticId** | https://admin-shell.io/aas/API/PutThumbnail/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| file | Thumbnail file | yes | File Content | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |

### Operation DeleteThumbnail

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | DeleteThumbnail | | | |
| **Explanation** | Deletes the thumbnail file | | | |
| **semanticId** | https://admin-shell.io/aas/API/DeleteThumbnail/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |

## Submodel Interface and Operations

### Submodel Interface

| Interface: Submodel | |
| --- | --- |
| **Operation Name** | **Description** |
| GetSubmodel | Returns the Submodel |
| GetAllSubmodelElements | Returns all submodel elements including their hierarchy |
| GetSubmodelElementByPath | Returns a specific submodel element from the Submodel at a specified path |
| GetFileByPath | Returns a specific file from the Submodel at a specified path |
| PutFileByPath | Replaces the file of an existing submodel element at a specified path within the submodel element hierarchy |
| DeleteFileByPath | Deletes the file of an existing submodel element at a specified path within the submodel element hierarchy |
| PutSubmodel | Replaces the Submodel |
| PatchSubmodel | Updates the Submodel |
| PostSubmodelElement | Creates a new submodel element as a child of the submodel. The idShort of the the new submodel element must be set in the payload. |
| PostSubmodelElementByPath | Creates a new submodel element at a specified path within the submodel elements hierarchy. The idShort of the the new submodel element must be set in the payload. |
| PutSubmodelElementByPath | Replaces an existing submodel element at a specified path within the submodel element hierarchy |
| PatchSubmodelElementByPath | Updates an existing submodel element at a specified path within the submodel element hierarchy |
| GetSubmodelElementValueByPath | Returns the value of the submodel element at a specified path according to the protocol-specific RAW-value payload |
| DeleteSubmodelElementByPath | Deletes a submodel element at a specified path within submodel element hierarchy |
| InvokeOperationSync | Synchronously invokes an Operation at a specified path with a client timeout in ms |
| InvokeOperationAsync | Asynchronously invokes an Operation at a specified path with a client timeout in ms |
| GetOperationAsyncStatus | Returns the current status of an asynchronously invoked operation |
| GetOperationAsyncResult | Returns the OperationResult of an asynchronously invoked operation |

### Operation GetSubmodel

| **Operation Name** | GetSubmodel | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns the Submodel | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetSubmodel/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | |  | |
| serializationModifier | Defines the format of the response | no | SerializationModifier | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Submodel | yes | Submodel | 1 |

### Operation GetAllSubmodelElements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetAllSubmodelElements | | | |
| **Explanation** | Returns all submodel elements including their hierarchy | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllSubmodelElements/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| serializationModifier | Defines the format of the response | no | SerializationModifier | 1 |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested submodel elements | yes | SubmodelElement | 0..\* |

### Operation GetSubmodelElementByPath

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetSubmodelElementByPath | | | |
| **Explanation** | Returns a specific submodel element from the Submodel at a specified path | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetSubmodelElementByPath/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| path | IdShort-Path via relative Reference/Keys to a submodel element | yes | Key | 1..\* |
| serializationModifier | Defines the format of the response | no | SerializationModifier | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested submodel element | yes | SubmodelElement | 0..1 |

### Operation GetFileByPath

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetFileByPath | | | |
| **Explanation** | Returns a specific file from the Submodel at a specified path | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetFileByPath/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| path | IdShort-Path via relative Reference/Keys to a submodel element | yes | Key | 1..\* |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested file | yes | File Content | 0..1 |

### Operation PutFileByPath

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | PutFileByPath | | | |
| **Explanation** | Replaces the file of an existing submodel element at a specified path within the submodel element hierarchy | | | |
| **semanticId** | https://admin-shell.io/aas/API/PutFileByPath/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| path | IdShort-Path via relative Reference/Keys to a submodel element | yes | Key | 1..\* |
| payload | Replacing file | yes | File Content | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |

### Operation DeleteFileByPath

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | DeleteFileByPath | | | |
| **Explanation** | Deletes the file of an existing submodel element at a specified path within the submodel element hierarchy | | | |
| **semanticId** | https://admin-shell.io/aas/API/DeleteFileByPath/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| path | IdShort-Path via relative Reference/Keys to a submodel element | yes | Key | 1..\* |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |

### Operation PutSubmodel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | PutSubmodel | | | |
| **Explanation** | Replaces the Submodel | | | |
| **semanticId** | https://admin-shell.io/aas/API/PutSubmodel/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| submodel | Submodel object | yes | Submodel | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Replaced submodel | yes | Submodel | 1 |

### Operation PatchSubmodel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | PatchSubmodel | | | |
| **Explanation** | Updates the Submodel | | | |
| **semanticId** | https://admin-shell.io/aas/API/PatchSubmodel/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** | |
| Input Parameter | | | | | |
| serializationModifier | Defines the format of the input  Note: values remain unchanged with content=metadata. | no | SerializationModifier | 1 | |
| submodel | Submodel object | yes | Submodel | 1 | |
| Output Parameter | | | | | |
| statusCode | Status code | yes | StatusCode | 1 | |
| payload | Updated submodel | yes | Submodel | 1 | |

### Operation PostSubmodelElement

| **Operation Name** | PostSubmodelElement | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Creates a new submodel element as a child of the submodel. The idShort of the new submodel element must be set in the payload.  Note: the creation of the idShort is out of scope and must be handled in a proprietary way. | | | |
| **semanticId** | https://admin-shell.io/aas/API/PostSubmodelElement/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| submodelElement | Submodel element object | yes | SubmodelElement | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Created submodel element | yes | SubmodelElement | 1 |

### Operation PostSubmodelElementByPath

| **Operation Name** | PostSubmodelElementByPath | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Creates a new submodel element at a specified path within the submodel element hierarchy. The idShort of the new submodel element must be set in the payload.  Note: the creation of the idShort is out of scope and must be handled in a proprietary way. | | | |
| **semanticId** | https://admin-shell.io/aas/API/PostSubmodelElementByPath/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| path | The IdShortPath to the SubmodelElement under which the new SubmodelElement shall be addedIdShort-Path via relative Reference/Keys to a submodel element | yes | Key | 1..\* |
| submodelElement | Submodel element object | yes | SubmodelElement | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Created submodel element | yes | SubmodelElement | 1 |

Note: if the PostSubmodelElementByPath is executed towards a SubmodelElementList, the new SubmodelElement is added to the end of the list.

### Operation PutSubmodelElementByPath

| **Operation Name** | PutSubmodelElementByPath | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Replaces an existing submodel element at a specified path within the submodel element hierarchy | | | |
| **semanticId** | https://admin-shell.io/aas/API/PutSubmodelElementByPath/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| path | The IdShortPath to the SubmodelElement which shall be replacedIdShort-Path via relative Reference/Keys to a submodel element | yes | Key | 1..\* |
| submodelElement | Submodel element object | yes | SubmodelElement | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Replaced submodel element | yes | SubmodelElement | 1 |

### Operation PatchSubmodelElementByPath

| **Operation Name** | PatchSubmodelElementByPath | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Updates an existing submodel element at a specified path within the submodel element hierarchy | | | |
| **semanticId** | https://admin-shell.io/aas/API/PatchSubmodelElementByPath/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| serializationModifier | Defines the format of the input  Note: values remain unchanged with content=metadata. | no | SerializationModifier | 1 |
| path | IdShort-Path via relative Reference/Keys to a submodel element | yes | Key | 1..\* |
| submodelElement | Submodel element object | yes | SubmodelElement | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Updated submodel element | yes | SubmodelElement | 1 |

### Operation GetSubmodelElementValueByPath

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetSubmodelElementValueByPath | | | |
| **Explanation** | Returns a specific submodel element value from the Submodel at a specified path according to the ValueOnly-serialization as defined in clause 11.4.1 | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetSubmodelElementValueByPath/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| path | IdShort-Path via relative Reference/Keys to a submodel elementhort-Path via relative Reference/Keys to a submodel element | yesIdShort-Path via relative Reference/Keys to a submodel element | Key | 1..\* |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested submodel element value | yes | SubmodelElement | 1 |

### Operation PatchSubmodelElementValueByPath

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | PatchSubmodelElementValueByPath | | | |
| **Explanation** | Sets the value of the submodel element at a specified path according to the ValueOnly-serialization as defined in clause 11.4.1 | | | |
| **semanticId** | https://admin-shell.io/aas/API/PatchSubmodelElementValueByPath/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| path | IdShort-Path via relative Reference/Keys to a submodel elementIdShort-Path via relative Reference/Keys to a submodel element | yes | Key | 1..\* |
| payload | The new value of the submodel element | yes | SubmodelElement | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |

### Operation DeleteSubmodelElementByPath

| **Operation Name** | DeleteSubmodelElementByPath | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Deletes a submodel element at a specified path within the submodel elements hierarchy | | | |
| **semanticId** | https://admin-shell.io/aas/API/DeleteSubmodelElementByPath/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| path | IdShort-Path via relative Reference/Keys to a submodel element | yes | Key | 1..\* |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |

### Operation InvokeOperationSync

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | InvokeOperationSync | | | |
| **Explanation** | Synchronously invokes an Operation at a specified path | | | |
| **semanticId** | https://admin-shell.io/aas/API/InvokeOperationSync/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| path | IdShort-Path via relative Reference/Keys to a submodel element, in this case an operation | yes | Key | 1..\* |
| inputArgument | Input argument | no | OperationVariable | 1..\* |
| inoutputArgument | Inoutput argument | no | OperationVariable | 1..\* |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | The Operation Result | yes | OperationResult | 1 |

### Operation InvokeOperationAsync

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | InvokeOperationAsync | | | |
| **Explanation** | Asynchronously invokes an Operation at a specified path | | | |
| **semanticId** | https://admin-shell.io/aas/API/InvokeOperationAsync/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| path | IdShort-Path via relative Reference/Keys to a submodel element, in this case an operation | yes | Key | 1..\* |
| inputArgument | Input argument | no | OperationVariable | 1..\* |
| inoutputArgument | Inoutput argument | no | OperationVariable | 1..\* |
| clientTimeoutDuration | Timestamp indicating when the client expects the server to have finished execution of the invoked operation | yes | duration | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | The returned handle of an operation’s asynchronous invocation used to request the current state of the operation’s execution | yes | OperationHandle | 1 |

### Operation GetOperationAsyncStatus

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetOperationAsyncStatus | | | |
| **Explanation** | Returns the current status of an asynchronously invoked operation | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetOperationAsnycStatus/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| operationHandle | The returned handle of an operation’s asynchronous invocation used to request the current state of the operation’s execution | yes | OperationHandle | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Execution state of the operation | yes | OperationResult | 1 |

### Operation GetOperationAsyncResult

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetOperationAsyncResult | | | |
| **Explanation** | Returns the OperationResult of an asynchronously invoked operation | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetOperationAsnycResult/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| operationHandle | The returned handle of an operation’s asynchronous invocation used to request the current state of the operation’s execution | yes | OperationHandle | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Operation Result | yes | OperationResult | 1 |

## Serialization Interface and Operations

### Serialization Interface

|  |  |
| --- | --- |
| Interface: Serialization | |
| **Operation Name** | **Description** |
| GenerateSerializationByIds | Returns an appropriate serialization based on the specified format (see SerializationFormat). |

### Operation GenerateSerializationByIds

| **Operation Name** | GenerateSerializationByIds | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns an appropriate serialization based on the specified format (see SerializationFormat). | | | |
| **semanticId** | https://admin-shell.io/aas/API/GenerateSerializationByIds/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| aasIds | The unique ids of the Asset Administration Shells to be contained in the serialization | no | Identifier | 1..\* |
| submodelIds | The unique ids of the Submodels to be contained in the serialization | no | Identifier | 1..\* |
| includeConceptDescriptions | Include concept descriptions | no | boolean | 1 |
| serializationFormat | Denotes in which serialization format the requested content shall be delivered | no | SerializationFormat | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Serialization of the requested Asset Administration Shells and/or Submodels with or without ConceptDescriptions in specified SerializationFormat. | yes | Environment | 1 |

| **Enumeration:** | SerializationFormat |
| --- | --- |
| **Explanation:** | Determines the format of serialization, i.e. JSON, XML, RDF, AML, etc.RFC 6838, IANA Media Types, and defined custom content types; additional elements may be added in future versions |
| **Set of:** | -- |
| **Literal** | **Explanation** |
| application/json | JSON serialization of the requested data object inside an AAS Environment structure |
| application/xml | XML serialization of the requested data object inside an AAS Environment structure (default) |
| application/asset-administration-shell-package+xml | AASX-Package (binary data) containing the requested data object |

## AASX File Server Interface and Operations

### AASX File Server Interface

|  |  |
| --- | --- |
| Interface: AASX File Server | |
| **Operation Name** | **Description** |
| GetAllAASXPackageIds | Returns a list of available AASX packages at the server |
| GetAASXByPackageId | Returns a specific AASX package from the server |
| PostAASXPackage | Creates an AASX package at the server |
| PutAASXByPackageId | Replaces the AASX package at the server |
| DeleteAASXByPackageId | Deletes a specific AASX package |

### Operation GetAllAASXPackageIds

| **Operation Name** | GetAllAASXPackageIds | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns a list of available AASX packages at the server | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllAASXPackageIds/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| aasId | Identifier of the AAS which must exist in each matching AASX package | no | Identifier | 1 |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Matching package list; the PackageDescription includes all Asset Administration Shell identifiers, also those which may have not been requested through the aasId input parameter | yes | PackageDescription | 0..\* |

### Operation GetAASXByPackageId

| **Operation Name** | GetAASXByPackageId | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns a specific AASX package from the server | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAASXByPackageId/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| packageId | Requested package ID from the package list | yes | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| filename | Filename of the AASX package | yes | string | 1 |
| payload | Requested AASX package | yes | AASX Package | 1 |

### Operation PostAASXPackage

| **Operation Name** | PostAASXPackage | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Creates an AASX package at the server | | | |
| **semanticId** | https://admin-shell.io/aas/API/PostAASXPackage/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| aasIds | Included AAS Ids  Note: it is not mandatory for servers to read and parse AASX packages. Servers may simply store the AASX files with their related given aasIds. | no | Identifier | 0..\* |
| file | New AASX package | yes | AASX package | 1 |
| filename | Filename of the AASX package | yes | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| packageId | New Package ID | yes | string | 1 |

### Operation PutAASXPackageById

| **Operation Name** | PutAASXPackageById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Replaces the AASX package at the server | | | |
| **semanticId** | https://admin-shell.io/aas/API/PutAASXPackageById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| packageId | Package ID from the package list | yes | string | 1 |
| aasIds | Included AAS Ids  Note: it is not mandatory for servers to read and parse AASX packages. Servers may simply store the AASX files with their related given aasIds. | no | Identifier | 0..\* |
| file | New AASX package | yes | AASX package | 1 |
| filename | Filename of the AASX package | yes | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |

### Operation DeleteAASXPackageById

| **Operation Name** | DeleteAASXPackageById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Deletes a specific AASX package from the server | | | |
| **semanticId** | https://admin-shell.io/aas/API/DeleteAASXPackageById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| packageId | Package ID from the package list | yes | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |

# Registration Interfaces

## General

These interfaces allow to register and unregister descriptors of administration shells or submodels. The descriptors contain the information needed to access the interfaces (as described in Clause 5) of the corresponding element. This required information includes the endpoint in the dedicated environment.

Lookup interfaces provide access to the registered descriptors by identifiers (Asset Administration Shell and Submodel ID). These identifiers may be discovered through the interfaces described in Clause 8.

## Asset Administration Shell Registry Interface and Operations

### Asset Administration Shell Registry Interface

|  |  |
| --- | --- |
| Interface: Asset Administration Shell Registry | |
| **Operation Name** | **Description** |
| GetAllAssetAdministrationShellDescriptors | Returns all Asset Administration Shell Descriptors |
| GetAssetAdministrationShellDescriptorById | Returns a specific Asset Administration Shell Descriptor |
| PostAssetAdministrationShellDescriptor | Creates a new Asset Administration Shell Descriptor, i.e. registers an AAS |
| PutAssetAdministrationShellDescriptorById | Replaces an existing Asset Administration Shell Descriptor, i.e. replaces registration information |
| DeleteAssetAdministrationShellDescriptorById | Deletes an Asset Administration Shell Descriptor, i.e. de-registers an AAS |

### Operation GetAllAssetAdministrationShellDescriptors

| **Operation Name** | GetAllAssetAdministrationShellDescriptors | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns all Asset Administration Shell Descriptors | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllAssetAdministrationShellDescriptors/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| assetKind | The kind of the assets to retrieve (Type, Instance) | Yes | AssetKind | 1 |
| assetType | The type of the assets to retrieve, encoded as unique id | Yes | Identifier | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | List of Asset Administration Shell Descriptors | no | AssetAdministrationShellDescriptor | 1..\* |

### Operation GetAssetAdministrationShellDescriptorById

| **Operation Name** | GetAssetAdministrationShellDescriptorById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns a specific Asset Administration Shell Descriptor | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAssetAdministrationShellDescriptorById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| aasIdentifier | The Asset Administration Shell’s unique id | yes | Identifier | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Asset Administration Shell Descriptor | yes | AssetAdministrationShellDescriptor | 1 |

### Operation PostAssetAdministrationShellDescriptor

| **Operation Name** | PostAssetAdministrationShellDescriptor | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Creates a new Asset Administration Shell Descriptor, i.e. registers an AAS | | | |
| **semanticId** | https://admin-shell.io/aas/API/PostAssetAdministrationShellDescriptor/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| shellDescriptor | Object containing the Asset Administration Shell’s identification and endpoint informationntaining the Asset Administration Shell’s identification and endpoint information | yes | AssetAdministrationShellDescriptor | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Created Asset Administration Shell Descriptor | yes | AssetAdministrationShellDescriptor | 1 |

### Operation PutAssetAdministrationShellDescriptorById

| **Operation Name** | PutAssetAdministrationShellDescriptorById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Replaces an existing Asset Administration Shell Descriptor, i.e. replaces registration information | | | |
| **semanticId** | https://admin-shell.io/aas/API/PutAssetAdministrationShellDescriptorById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| shellDescriptor | Object containing the Asset Administration Shell’s identification and endpoint informationcontaining the Asset Administration Shell’s identification and endpoint information | yes | AssetAdministrationShellDescriptor | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Replaced Asset Administration Shell Descriptor | yes | AssetAdministrationShellDescriptor | 1 |

### Operation DeleteAssetAdministrationShellDescriptorById

| **Operation Name** | DeleteAssetAdministrationShellDescriptorById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Deletes an Asset Administration Shell Descriptor, i.e. de-registers an AAS | | | |
| **semanticId** | https://admin-shell.io/aas/API/DeleteAssetAdministrationShellDescriptorById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| aasIdentifier | The Asset Administration Shell’s unique id | yes | Identifer | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |

## Submodel Registry Interface and Operations

### Submodel Registry Interface

| Interface:Submodel Registry | |
| --- | --- |
| **Operation Name** | **Description** |
| GetAllSubmodelDescriptors | Returns all submodel descriptors |
| GetSubmodelDescriptorById | Returns a specific submodel descriptor |
| PostSubmodelDescriptor | Creates a new submodel descriptor, i.e. registers a submodel |
| PutSubmodelDescriptorById | Replaces an existing submodel descriptor, i.e. replaces registration information |
| DeleteSubmodelDescriptorById | Deletes a submodel descriptor, i.e. de-registers a submodel |

### Operation GetAllSubmodelDescriptors

| **Operation Name** | GetAllSubmodelDescriptors | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns all submodel descriptors | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllSubmodelDescriptors/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | | | |
| statusCode | The maximum size of the result set | yes | StatusCode | 1 |
| payload | The position from which to resume a result listing | no | SubmodelDescriptor | 1..\* |

### Operation GetSubmodelDescriptorById

| **Operation Name** | GetSubmodelDescriptorById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns a specific Submodel Descriptor | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetSubmodelDescriptorById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| submodelIdentifier | The Submodel’s unique id | yes | Identifier | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested submodel descriptor | yes | SubmodelDescriptor | 1 |

### Operation PostSubmodelDescriptor

| **Operation Name** | PostSubmodelDescriptor | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Creates a new submodel descriptor, i.e. registers a submodel | | | |
| **semanticId** | https://admin-shell.io/aas/API/PostSubmodelDescriptor/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| submodel Descriptor | Object containing the Submodel’s identification and endpoint information | yes | SubmodelDescriptor | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Created submodel descriptor | yes | SubmodelDescriptor | 1 |

### Operation PutSubmodelDescriptorById

| **Operation Name** | PutSubmodelDescriptorById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Replaces an existing submodel descriptor, i.e. replaces registration information | | | |
| **semanticId** | https://admin-shell.io/aas/API/PutSubmodelDescriptorById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| submodel Descriptor | Object containing the Submodel’s identification and endpoint information | yes | SubmodelDescriptor | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Replaced submodel descriptor | yes | SubmodelDescriptor | 1 |

### Operation DeleteSubmodelDescriptorById

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | DeleteSubmodelDescriptorById | | | |
| **Explanation** | Deletes a Submodel Descriptor, i.e. de-registers a submodel | | | |
| **semanticId** | https://admin-shell.io/aas/API/DeleteSubmodelDescriptorById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| submodelIdentifier | The Submodel’s unique id | yes | Identifier | 1 |
| Output Parameter | | |  |  |
| statusCode | Status code | yes | StatusCode | 1 |

# Repository Interfaces

## General

These interfaces allow to manage Asset Administration Shells, Submodels, and Concept Descriptions. They further provide access to the data of these elements through interfaces described in Clause 5. A repository can host multiple entities. These entities can be stored in individual repositories of a decentral system. The endpoints of the entities managed by one repository shall be resolved by subsequent calls to discover (Clause 8) and lookup (Clause 6) interfaces to such decentralized systems.

Sometimes, these kinds of services are also classified as Asset Administration Shell management services.

The interfaces that provide access to the entities (Asset Administration Shells, Submodels, Concept Descriptions) themselves are convenience interfaces that provide access in a system where the services are managed by central repositories.

## Asset Administration Shell Repository Interface and Operations

### Asset Administration Shell Repository Interface

|  |  |
| --- | --- |
| Interface: Asset Administration Shell Registry | |
| **Operation Name** | **Description** |
| GetAllAssetAdministrationShells | Returns all Asset Administration Shells |
| GetAssetAdministrationShellById | Returns a specific Asset Administration Shell |
| GetAllAssetAdministrationShellsByAssetId | Returns all Asset Administration Shells that are linked to a globally unique asset identifier or to specific asset ids |
| GetAllAssetAdministrationShellsByIdShort | Returns all Asset Administration Shells with a specific idShort |
| PostAssetAdministrationShell | Creates a new Asset Administration Shell. The id of the new Asset Administration Shell must be set in the payload.  Note: the creation of the idShort is out of scope and must be handled in a proprietary way. |
| PutAssetAdministrationShellById | Replaces an existing Asset Administration Shell |
| DeleteAssetAdministrationShellById | Deletes an Asset Administration Shell |

### Operation GetAllAssetAdministrationShells

| **Operation Name** | GetAllAssetAdministrationShells | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns all Asset Administration Shells | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllAssetAdministrationShells/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | |  |  |
| serializationModifier | Defines the format of the response | yes | SerializationModifier | 1 |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | |  |  |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | List of Asset Administration Shells | no | AssetAdministrationShell | 1 |

### Operation GetAssetAdministrationShellById

| **Operation Name** | GetAssetAdministrationShellById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns a specific Asset Administration Shell | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAssetAdministrationShellById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| id | The Asset Administration Shell’s unique id | yes | Identifier | 1 |
| serializationModifier | Defines the format of the response | yes | SerializationModifier | 1 |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Asset Administration Shell | yes | AssetAdministrationShell | 1 |

### Operation GetAllAssetAdministrationShellsByAssetId

| **Operation Name** | GetAllAssetAdministrationShellsByAssetId | | | | |
| --- | --- | --- | --- | --- | --- |
| **Explanation** | Returns all Asset Administration Shells that are linked to a globally unique asset identifier or to specific asset ids | | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllAssetAdministrationShellsByAssetId/3/0 | | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** | |
| Input Parameter | | | | | |
| key | The key of the AssetId The name of the specific asset identifier or the predefined name “*globalAssetId*” that would refer to the *AssetInformation/globalAssetId* | yes | string | 1 | |
| keyIdentifier | The key identifier object | yes | string | 1 | |
| serializationModifier | Defines the format of the response | yes | SerializationModifier | 1 | |
| limit | The maximum size of the result set | yes | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | yes | string | 1 |
| Output Parameter | | | | | |
| statusCode | Status code | yes | StatusCode | 1 | |
| payload | Requested Asset Administration Shells | no | AssetAdministrationShell | 1..\* | |

### Operation GetAllAssetAdministrationShellsByIdShort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetAllAssetAdministrationShellsByIdShort | | | |
| **Explanation** | Returns all Asset Administration Shells with a specific *idShort* | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllAssetAdministrationShellsByIdShort/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | |  |  |
| idShort | The Asset Administration Shell’s idShort | yes | NameType |  |
| serializationModifier | Defines the format of the response | yes | SerializationModifier |  |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | |  |  |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Asset Administration Shells | no | AssetAdministrationShell | 1 |

### Operation PostAssetAdministrationShell

| **Operation Name** | PostAssetAdministrationShell | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Creates a new Asset Administration Shell. The id of the new Asset Administration Shell must be set in the payload.  Note: the creation of the idShort is out of scope and must be handled in a proprietary way. | | | |
| **semanticId** | https://admin-shell.io/aas/API/PostAssetAdministrationShell/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| aas | Asset Administration Shell object | yes | AssetAdministrationShell | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Created Asset Administration Shell | yes | AssetAdministrationShell | 1 |

### Operation PutAssetAdministrationShellById

| **Operation Name** | PutAssetAdministrationShellById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Replaces an existing Asset Administration Shell | | | |
| **semanticId** | https://admin-shell.io/aas/API/PutAssetAdministrationShellById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| aas | Asset Administration Shell object | yes | AssetAdministrationShell | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Replaced Asset Administration Shell | yes | AssetAdministrationShell | 1 |

### Operation DeleteAssetAdministrationShellById

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | DeleteAssetAdministrationShellById | | | |
| **Explanation** | Deletes an Asset Administration Shell | | | |
| **semanticId** | https://admin-shell.io/aas/API/DeleteAssetAdministrationShellById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| id | The Asset Administration Shell’s unique id | yes | Identifier | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |

## Submodel Repository Interface and Operations

### Submodel Repository Interface

| Interface: Submodel Repository | |
| --- | --- |
| **Operation Name** | **Description** |
| GetAllSubmodels | Returns all Submodels |
| GetSubmodelById | Returns a specific Submodel |
| GetAllSubmodelsBySemanticId | Returns all Submodels with a specific SemanticId |
| GetAllSubmodelsBySupplementalSemanticId | Returns all Submodels with a specific SupplementalSemanticId |
| GetAllSubmodelsByIdShort | Returns all Submodels with a specific *idShort* |
| PostSubmodel | Creates a new Submodel. The id of the new submodel must be set in the payload.  Note: the creation of the idShort is out of scope and must be handled in a proprietary way. |
| PutSubmodelById | Replaces an existing Submodel |
| PatchSubmodelById | Updates an existing submodel |
| DeleteSubmodelById | Deletes a Submodel |

### Operation GetAllSubmodels

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetAllSubmodels | | | |
| **Explanation** | Returns all Submodels | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllSubmodels/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| serializationModifier | Defines the format of the response | yes | SerializationModifier | 1 |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | List of Submodels | no | Submodel | 1..\* |

### Operation GetSubmodelById

| **Operation Name** | GetSubmodelById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns a specific Submodel | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetSubmodelById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| id | The Submodel’s unique id | yes | Identifier | 1 |
| serializationModifier | Defines the format of the response | yes | SerializationModifier | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Submodel | yes | Submodel | 1 |

### Operation GetAllSubmodelsBySemanticId

| **Operation Name** | GetAllSubmodelsBySemanticId | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns all Submodels with a specific SemanticId or SupplementalSemanticId. If either the semanticId fits to the input parameter or at least one of the SupplementalSemanticIds, the submodel is returned. | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllSubmodelsBySemanticId/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| semanticId | Identifier of the semantic definition | yes | Reference | 1 |
| serializationModifier | Defines the format of the response | yes | SerializationModifier | 1 |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Submodels | no | Submodel | 1..\* |

### Operation GetAllSubmodelsByIdShort

| **Operation Name** | GetAllSubmodelsByIdShort | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns all Submodels with a specific *idShort* | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllSubmodelsByIdShort/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| idShort | The Submodel’s idShort | yes | NameType | 1 |
| serializationModifier | Defines the format of the response | yes | SerializationModifier | 1 |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Submodels | no | Submodel | 1..\* |

### Operation PostSubmodel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | PostSubmodel | | | |
| **Explanation** | Creates a new Submodel. The id of the new submodel must be set in the payload.  Note: the creation of the idShort is out of scope and must be handled in a proprietary way. | | | |
| **semanticId** | https://admin-shell.io/aas/API/PostSubmodel/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| submodel | Submodel object | yes | Submodel | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Created Submodel | yes | Submodel | 1 |

### Operation PutSubmodelById

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | PutSubmodelById | | | |
| **Explanation** | Replaces an existing Submodel | | | |
| **semanticId** | https://admin-shell.io/aas/API/PutSubmodelById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| submodel | Submodel object | yes | Submodel | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Replaced Submodel | yes | Submodel | 1 |

### Operation PatchSubmodelById

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | PatchSubmodelById | | | |
| **Explanation** | Updates an existing Submodel | | | |
| **semanticId** | https://admin-shell.io/aas/API/PatchSubmodelById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| serializationModifier | Defines the format of the input  Note: values remain unchanged with content=metadata. | yes | SerializationModifier | 1 |
| submodel | Submodel object | yes | Submodel | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Updated submodel | yes | Submodel | 1 |

### Operation DeleteSubmodelById

| **Operation Name** | DeleteSubmodelById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Deletes a Submodel | | | |
| **semanticId** | https://admin-shell.io/aas/API/DeleteSubmodelById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| id | The Submodel’s unique id | yes | Identifier | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |

## Concept Description Repository Interface and Operations

### Concept Description Repository Interface

|  |  |
| --- | --- |
| Interface: Concept Description Repository | |
| **Operation Name** | **Description** |
| GetAllConceptDescriptions | Returns all Concept Descriptions |
| GetConceptDescriptionById | Returns a specific Concept Description |
| GetAllConceptDescriptionsByIdShort | Returns all Concept Descriptions with a specific *idShort* |
| GetAllConceptDescriptionsByIsCaseOf | Returns all Concept Descriptions with a specific *IsCaseOf*-reference |
| GetAllConceptDescriptionsByDataSpecificationReference | Returns all Concept Descriptions with a specific *dataSpecification* reference |
| PostConceptDescription | Creates a new Concept Description. The id of the the new Concept Description must be set in the payload.  Note: the creation of the idShort is out of scope and must be handled in a proprietary way. |
| PutConceptDescriptionById | Replaces an existing Concept Description |
| DeleteConceptDescriptionById | Deletes a Concept Description |

### Operation GetAllConceptDescriptions

| **Operation Name** | GetAllConceptDescriptions | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns all Concept Descriptions | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllConceptDescriptions/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| limit | The maximum size of the result set | No | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | String | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | List of Concept Descriptions | no | ConceptDescription | 1..\* |

### Operation GetConceptDescriptionById

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetConceptDescriptionById | | | |
| **Explanation** | Returns a specific Concept Description | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetConceptDescriptionById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| id | The Concept Description’s unique id | yes | Identifier | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Concept Description | yes | ConceptDescription | 1 |

### Operation GetAllConceptDescriptionsByIdShort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetAllConceptDescriptionsByIdShort | | | |
| **Explanation** | Returns all Concept Descriptions with a specific *idShort* | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllConceptDescriptionsByIdShort/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| idShort | The Concept Description’s idShort | yes | NameType | 1 |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Concept Descriptions | no | ConceptDescription | 1..\* |

### Operation GetAllConceptDescriptionsByIsCaseOf

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetAllConceptDescriptionsByIsCaseOf | | | |
| **Explanation** | Returns all Concept Descriptions with a specific *IsCaseOf* reference | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllConceptDescriptionsByIsCaseOf/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| isCaseOf | IsCaseOf reference | yes | Reference | 1 |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Concept Descriptions | no | ConceptDescription | 1..\* |

### Operation GetAllConceptDescriptionsByDataSpecificationReference

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | GetAllConceptDescriptionsByDataSpecificationReference | | | |
| **Explanation** | Returns all Concept Descriptions with a specific *dataSpecification* reference | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllConceptDescriptionsByDataSpecificationReference/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| dataSpecification-Reference | *DataSpecification* reference | yes | Reference | *1* |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested Concept Descriptions | no | ConceptDescription | 1..\* |

### Operation PostConceptDescription

| **Operation Name** | PostConceptDescription | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Creates a new Concept Description. The id of the new Concept Description must be set in the payload.  Note: the creation of the idShort is out of scope and must be handled in a proprietary way. | | | |
| **semanticId** | https://admin-shell.io/aas/API/PostConceptDescription/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| conceptDescription | Concept Description object | yes | ConceptDescription | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Created Concept Description | yes | ConceptDescription | 1 |

### Operation PutConceptDescriptionById

| **Operation Name** | PutConceptDescriptionById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Replaces an existing Concept Description | | | |
| **semanticId** | https://admin-shell.io/aas/API/PutConceptDescriptionById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| conceptDescription | Concept Description object | yes | ConceptDescription | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Replaced Concept Description | yes | ConceptDescription | 1 |

### Operation DeleteConceptDescriptionById

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation Name** | DeleteConceptDescriptionById | | | |
| **Explanation** | Deletes a Concept Description | | | |
| **semanticId** | https://admin-shell.io/aas/API/DeleteConceptDescriptionById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| cdIdentifier | The Concept Description’s unique id | yes | Identifier | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |

# Publish and Discovery Interfaces

## General

These interfaces allow to publish information about Asset Administration Shells that enable a search for asset IDs of the corresponding Asset Administration Shells in a subsequent discovery interface call.

## Asset Administration Shell Basic Discovery Interface and Operations

### Asset Administration Shell Basic Discovery Interface

|  |  |
| --- | --- |
| Interface: Asset Administration Shell Basic Discovery | |
| **Operation Name** | **Description** |
| GetAllAssetAdministrationShellIdsByAssetLink | Returns a list of Asset Administration Shell ids based on asset identifier key-value-pairs |
| GetAllAssetLinksById | Returns a list of asset identifier key-value-pairs based on a given Asset Administration Shell id |
| PostAllAssetLinksById | Creates or replaces all asset identifier key-value-pairs linked to an Asset Administration Shell to edit discoverable content |
| DeleteAllAssetLinksById | Deletes all asset identifier key-value-pair linked to an Asset Administration Shell |

### Operation GetAllAssetAdministrationShellIdsByAssetLink

| **Operation Name** | GetAllAssetAdministrationShellIdsByAssetLink | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns a list of Asset Administration Shell ids based on asset identifier key-value-pairs | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetAllAssetAdministrationShellIdsByAssetLink/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| assetIds | The specific assetId of an asset identifier, which could be the globalAssetId or specificAssetIds.  Note: The key of the asset identifier key-value-pair for the globalAssetId is defined in Clause 4.5. It is the predefined key “*globalAssetId*” that would refer to the *AssetInformation/globalAssetId*. | yes | SpecificAssetId | 1..\* |
| limit | The maximum size of the result set | no | nonNegativeInteger | 1 |
| cursor | The position from which to resume a result listing | no | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Identifiers of all Asset Administration Shells which contain all asset identifier key-value-pairs in their asset information, i.e. AND-match of key-value-pairs per Asset Administration Shell | yes | Identifier | 1..\* |

### Operation GetAllAssetLinksById

| **Operation Name** | GetAllAssetLinksById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns a list of asset identifier key-value-pairs based on an Asset Administration Shell id to edit discoverable content | | | |
| **semanticId** | [https://admin-shell.io/aas/API/GetAllAssetLinksById/3/0](https://admin-shell.io/aas/API/AssetAdministrationShellBasicDiscoveryInterfae/GetAllAssetLinksById/1/0/RC02) | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| aasIdentifier | The Asset Administration Shell’s unique id | yes | string | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Requested asset identifier, which could be the globalAssetId or specificAssetIds.  Note: the name of the SpecificAssetId for the globalAssetId is defined in Clause 4.5. It is the predefined name “*globalAssetId*” that would refer to the *AssetInformation/globalAssetId*. | no | SpecificAssetId | 1..\* |

### Operation PostAllAssetLinksById

| **Operation Name** | PostAllAssetLinksById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Creates new asset identifier key-value-pairs linked to an Asset Administration Shell for discoverable content. The existing content might have to be deleted first. | | | |
| **semanticId** | https://admin-shell.io/aas/API/PostAllAssetLinksById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| aasIdentifier | The Asset Administration Shell’s unique id | yes | string | 1 |
| assetLinks | Asset identifier, which could be the globalAssetId or specificAssetIds.  Note: the name for the globalAssetId is defined in Clause 4.5. It is the predefined key “*globalAssetId*” that would refer to the *AssetInformation/globalAssetId*. | yes | SpecificAssetId | 1 |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| payload | Asset identifier created successfully | yes | SpecificAssetId | 1 |

### Operation DeleteAllAssetLinksById

| **Operation Name** | DeleteAllAssetLinksById | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Deletes all asset identifier key-value-pairs linked to an Asset Administration Shell to edit discoverable content | | | |
| **semanticId** | https://admin-shell.io/aas/API/DeleteAllAssetLinksById/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | |  |  |
| aasIdentifier | The Asset Administration Shell’s unique id | yes | string | 1 |
| Output Parameter | | |  |  |
| statusCode | Status code | yes | StatusCode | 1 |

# Self Description Interface

## Self-Description Interface

|  |  |
| --- | --- |
| Interface: Self-Description | |
| **Operation Name** | **Description** |
| GetSelfDescription | Returns a description object containing the capabilities and supported features of the server. |

## Operation GetSelfDescription

| **Operation Name** | GetSelfDescription | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Returns a description object containing the capabilities and supported features of the server. | | | |
| **semanticId** | https://admin-shell.io/aas/API/GetSelfDescription/3/0 | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Output Parameter | | | | |
| statusCode | Status code | yes | StatusCode | 1 |
| description | Key-value-pairs that describe the capabilities of the providing server | yes | ServiceDescription | 1 |

Note 1: a server implementing more than one service specification profile, e.g. hosting a repository and a registry at the same time, adds both ServiceSpecificationProfileEnum items in the profiles list.

Note 2: a profile value must only be used if the related API is implemented at the path where the API Operation “GetDescription” is published, or child paths.

# Data Types for Payload

## General

For metamodel elements like AssetAdministrationShell, Submodel, Identifier, etc. that are specified in Part 1 [1], please refer to the specification in Bibliography. The AAS package format and the AAS Package type are defined in Part 5 [3]. This clause only defines additional classes that are needed for communication with the API.

## Metamodel Specification Details

The following type definitions are used to describe specific metamodel elements like Asset Administration Shells and submodels regarding their network and deployment configuration. They use certain attributes copied from the model element itself to describe it – hence the name *Descriptor*.

### Descriptor

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | Descriptor | | |
| **Explanation** | The self-describing information of a network resource. This class is not part of the metamodel. | | |
| **Inherits from** | -- | | |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/Descriptor/3/0 | | |
| **Attribute** | **Explanation** | **Type** | **Card.** |
| description | Description or comments on the element  The description can be provided in several languages | MultiLanguageTextType | 0..1 |
| displayName | Display name; can be provided in several languages | MultiLanguageNameType | 0..1 |
| extension | An extension of the element | Extension | 0..\* |

### AssetAdministrationShellDescriptor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class Name** | | AssetAdministrationShellDescriptor | | |
| **Explanation** | | Descriptor of an Asset Administration Shell | | |
| **Inherits from** | | Descriptor | | |
| **semanticId** | | https://admin-shell.io/aas/API/DataTypes/AssetAdministrationShellDescriptor/3/0 | | |
| **Attribute** | **Explanation** | | **Type** | **Card.** |
| administration | Administrative information of the Asset Administration Shell | | AdministrativeInformation | 0..1 |
| assetKind | Denotes whether the asset of the described Asset Administration Shell is of kind “Type” or “Instance” | | AssetKind | 0..1 |
| assetType | The type of the asset described by the Asset Administration Shell of this Descriptor. See AssetInformation/assetType for further information. | | Identifier | 0..1 |
| endpoint | Endpoint of the network resource | | Endpoint | 0..\* |
| globalAssetId | Global reference to the asset the AAS is representing | | Identifier | 0..1 |
| idShort | Short name of the Asset Administration Shell | | NameType | 0..1 |
| id | Globally unique identification of the Asset Administration Shell | | Identifier | 1 |
| specificAssetId | Specific asset identifier | | SpecificAssetId | 0..\* |
| submodelDescriptor | Descriptor of a submodel of the Asset Administration Shell | | SubmodelDescriptor | 0..\* |

Note: the cardinality restriction for AssetAdministrationShellDescriptor/endpoint (optional: 0..\*) allows a provider to skip the declaration of the location of an AssetAdministrationShell and directly point to the endpoints of the contained Submodels through the path AssetAdministrationShellDescriptor/submodelDescriptor-> SubmodelDescriptor/endpoint. A client, therefore, might decide to skip the lookup on the AssetAdministrationShell. Nevertheless, in case the information contained in the AssetAdministrationShellDescriptor deviates from the related AssetAdministrationShell, or attributes are missing, the AssetAdministrationShell is always the source of truth.

### SubmodelDescriptor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class Name** | | SubmodelDescriptor | | |
| **Explanation** | | A descriptor of a submodel | | |
| **Inherits from** | | Descriptor | | |
| **semanticId** | | https://admin-shell.io/aas/API/DataTypes/SubmodelDescriptor/3/0 | | |
| **Attribute** | **Explanation** | | **Type** | **Card.** |
| administration | Administrative information of the Submodel | | AdministrativeInformation | 0..1 |
| endpoint | Endpoint of the network resource | | Endpoint | 1..\* |
| idShort | Short name of the Submodel | | NameType | 0..1 |
| id | Globally unique identification of the Submodel | | Identifier | 1 |
| semanticId | Identifier of the semantic definition of the Submodel | | Reference | 0..1 |
| supplementalSemanticId | Identifier of a supplemental semantic definition of the element called supplemental semantic ID of the element | | Reference | 0..\* |

### Endpoint

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | Endpoint | | |
| **Explanation** | The endpoint description of a network resource. This class is not part of the metamodel. | | |
| **Inherits from** | - | | |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/Endpoint/3/0 | | |
| **Attribute** | **Explanation** | **Type** | **Card.** |
| protocolInformation | Protocol information of the network resource endpoint | ProtocolInformation | 1 |
| interface | Name of the offered interface at the endpoint | IdShortType | 1 |

The following names will be used for the interfaces:

|  |  |
| --- | --- |
| **Interface** | **interface-shortName** |
| Asset Administration Shell Interface | AAS |
| Submodel Interface | SUBMODEL |
| Serialization Interface | SERIALIZE |
| AASX File Server Interface | AASX-FILE |
| Asset Administration Shell Registry Interface | AAS-REGISTRY |
| Submodel Registry Interface | SUBMODEL-REGISTRY |
| Asset Administration Shell Repository Interface | AAS-REPOSITORY |
| Submodel Repository Interface | SUBMODEL-REPOSITORY |
| Concept Description Repository Interface | CD-REPOSITORY |
| Asset Administration Shell Basic Discovery Interface | AAS-DISCOVERY |

The value for the interface attribute is “{interface-shortName}-{interface-version}”.

The interface-version of this specification is “3.0”, e.g. the entry for the Asset Administration Shell Interface is “AAS-3.0”.

See the following example for a descriptor with several endpoints:



### ProtocolInformation

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | ProtocolInformation | | |
| **Explanation** | The protocol information of a network resource endpoint will be defined in DIN SPEC 16593-2. After the release of DIN SPEC 16593-2, any required updates will be made. This class is not part of the metamodel.  The information in this table is a 1:1 copy from DIN SPEC 16593-2. Required changes need to be made by the related DIN working group. | | |
| **Inherits from** | -- | | |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/ProtocolInformation/3/0 | | |
| **Attribute** | **Explanation** | **Type** | **Card.** |
| href | The endpoint address as an URL | String 2048 | 1 |
| endpointProtocol | Either scheme of endpointAdress or scheme + further information. Scheme denotes the highest level of doubtless transmission. | IdShortType | 0..1 |
| endpointProtocolVersion | Array of strings, each entry represents one supported version at this very endpoint, the entry shall be formatted according to the regulations of the protocol specified in the href | IdShortType | 0..\* |
| subprotocol | Allows for referencing sub-protocols that may be used in the context of that endpoint e.g. “OPC Basic SOAP” or UA Binary | IdShortType | 0..1 |
| subprotocolBody | If the sub-protocol field is present, a subprotocolBody might be given to hold extra information, e.g. node and namespace in an OPC UA server | IdShortType | 0..1 |
| subprotocolBodyEncoding | If subprotocolBody is present, the encoding might be explicitly defined, otherwise it shall default to subprotocols encoding scheme | IdShortType | 0..1 |
| securityAttributes | Array of securityAttribute objects, each attribute has 3 properties:  { **type** = Enum security type or standard:   * ‘NONE’, * 'RFC\_TLSA' - TLSA according to rfc6698 * 'W3C\_DID' - W3C DID document ,   **key** = security attribute key according to standard definitions of the security type,  **value** = security attribute value e.g. DANE TLSA Ressource Record }  The securityAttribute objects are treated as possible alternatives (logical “or”) | SecurityAttributeObject | 1..\* |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | SecurityAttributeObject | | |
| **Explanation** | Security attributes as defined by DIN SPEC 16593-2. After the release of DIN SPEC 16593-2, any required updates will be made. This class is not part of the metamodel.  The information in this table is derived from DIN SPEC 16593-2. Required changes need to be made by the related DIN working group. | | |
| **Inherits from** | -- | | |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/SecurityAttributeObject/3/0 | | |
| **Attribute** | **Explanation** | **Type** | **Card.** |
| type | Enum security type or standard | SecurityTypeEnum | 1 |
| key | Security attribute key according to standard definitions of the security type | string | 1 |
| value | Security attribute value e.g. DANE TLSA Ressource Record | string | 1 |

|  |  |
| --- | --- |
| **Enumeration** | SecurityTypeEnum |
| **Explanation** | The security types as defined by DIN SPEC 16593-2. After the release of DIN SPEC 16593-2, any required updates will be made. This class is not part of the metamodel.  The information in this table is derived from DIN SPEC 16593-2. Required changes need to be made by the related DIN working group. |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/ SecurityTypeEnum/3/0 |
| **Literal** | **Explanation** |
| NONE | No predefined security type available |
| RFC\_TLSA | TLSA according to RFC 6698 |
| W3C\_DID | Decentralized Identifiers according to the W3C Recommendation [7] |

### ServiceDescription

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | ServiceDescription | | |
| **Explanation** | The self-describing information of an API Implementation. It enables servers to present their capabilities to the clients, in particular which profiles they implement. At least one defined profile is required. Additional, proprietary attributes might be included. Nevertheless, the server must not expect that a regular client understands them.  This class is not part of the metamodel. | | |
| **Inherits from** | -- | | |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/ServiceDescription/3/0 | | |
| **Attribute** | **Explanation** | **Type** | **Card.** |
| profiles | List of implemented server specification profiles. | ServiceSpecificationProfileEnum | 1..\* |

|  |  |
| --- | --- |
| **Enumeration** | ServiceSpecificationProfileEnum |
| **Explanation** | The identifiers of the standardized service specification profiles. See also clause 12.12 for further details. |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/ ServiceSpecificationProfileEnum/3/0 |
| **Literal** | **Explanation** |
| https://admin-shell.io/aas/API/3/0/ AssetAdministrationShellServiceSpecification/SSP-001 | Indicates that the server implemented all features of the Asset Administration Shell Service Specification Full Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ AssetAdministrationShellServiceSpecification/SSP-002 | Indicates that the server implemented all features of the Asset Administration Shell Service Specification Read Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ SubmodelServiceSpecification/SSP-001 | Indicates that the server implemented all features of the Submodel Service Specification Full Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ SubmodelServiceSpecification/SSP-002 | Indicates that the server implemented all features of the Submodel Service Specification Value Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ SubmodelServiceSpecification/SSP-003 | Indicates that the server implemented all features of the Submodel Service Specification Read Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ AasxFileServerServiceSpecification/SSP-001 | Indicates that the server implemented all details of the AASX File Server Service Specification Full Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ RegistryServiceSpecification/SSP-001 | Indicates that the server implemented all details of the Registry Service Specification Full Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ RegistryServiceSpecification/SSP-002 | Indicates that the server implemented all details of the Asset Administration Shell Registry Service Specification Full Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ RegistryServiceSpecification/SSP-003 | Indicates that the server implemented all details of the Asset Administration Shell Registry Service Specification Read Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ RegistryServiceSpecification/SSP-004 | Indicates that the server implemented all details of the Submodel Registry Service Specification Full Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ RegistryServiceSpecification/SSP-005 | Indicates that the server implemented all details of the Submodel Registry Service Specification Read Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ DiscoveryServiceSpecification/SSP-001 | Indicates that the server implemented all details of the Discovery Service Specification Full Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ RepositoryServiceSpecification/SSP-001 | Indicates that the server implemented all details of the Repository Service Specification Full Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ RepositoryServiceSpecification/SSP-002 | Indicates that the server implemented all details of the Repository Service Specification Read Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ AssetAdministrationShellRepositoryServiceSpecification/SSP-001 | Indicates that the server implemented all details of the Asset Administration Shell Repository Service Specification Full Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ AssetAdministrationShellRepositoryServiceSpecification/SSP-002 | Indicates that the server implemented all details of the Asset Administration Shell Repository Service Specification Read Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ SubmodelRepositoryServiceSpecification/SSP-001 | Indicates that the server implemented all details of the Submodel Service Specification Full Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ SubmodelRepositoryServiceSpecification/SSP-002 | Indicates that the server implemented all details of the Submodel Service Specification Read Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ SubmodelRepositoryServiceSpecification/SSP-003 | Indicates that the server implemented all details of the Submodel Service Specification Read Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ SubmodelRepositoryServiceSpecification/SSP-004 | Indicates that the server implemented all details of the Submodel Service Specification Template Profile in version 3.0. |
| https://admin-shell.io/aas/API/3/0/ ConceptDescriptionServiceSpecification/SSP-001 | Indicates that the server implemented all details of the Concept Description Service Specification Read Template Profile in version 3.0. |

An example ServiceDescription object might look like the following, indicating that the server supports two profiles at the same time (see Clause 12.12.3 for further details on service specifications and profiles):



### Simple Data Types

All simple data types from Part 1 [1] apply also to the specifications described in this document. Additional data types are defined in Table 2.

Table 2 Simple Data Types used for API-specific Classes

| **Primitive** | **Definition** | **Value Examples** |
| --- | --- | --- |
| NonNegativeInteger | The *nonNegativeInteger* datatype as defined by XML Schema Part 2 in version 1.0[[4]](#footnote-5) | 0  42 |

### Primitive Data Types

All primitive data types from Part 1 version 3.0 apply also to the specifications described in this document. All constraints and spelling patterns apply as well. In addition, the following data types are defined.

Table 3 Primitive Data Types used for the API-specific Classes

| **Primitive** | **Definition** | **Value Examples** |
| --- | --- | --- |
| CodeType | *string* with max 32 and min 1 characters | “409”  “Bad\_UserAccessDenied” |
| ShortIdType | same as *NameType*  Note: ShortIdType is *not* the data type of idShort attributes but for IDs which shall be shorter than the identifier type. | “02063059-b81c-482b-97d1-d29cbe382ef6”  “my-random-id” |
|  |  |  |

### Status Code, Error Handling & Result Messages

This clause deals with the error and result handling of an operation’s execution in a technology-independent manner.

The first clause covers generic status codes that are returned on each request, independent of the operation’s success or failure. The subsequent clause describes the result object that is returned in case of failure.

#### Generic Status Codes

Successful operations return one of the success status codes and their respective payload. Unsuccessful operations return one of the failure status codes and a result object as defined in Clause 0.

Table 4 shows generic status codes returned to the requester. Additionally, the table indicates whether a specific status code comes with a result object in the returned payload.

Table 4 Status Codes

|  |  |  |
| --- | --- | --- |
| **Generic Status Code** | **Meaning** | **Has Result Object** |
| Success | Success | No |
| SuccessCreated | Successful creation of a new resource | No |
| SuccessAccepted | The reception of the request was successful | No |
| SuccessNoContent | Success with explicitly no content in the payload | No |
| ClientErrorBadRequest | Bad or malformed request | Yes |
| ClientNotAuthorized | Wrong or missing authorization credentials | Yes |
| ClientForbidden | Authorization has been refused | Yes |
| ClientMethodNotAllowed | Operation request is not allowed | Yes |
| ClientErrorResourceNotFound | Resource not found | Yes |
| ClientResourceConflict | Conflict-creating resource (resource already exists) | Yes |
| ServerInternalError | Unexpected error | Yes |
| ServerErrorBadGateway | Bad gateway | Yes |

#### 

#### General Result Object

In case of a failed operation execution, a result object shall be returned containing more information about the reasons why the operation failed to execute.

| **Class Name** | Result | | |
| --- | --- | --- | --- |
| **Explanation** | The result object | | |
| **Inherits from** | -- | | |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/Result/3/0 | | |
| **Attribute** | **Explanation** | **Type** | **Card.** |
| message | Additional message containing information for the requester | Message | 0..\* |

| **Class Name** | Message | | |
| --- | --- | --- | --- |
| **Explanation** | A message containing more information for the requester about a certain happening in the backend | | |
| **Inherits from** | -- | | |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/Message/3/0 | | |
| **Attribute** | **Explanation** | **Type** | **Card.** |
| messageType | The message type | MessageTypeEnum | 1 |
| text | The message text | string | 1 |
| code | Technology-dependent status or error code | CodeType | 0..1 |
| correlationId | Identifier to relate several result messages throughout several systems | ShortIdType | 0..1 |
| timestamp | Timestamp of the message | dateTime | 0..1 |

|  |  |
| --- | --- |
| **Enumeration** | MessageTypeEnum |
| **Explanation** | The message type |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/MessageTypeEnum/3/0 |
| **Literal** | **Explanation** |
| Info | Used to inform the user about a certain fact |
| Warning | Used for warnings; warnings may lead to errors in the subsequent execution |
| Error | Used for handling errors |
| Exception | Used in case of an internal and/or unhandled exception |

#### Operation Objects

The following type definitions are used to call and handle the requests and responses while performing synchronous or asynchronous operation invocation.

##### OperationRequest

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | OperationRequest | | |
| **Explanation** | The operation request object | | |
| **Inherits from** | -- | | |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/OperationRequest/3/0 | | |
| **Attribute** | **Explanation** | **Type** | **Card.** |
| inputArguments | Input argument | OperationVariable | 0..\* |
| inoutputArguments | InOutput argument | OperationVariable | 0..\* |
| clientTimeoutDuration | Duration indicating when the client expects the server to have finished execution of the invoked operation | duration | 0..1 |

##### OperationResult

| **Class Name** | OperationResult | | |
| --- | --- | --- | --- |
| **Explanation** | The operation’s invocation result object | | |
| **Inherits from** | Result | | |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/OperationResult/3/0 | | |
| **Attribute (\* = mandatory)** | **Explanation** | **Type** | **Card.** |
| outputArguments | Output argument | OperationVariable | 0..\* |
| inoutputArguments | InOutput argument | OperationVariable | 0..\* |
| executionState | Execution state | ExecutionState | 1 |
| success | Flag indicating whether the business operation behind the operation was successful (true) or not (false) | boolean | 0..1 |

##### Enumeration ExecutionState

|  |  |
| --- | --- |
| **Enumeration** | ExecutionState |
| **Explanation** | The operation’s invocation result state |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/ExecutionState/3/0 |
| **Literal** | **Explanation** |
| Initiated | The operation is ready to be executed (initial state) |
| Running | The operation is running |
| Completed | The operation is completed |
| Canceled | The operation was cancelled externally |
| Failed | The operation failed |
| Timeout | The operation has timed out due to given client or server timeout |

##### OperationHandle

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | OperationHandle | | |
| **Explanation** | The returned handle of an operation’s asynchronous invocation used to request the current state of the operation’s execution | | |
| **Inherits from** | -- | | |
| **semanticId** | https://admin-shell.io/aas/API/DataTypes/OperationHandle/3/0 | | |
| **Attribute** | **Explanation** | **Type** | **Card.** |
| handleId | Handle id | ShortIdType | 1 |

### File Content

The “File Content” type of the operations mentioned above is seen as “arbitrary binary data” according to RFC 2046 and is as such defined as byte-array in UTF8-encoding. If a content type is required, “application/octet-stream” must be used as defined in RFC 2046.

# Basic Operation Parameters

## General

This clause specifies the parameters for API operations.

## SerializationModifiers in Operations

**Definition**

For GET operations, a SerializationModifier indicates the requester’s expected or desired response content. For PUT and PATCH operations, a SerializationModifier indicates the input content. The SerializationModifier comprises three orthogonal enumerations. When combined, these enumerations influence the input or response content of the requested operation.

Note: values remain unchanged with content=metadata.

1. **Enumeration: Level**

The first enumeration *Level* indicates the depth of the structure of the response or input content.

Table 5 Level Parameters

|  |  |
| --- | --- |
| **Value** | **Explanation** |
| Deep (Default) | All elements of a requested hierarchy level and all children on all sublevels are returned. Children in this sense are SubmodelElements which are contained at the ‘submodelElements’ field of Submodels, the ‘value’ field of SubmodelElementCollections or SubmodelElementLists, the ‘statements’ field of Entities, or the ‘annotations’ field of AnnotatedRelationshipElements. |
| Core | Only elements of a requested hierarchy level as well as direct children are returned. By this, a client can iterate the hierarchy step by step. |

Note: level parameters are mapped to the query parameter “?level” in the HTTP/REST APIs, see also Clause 12.8.

1. **Enumeration: Content**

The second enumeration *Content* indicates the kind of serialization of the response or input content.

For Content equal to Value see Clause 0 for details.

Table 6 Content Parameters

|  |  |
| --- | --- |
| **Value** | **Explanation** |
| Normal (Default) | The standard serialization of the model element or child elements is applied. |
| Metadata | Only metadata of an element or child elements is returned; the value is not . |
| Value | Only the raw value of the model element or child elements is returned; it is commonly referred to as *ValueOnly*-serialization. |
| Reference | Only applicable to Referables. Only the reference to the found element is returned; potential child elements are ignored. |
| Path | Returns the idShort of the requested element and a list of *idShort* paths to child elements if the requested element is a Submodel, a SubmodelElementCollection, a SubmodelElementList, a AnnotatedRelationshipElement, or an Entity. |

Note: level parameters are mapped to path suffixes “/$<content>” in the HTTP/REST APIs, see also Clause 12.8.

1. **Enumeration: Extent**

The third enumeration *Extent* indicates to which extent the response or input content is being serialized. At this stage, the listed values could also be represented as binary values on BLOB-elements. They are, however, kept as generic extent values for the sake of extension.

Table 7 Extent Parameters

|  |  |
| --- | --- |
| **Value** | **Explanation** |
| WithoutBLOBValue (Default) | Only applicable to BLOB-elements; the BLOB content is not returned. |
| WithBLOBValue | Only applicable to BLOB-elements; the BLOB content is returned as *base64*-encoded string. |

Note: level parameters are mapped to the query parameter “?extent” in the HTTP/REST APIs, see also Clause 12.8.

## Applicability of SerializationModifiers

The defined SerializationModifiers are only valid for specific operations due to their generic nature. Also, the applicability depends on the kind of the accessed resource. The following list defines the applicability of the modifiers to the resources.

GET and PATCH operations may combine all SerializationModifiers as listed below. PUT operations may only use the Extent Modifier. POST operations do not use SerializationModifiers.

Table 8 Applicability of SerializationModifiers

| **Resource Name** | **Level Modifier** | **Content Modifier** | **Extent Modifier** |
| --- | --- | --- | --- |
| Asset Administration Shell | No | Normal/Reference | No |
| Submodel Reference | No | No | No |
| Submodel | Deep/Core | Normal/ Metadata/Value/Reference/Path | WithoutBLOBValue/  WithBLOBValue |
| **SubmodelElements** |  |  |  |
| SubmodelElementCollection | Deep/Core | Normal/ Metadata/Value/Reference/Path | WithoutBLOBValue/  WithBLOBValue |
| SubmodelElementList | Deep/Core | Normal/ Metadata/Value/Reference/Path | WithoutBLOBValue/  WithBLOBValue |
| Entity | Deep/Core | Normal/ Metadata/Value/Reference/Path | WithoutBLOBValue/  WithBLOBValue |
| BasicEventElement | No | Normal/ Metadata/Value/Reference | No |
| Capability | No | Normal/Reference | No |
| Operation | No | Normal/Reference | No |
| **DataElements** |  |  |  |
| Property | No | Normal/ Metadata/Value/Reference | No |
| MultilanguageProperty | No | Normal/ Metadata/Value/Reference | No |
| Range | No | Normal/ Metadata/Value/Reference | No |
| ReferenceElement | No | Normal/ Metadata/Value/Reference | No |
| RelationshipElement | No | Normal/ Metadata/Value/Reference | No |
| AnnotatedRelationshipElement | No | Normal/ Metadata/Value/Reference | No |
| Blob | No | Normal/ Metadata/Value/Reference | WithoutBLOBValue/  WithBLOBValue |
| File | No | Normal/ Metadata/Value/Reference | No |

Note: EventPayload defines the necessary information of an event instance sent out or received. It is, however not part of the AAS and submodel hierarchical structure.

## Serialization in Specified Formats (SerializationModifier *Content*)

### General

If the SerializationModifier *Content* is set to **Value**, the ValueOnly-Serialization is used as described below.

Note: to date, only the serialization in JSON has been specified. Other serialization formats (e.g. XML, RDF, etc.) will be defined in future versions of this document.

### ValueOnly-Serialization in JSON

Note: this clause explains how to return the submodel element’s value only if the SerializationModifier *Content* is set to *Value*.

In many cases, applications using data from Asset Administration Shells already know the Submodel regarding its structure, attributes, and semantics. Consequently, there is not always a need to receive the entire model information, which can be requested separately via *Content* modifier set to *Metadata*, in each request since it is constant most of the time. Instead, applications are most likely only interested in the values of the modelled data. Furthermore, having limited processing power or limited bandwidth, one use case of this SerializationModifier is to transfer data as efficiently as possible. Semantics and data might be split into two separate architecture building blocks. For example, a database would suit the needs for querying semantics, while a device would only provide the data at runtime. Two separate requests make it possible to build up a user interface (UI) and show new upcoming values highly efficiently.

Values are only available for

* All subtypes of abstract type *DataElement*,
* SubmodelElementList and SubmodelElementCollection resp. for their included SubmodelElements,
* ReferenceElement,
* RelationshipElement + AnnotatedRelationshipElement,
* Entity,
* BasicEventElement.

Capabilities are excluded from the SerializationModifier’s scope since only data containing elements are in the focus. They are consequently omitted in the serialization.

The following rules shall be adhered to when serializing a submodel with the SerializationModifier *Value*:

* A submodel is serialized as an unnamed JSON object.
* A submodel element is considered a leaf submodel elementif it does not contain other submodel elements. A leaf submodel element follows the rules for the different submodel elements considered in the serialization, as described below. If it is not a leaf element, the serialization rules must be transitively followed until the value is a leaf submodel element.
* For each submodel element:  
  + *Property* is serialized as ${Property/idShort}: ${Property/value} where ${Property/value} is the JSON serialization of the respective property’s value in accordance with the data type to value mapping (see table after this section).
  + *MultiLanguageProperty* is serialized as named JSON object with ${MultiLanguageProperty/idShort} as the name of the containing JSON property. The JSON object contains an array of JSON objects for each language of the *MultiLanguageProperty* with the language as name and the corresponding localized string as value of the respective JSON property. The language name is defined as two chars according to ISO 639-1.
  + *Range* is serialized as named JSON object with ${Range/idShort} as the name of the containing JSON property. The JSON object contains two JSON properties. The first is named “min”. The second is named “max”. Their corresponding values are ${Range/min} and ${Range/max}.
  + *File* and *Blob* are serialized as named JSON objects with ${File/idShort} or ${Blob/idShort}as the name of the containing JSON property. The JSON object contains two JSON properties. The first refers to the content type named ${File/contentType} resp. ${Blob/contentType}. The latter refers to the value named “value” ${File/value} resp. ${Blob/value}. The resulting ValueOnly object is indistinguishable whether it contains File or Blob attributes. Therefore, the receiver needs to take the type of the target resource into account. Since the receiver knows in advance if a File or a Blob SubmodelElement shall be manipulated, it can parse the transferred ValueOnly object accordingly as a File or Blob object.
  + *SubmodelElementCollection* is serialized as named JSON object with ${SubmodelElementCollection/idShort} as the name of the containing JSON property. The elements contained within the struct are serialized according to their respective type with ${SubmodelElement/idShort} as the name of the containing JSON property.
  + *SubmodelElementList* is serialized as a JSON array with the index of the contained SubmodelElement in the list as the position in the JSON array. The elements contained within the list are serialized according to their respective type.
  + *ReferenceElement* is serialized as ${ReferenceElement/idShort}: ${ReferenceElement/value} where ${ReferenceElement/value} is the serialization of the *Reference* class.
  + *RelationshipElement* is serialized as named JSON object with ${RelationshipElement/idShort} as the name of the containing JSON property. The JSON object contains two JSON properties. The first is named “first”. The second is named “second”. Their corresponding values are ${RelationshipElement/first} resp. ${Relationship/second}. The values are serialized according to the serialization of a *ReferenceElement* (see above).
  + *AnnotatedRelationshipElement* is serialized according to the serialization of a *ReleationshipElement* (see above). Additionally, a third named JSON object is introduced with “annotations” as the name of the containing JSON property. The value is ${AnnotatedRelationshipElement/annotations}. The values of the array items are serialized depending on the type of the annotation data element.
  + *Entity* is serialized as named JSON object with ${Entity/idShort} as the name of the containing JSON property. The JSON object contains three JSON properties. The first is named “statements” ${Entity/statements} and contains an array of the serialized submodel elements according to their respective serialization mentioned in this clause. The second is named either “globalAssetId” or “specificAssetId” and contains either a *Reference* (see above) or a *SpecificAssetId*. The third property is named “entityType” and contains a string representation of ${Entity/entityType}.
  + *BasicEventElement* is serialized as named JSON object with ${BasicEventElement/idShort} as the name of the containing JSON property. The JSON object contains one JSON property named “observed” with the corresponding value of ${BasicEventElement/observed} as the standard serialization of the *Reference* class.
  + *SpecificAssetId* is serialized as named JSON object with three JSON properties named as the attributes of *SpecificAssetId.*
* Submodel elements defined in the submodel other than the ones mentioned above are not subject to serialization of that SerializationModifier.

**Data type to value mapping**[[5]](#footnote-6)

The serialization of submodel element values is described in the following table. The left column “Data Type” shows the data types which can be used for submodel element values. The data types are defined according to the W3C XML Schema (https://www.w3.org/TR/xmlschema-2/#built-in-datatypes and https://www.w3.org/TR/xmlschema-2/#built-in-derived). “Value Range” further explains the possible range of data values for this data type. The right column comprises related examples of the serialization of submodel element values.

Table 9 Mapping of Data Types in ValueOnly-Serialization

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Data Type** | **JSON Type** | **Value Range** | **Sample Values** |
| Core Types | xs:string | string | Character string | "Hello world", "Καλημέρα κόσμε", "コンニチハ" |
| xs:boolean | boolean | true, false | true, false |
| xs:decimal | number | Arbitrary-precision decimal numbers | -1.23, 126789672374892739424.543233, 100000.00, 210 |
| xs:integer | number | Arbitrary-size integer numbers | -1, 0, 126789675432332938792837429837429837429, 100000 |
| IEEE-floating-point numbers | xs:double | number | 64-bit floating point numbers | -1.0, -0.0, 0.0, 234.567e8, 234.567e+8, 234.567e-8 |
| xs:float | number | 32-bit floating point numbers | -1.0, -0.0, 0.0, 234.567e8, 234.567e+8, 234.567e-8 |
| Time and data | xs:date | string | Dates (yyyy-mm-dd) with or without time zone | "2000-01-01","2000-01-01Z", "2000-01-01+12:05" |
| xs:time | string | Times (hh:mm:ss.sss…​) with or without time zone | "14:23:00", "14:23:00.527634Z", "14:23:00+03:00" |
| xs:dateTime | string | Date and time with or without time zone | "2000-01-01T14:23:00", "2000-01-01T14:23:00.66372+14:00" |
| xs:dateTimeStamp | string | Date and time with required time zone | "2000-01-01T14:23:00.66372+14:00" |
| Recurring and partial dates | xs:gYear | string | Gregorian calendar year | "2000", "2000+03:00" |
| xs:gMonth | string | Gregorian calendar month | "--04", "--04+03:00" |
| xs:gDay | string | Gregorian calendar day of the month | "---04", "---04+03:00" |
| xs:gYearMonth | string | Gregorian calendar year and month | "2000-01", "2000-01+03:00" |
| xs:gMonthDay | string | Gregorian calendar month and day | "--01-01", "--01-01+03:00" |
| xs:duration | string | Duration of time | "P30D", "-P1Y2M3DT1H", "PT1H5M0S" |
| xs:yearMonthDuration | string | Duration of time (months and years only) | "P10M", 'P5Y2M" |
| xs:dayTimeDuration | string | Duration of time (days, hours, minutes, seconds only) | "P30D", 'P1DT5H", 'PT1H5M0S" |
| Limited-range integer numbers | xs:byte | number | -128…+127 (8 bit) | -1, 0, 127 |
| xs:short | number | -32768…+32767 (16 bit) | -1, 0, 32767 |
| xs:int | number | 2147483648…+2147483647 (32 bit) | -1, 0, 2147483647 |
| xs:long | number | -9223372036854775808…+9223372036854775807 (64 bit) | -1, 0, 9223372036854775807 |
| xs:unsignedByte | number | 0…255 (8 bit) | 0, 1, 255 |
| xs:unsignedShort | number | 0…65535 (16 bit) | 0, 1, 65535 |
| xs:unsignedInt | number | 0…4294967295 (32 bit) | 0, 1, 4294967295 |
| xs:unsignedLong | number | 0…18446744073709551615 (64 bit) | 0, 1, 18446744073709551615 |
| xs:positiveInteger | number | Integer numbers >0 | 1, 7345683746578364857368475638745 |
| xs:nonNegativeInteger | number | Integer numbers ≥0 | 0, 1, 7345683746578364857368475638745 |
| xs:negativeInteger | number | Integer numbers <0 | -1, -23487263847628376482736487263847 |
| xs:nonPositiveInteger | number | Integer numbers ≤0 | -1, 0, -93845837498573987498798987394 |
| Encoded binary data | xs:hexBinary | string | Hex-encoded binary data | "6b756d6f77617368657265" |
| xs:base64Binary | string | base64-encoded binary data | "a3Vtb3dhc2hlcmU=" |
| Miscellaneous types | xs:anyURI | string | Absolute or relative URIs and IRIs | "http://customer.com/demo/aas/1/1/1234859590", "urn:example:company:1.0.0" |
| rdf:langString | string | Strings with language tags | "'Hello'@en", "'Hallo'@de"  Note: the examples are written in RDF/Turtle syntax, and only "Hello" and "Hallo" are the actual values. |

The following types defined by the XSD and RDF specifications are explicitly omitted for serialization:

xs:language, xs:normalizedString, xs:token, xs:NMTOKEN, xs:Name, xs:NCName, xs:QName, xs:ENTITY, xs:ID, xs:IDREF, xs:NOTATION, xs:IDREFS, xs:ENTITIES, xs:NMTOKENS, rdf:HTML and rdf:XMLLiteral.

Note 1: due to the limits in the representation of numbers in JSON, the maximum integer number that can be used without losing precision is 2⁵³-1 (defined as Number.MAX\_SAFE\_INTEGER). Even if the used data type would allow higher or lower values, they cannot be used if they cannot be represented in JSON. Affected data types are unbounded numeric types xs:decimal, xs:integer, xs:positiveInteger, xs:nonNegativeInteger, xs:negativeInteger, xs:nonPositiveInteger and the bounded type xs:unsignedLong. Other numeric types are not affected.[[6]](#footnote-7)

Note 2: the ValueOnly-serialization uses JSON native data types, AAS in general uses XML Schema Built-in Datatypes for Simple Data Types and ValueDataType. In case of booleans, JSON accepts only literals true and false, whereas xs:boolean also accepts 1 and 0, respectively. In case of double, JSON number is used in ValueOnly, but JSON number does not support INF/-INF (positive Infinity/negative), which is supported by xs:double. Furthermore, NaN (Not a Number) is also not supported.  
(See <https://datatracker.ietf.org/doc/html/rfc8259#section-6> )

Note 3: language-tagged strings (rdf:langString) containing single quotes (‘) or double quotes (“) are not supported.

**Examples conformant to [1]:**

Full serialization of single submodel element *Property*:



With the SerializationModifier set to *Value,* the payload is minimized to the following:



For a *SubmodelElementCollection,* the struct is serialized as objects denoted by curly brackets:



For a *SubmodelElementList,* the struct is serialized as array denoted by square brackets:



For a *MultiLanguageProperty* named “Label”, the payload is minimized to the following:



Note: in accordance with IETF [RFC 5646](https://tools.ietf.org/html/rfc5646#page-5), the language names match the following regular expression:

^[a-z]{2,4}(-[A-Z][a-z]{3})?(-([A-Z]{2}|[0-9]{3}))?$

For a *Range* named “TorqueRange”, the payload is minimized to the following:



For a *ReferenceElement* named “MaxRotationSpeedReference”, the payload is minimized to the following:

For the same *ReferenceElement,* the payload is minimized to the following in case the *Reference* is of subtype *GlobalReference*:



For a *File* named “Document”, the payload is minimized to the following:



For a *Blob* named “Library”, the payload is minimized to the following if the SerializationModifier *Extent* is set to ***WithoutBLOBValue***



If the SerializationModifier Extent is set to ***WithBlobValue***, there is an additional attribute containing the base64-encoded value:



For a *RelationshipElement* named “CurrentFlowsFrom”, the payload is minimized to the following:

For an *AnnotatedRelationshipElement* named “CurrentFlowFrom”, with an annotated *Property*-DataElement “AppliedRule”, the payload is minimized to the following:



For an *Entity* named “MySubAssetEntity”, the payload is minimized to the following:



For a BasicEventElement named “MyBasicEvent”, the payload is minimized to the following:



### JSON-Schema for the ValueOnly-Serialization

The following JSON-Schema represents the validation schema for the ValueOnly-Serialization of submodel elements. This holds true for all submodel elements mentioned in the previous clause except for *SubmodelElementCollections*. Since *SubmodelElementCollections* are treated as objects containing submodel elements of any kind, the integration into the same validation schema would result in a circular reference or ambiguous results ignoring the actual validation of submodel elements other than *SubmodelElementCollections*. Hence, the same validation schema must be applied for each *SubmodelElementCollection* within a submodel element hierarchy. In this case, it may be necessary to create a specific JSON-Schema for the individual use case. The *SubmodelElementCollection* is added to the following schema for completeness and clarity. It is, however, not referenced from the *SubmodelElementValue*-oneOf-Enumeration due to the reasons mentioned above.  
See Annex B for an example that validates against this schema.

|  |
| --- |
| {  "$schema": "https://json-schema.org/draft/2019-09/schema",  "title": "ValueOnly-Serialization-Schema",  "$id": "https://admin-shell.io/schema/valueonly/json/V3.0",  "definitions": {  "AnnotatedRelationshipElementValue": {  "type": "object",  "properties": {  "first": {  "$ref": "#/definitions/ReferenceValue"  },  "second": {  "$ref": "#/definitions/ReferenceValue"  },  "annotation": {  "type": "array",  "items": {  "$ref": "#/definitions/ValueOnly"  }  }  },  "required": [  "first",  "second",  "annotation"  ],  "additionalProperties": **false**  },  "BasicEventElementValue": {  "type": "object",  "properties": {  "observed": {  "$ref": "#/definitions/ReferenceValue"  }  },  "required": [  "observed"  ],  "additionalProperties": **false**  },  "BlobValue": {  "type": "object",  "properties": {  "contentType": {  "type": "string",  "minLength": "1",  "maxLength": "100"  },  "value": {  "type": "string"  }  },  "required": [  "contentType",  "value"  ],  "additionalProperties": **false**  },  "BooleanValue": {  "type": "boolean",  "additionalProperties": **false**  },  "EntityValue": {  "type": "object",  "properties": {  "statements": {  "$ref": "#/definitions/ValueOnly"  },  "entityType": {  "enum": [  "SelfManagedEntity",  "CoManagedEntity"  ]  },  "globalAssetId": {  "type": "string"  },  "specificAssetIds": {  "type": "array",  "items": {  "$ref": "#/definitions/SpecificAssetIdValue"  }  }  },  "required": [  "statements",  "entityType"  ],  "additionalProperties": **false**  },  "FileValue": {  "type": "object",  "properties": {  "contentType": {  "type": "string",  "minLength": "1",  "maxLength": "100"  },  "value": {  "type": "string",  "minLength": "1",  "maxLength": "200"  }  },  "required": [  "contentType",  "value"  ],  "additionalProperties": **false**  },  "Identifier": {  "type": "string"  },  "Key": {  "type": "object",  "properties": {  "type": {  "type": "string"  },  "value": {  "type": "string"  }  },  "required": [  "type",  "value"  ],  "additionalProperties": **false**  },  "LangString": {  "type": "object",  "patternProperties": {  "^[a-z]{2,4}(-[A-Z][a-z]{3})?(-([A-Z]{2}|[0-9]{3}))?$": {  "type": "string"  }  },  "additionalProperties": **false**  },  "MultiLanguagePropertyValue": {  "type": "array",  "items": {  "$ref": "#/definitions/LangString"  },  "additionalProperties": **false**  },  "NumberValue": {  "type": "number",  "additionalProperties": **false**  },  "OperationRequestValueOnly": {  "inoutputArguments": {  "$ref": "#/definitions/ValueOnly"  },  "inputArguments": {  "$ref": "#/definitions/ValueOnly"  },  "timestamp": {  "type": "string",  "pattern": "^-?(([1-9][0-9][0-9][0-9]+)|(0[0-9][0-9][0-9]))-((0[1-9])|(1[0-2]))-((0[1-9])|([12][0-9])|(3[01]))T(((([01][0-9])|(2[0-3])):[0-5][0-9]:([0-5][0-9])(\\.[0-9]+)?)|24:00:00(\\.0+)?)(Z|\\+00:00|-00:00)$"  },  "additionalProperties": **false**  },  "OperationResultValueOnly": {  "executionState": {  "type": "string",  "enum": ["Initiated", "Running", "Completed", "Canceled", "string",   "Failed", "Timeout"]  },  "inoutputArguments": {  "$ref": "#/definitions/ValueOnly"  },  "outputArguments": {  "$ref": "#/definitions/ValueOnly"  },  "additionalProperties": **false**  },  "PropertyValue": {  "oneOf": [  {  "$ref": "#/definitions/StringValue"  },  {  "$ref": "#/definitions/NumberValue"  },  {  "$ref": "#/definitions/BooleanValue"  }  ]  },  "RangeValue": {  "type": "object",  "properties": {  "min": {  "type": "number"  },  "max": {  "type": "number"  }  },  "required": [  "min",  "max"  ],  "additionalProperties": **false**  },  "ReferenceElementValue": {  "$ref": "#/definitions/ReferenceValue"  },  "ReferenceValue": {  "type": "object",  "properties": {  "type": {  "type": "string",  "enum": ["ModelReference", "ExternalReference"]  },  "keys": {  "type": "array",  "items": {  "$ref": "#/definitions/Key"  }  }  },  "additionalProperties": **false**  },  "RelationshipElementValue": {  "type": "object",  "properties": {  "first": {  "$ref": "#/definitions/ReferenceValue"  },  "second": {  "$ref": "#/definitions/ReferenceValue"  }  },  "required": [  "first",  "second"  ],  "additionalProperties": **false**  },  "SpecificAssetIdValue": {  "type": "object",  "patternProperties": {  "(.\*?)": {  "type": "string"  }  }  },  "StringValue": {  "type": "string",  "additionalProperties": **false**  },  "SubmodelElementCollectionValue": {  "$ref": "#/definitions/ValueOnly"  },  "SubmodelElementListValue": {  "type": "array",  "items": {  "$ref": "#/definitions/SubmodelElementValue"  }  },  "SubmodelElementValue": {  "oneOf": [  {  "$ref": "#/definitions/BasicEventElementValue"  },  {  "$ref": "#/definitions/RangeValue"  },  {  "$ref": "#/definitions/MultiLanguagePropertyValue"  },  {  "$ref": "#/definitions/FileBlobValue"  },  {  "$ref": "#/definitions/ReferenceElementValue"  },  {  "$ref": "#/definitions/RelationshipElementValue"  },  {  "$ref": "#/definitions/AnnotatedRelationshipElementValue"  },  {  "$ref": "#/definitions/EntityValue"  },  {  "$ref": "#/definitions/PropertyValue"  },  {  "$ref": "#/definitions/SubmodelElementListValue"  }  ]  },  "ValueOnly": {  "propertyNames": {  "pattern": "^[A-Za-z\_][A-Za-z0-9\_-]\*$"  },  "patternProperties": {  "^[A-Za-z\_][A-Za-z0-9\_-]\*$": {  "$ref": "#/definitions/SubmodelElementValue"  }  },  "additionalProperties": **false**  }  }  } |

### IdShortPath Serialization

To get only the idShort paths of a submodel element hierarchy, the serialization format is specified in terms of an idShortPath notation to be returned in an unnamed JSON-array. The notation differs depending on whether a SubmodelElementCollection or a SubmodelElementList is used. In the first case, the submodel element’s idShort is separated by “.” (dot) from top level down to child level. In the second case, square brackets with an index “[<<index>>]” are appended after the idShort of the containing SubmodelElementList.

In the following example, where a request for idShort paths starts at *MySubmodelElementCollection* with SerializationModifier level = deep, the list of idShort paths is returned as follows:

Submodel: MySubmodel

* Property: MyTopLevelProperty
* SMC: MySubmodelElementCollection
  + Property: MySubProperty1
  + Property: MySubProperty2
  + SMC: MySubSubmodelElementCollection
    - Property: MySubSubProperty1
    - Property: MySubSubProperty2
  + SML: MySubSubmodelElementList1
    - Property: “MySubTestValue1”,
    - Property: “MySubTestValue2”,
  + SML: MySubSubmodelElementList2
    - SML
      * Property: “MySubTestValue3”



# HTTP/REST API

## General

This clause describes the technology mapping to HTTP/REST APIs.

The OpenAPI specification of the HTTP/REST APIs can be found at SwaggerHub.

To clearly separate the different parts of the AAS model, the model has been split into several HTTP/REST APIs. Combinations then form service specifications and profiles, each materialized as an individual OpenAPI document.

The schema for the metamodel of Part 1 is available at:   
[https://app.swaggerhub.com/domains/Plattform\_i40/Part1-MetaModel-Schemas/V3.0#](https://app.swaggerhub.com/domains/Plattform_i40/Part1-MetaModel-Schemas/V3.0)   
This schema includes general objects, which are used in the further defined APIs.

Additional objects are needed for Part 2, e.g. for the ValueOnly-Serialization or the descriptors for the registry. The related schema of Part 2 objects is available at:   
[https://app.swaggerhub.com/domains/Plattform\_i40/Part2-API-Schemas/V3.0#](https://app.swaggerhub.com/domains/Plattform_i40/Part2-API-Schemas/V3.0)   
This schema includes general objects, which are used in the further defined APIs.

The definition of endpoints is based on DIN SPEC 16593. The related schema for DIN SPEC 16593 is available at: [https://app.swaggerhub.com/domains/Plattform\_i40/DINSPEC16593-Schemas/V3.0#](https://app.swaggerhub.com/domains/Plattform_i40/DINSPEC16593-Schemas/V3.0)   
This schema includes general objects, which are used in the further defined APIs. These objects are the basis for the definition of Part 2 APIs.

The AAS API including the Submodel API, Serialization API, and Self-Description API is available at:   
[https://app.swaggerhub.com/apis/Plattform\_i40/AssetAdministrationShellServiceSpecification/V3.0\_SSP-001#](https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellServiceSpecification/V3.0_SSP-001)   
This is a combination of APIs, which forms a service specification according to the Industrie 4.0 Service Model in Clause 4.1.

The Submodel API including the Serialization API, and Self-Description API is available at:   
[https://app.swaggerhub.com/apis/Plattform\_i40/SubmodelServiceSpecification/V3.0\_SSP-001#](https://app.swaggerhub.com/apis/Plattform_i40/SubmodelServiceSpecification/V3.0_SSP-001)   
This is a combination of APIs, which forms a service specification according to the Industrie 4.0 Service Model in Clause 4.1.

The AAS Repository API including AAS API, Submodel API, Submodel Repository API, Serialization API, and Self-Description API is available at: [https://app.swaggerhub.com/apis/Plattform\_i40/RepositoryServiceSpecification/V3.0\_SSP-001#](https://app.swaggerhub.com/apis/Plattform_i40/RepositoryServiceSpecification/V3.0_SSP-001)   
This is a combination of APIs, which forms a service specification according to the Industrie 4.0 Service Model in Clause 4.1.

The Registry and Discovery APIs are independent from the other APIs. An AAS Registry including an AAS Registry API, Submodel Registry API, and Self-Description API is available at:  
[https://app.swaggerhub.com/apis/Plattform\_i40/RegistryServiceSpecification/V3.0\_SSP-001#](https://app.swaggerhub.com/apis/Plattform_i40/RegistryServiceSpecification/V3.0_SSP-001)   
The Discovery API including the Self-Description API is available at:  
[https://app.swaggerhub.com/apis/Plattform\_i40/DiscoveryServiceSpecification/V3.0\_SSP-001#](https://app.swaggerhub.com/apis/Plattform_i40/DiscoveryServiceSpecification/V3.0_SSP-001)   
Both are a combination of APIs, which form a service specification according to the Industrie 4.0 Service Model in Clause 4.1.

This clause gives an overview of the HTTP/REST API and describes general design decisions.

## Design Decisions

The following design decisions and constraints hold for the HTTP/REST API:

* OpenAPI and Swaggerhub shall be used for specification. This leads to the constraint that one operation can only provide one type of a resulting payload.
* This document assumes version 1.1 of HTTP.
* An endpoint of the HTTP/REST API shall always use HTTPS (Port 443) with an up-to-date level of encryption.
* The SerializationModifier “content” changes the type the of payload for inputs or results. To ensure type-safe APIs, this parameter is mapped to the path suffixes “/$value”, “/$metadata”, “/$reference”, and “/$path”. “content=Normal” is mapped to the path without any “/$<content>” suffix.
* Generic SerializationModifiers changing the size of payload for input or result have been mapped to corresponding query parameters, e.g. ”?level=” or “?extent=”.
* Query parameters are also used when the type of a resulting payload is a list of objects and the type remains the same, while the query parameter filters the content of the list, e.g. GetAllSubmodels with optional query parameters “?semanticId=” or “?idShort=”.
* Complete objects are provided as requested payloads, e.g. a complete submodel. This corresponds to the generic SerializationModifier content=”Normal”. Reduced objects can be requested by the path suffix “/$<content>”. See Clause 12.5 for further details. Exceptions to this rule are API Operations requiring pagination and error cases.
* By default, blobs are not part of the payload. Using ?extent=WithBLOBValue includes blobs for submodel elements of kind BLOB.
* Submodels define a hierarchical structure. Certain operations use an idShort-path to access deeper parts in the hierarchy. To easily support this in the REST API, “.” or “[index]” is used as a delimiter in the idShort-paths. Please see Clause 12.3. Since an idShort-path could include square brackets like “[index]”, the idShort-path must be URL-encoded.
* Identifiers of Identifiables are base64url-encoded to be passed to the HTTP/REST API (see https://www.base64url.com/). These may be identifiers for Asset Administration Shells, Submodels, or Concept Descriptions.  
  Identifiers may also be passed as base64url-encoded query parameters, e.g. for semanticId or assetId. Such query parameters are typically used when a list of objects may be retrieved in the resulting payload. A list of base64url-encoded ids is simply passed as comma-separated query parameters.
* Please note that base64url-encoding differs slightly from base64-encoding and has been specifically defined for passing URLs. An appropriate base64url implementation needs to be used for encoding/decoding. See RFC 4648 for further details.
* When base64url or base64-encoding is mentioned in connection with string values (e.g. Identifiers), the UTF-8 decoded byte array representation of that string is used for the base64url or base64-encoding.
* When retrieving AssetAdministrationShells (/shells, /lookup/shells), a query parameter “?assetids=” can be specified. Such assetId may be a globalAssetId or specificAssetId. The corresponding key-value-pair is first serialized to JSON and then base64url-encoded. The resulting encoded string is the value of “?assetids=”.
* In some operations, references are part of the query parameters e.g. “?semanticId=”. The corresponding reference is first serialized to JSON and then base64url-encoded. The resulting encoded string is the value of “?semanticId=”.
* Even though the metamodel of the AAS distinguishes between the attributes “semanticId” and “supplementalSemanticId”, the query parameter “?semanticId” targets both.
* This encoding (serialize to JSON + base64url) is also used for SpecificAssetIds, i.e. for GetAllAssetAdministrationShellIdsByAssetLink (/lookup/shells). For the example “[{"key": "globalAssetId","value": "http://example.company/myAsset"},{"key": "myOwnInternalAssetId","value": "12345ABC"}]”, the resulting base64url-encoded value of the query parameter is  
  “?assetIds=W3sia2V5IjogImdsb2JhbEFzc2V0SWQiLCJ2YWx1ZSI6ICJodHRwOi8vZXhhbXBsZS5jb21wYW55L215QXNzZXQifSx7ImtleSI6ICJteU93bkludGVybmFsQXNzZXRJZCIsInZhbHVlIjogIjEyMzQ1QUJDIn1d”.  
  If several key-value-pairs are included, all must be part of the key-value-pairs on the server.
* Comparisons of idShort are made case-sensitive in the HTTP/REST API to avoid repeating toupper()/tolower() conversions.

Note: this is conformant to the change made in Part 1 V3.0 [1].

* GetAll…-API Operations will retrieve a list of objects as the resulting payload, e.g. GetAllSubmodelElements.
* The splitting of big result sets into smaller pieces, commonly referred to as “pagination”, is executed using the cursor query parameter. Therefore, result objects for GetAll…-API Operations and others requiring pagination return their content inside a Result structure. See Clause 12.6 for further explanations.
* In general, only GET, POST, PUT, PATCH and DELETE are used. POST is used to create new objects and to invoke operations.
* Some interfaces may be combined in a so-called “superpaths”, e.g. the Asset Administration Shell Repository Interface may be combined with the AAS Interface and the Submodel Interface. This results in a complete path like “/shells/{aas-identifier}/submodels/{submodel-identifier}/\*”. This is especially useful when all data is hosted in the same repository. Superpaths are defined as part of the service specifications and profiles.
* The attribute AssetAdministrationShell/submodels (array of References) maps to the path segment “/submodel-refs” to distinguish it from the superpath segment “/submodels” (array of Submodels).
* Each interface includes a “/description” operation for self-discovery to provide detailed information about the interface. A server supporting the HTTP/REST API may also provide a server global “/description” to provide the information about all available profiles on that server.
* The recursive nature of the reference class (Reference/referredSemanticId points to Reference again) cannot be represented in SwaggerHub due to a bug in the SwaggerUI code. Therefore, the additional class “ReferenceParent" has been added. “ReferenceParent" shall not be used in productive operations and is only a placeholder for “Reference”. When implementing generated code originating from the SwaggerHub schemas, please delete “ReferenceParent” and add its attributes to “Reference”.

## API Versioning

API versioning provides a way to deal with different versions of the same API at the same time. This way, older versions may still be accessible on the same server to provide services to legacy clients without breaking existing functionality.

There are different solutions regarding API versioning involving URL-based versioning, query parameter-based versioning, as well as HTTP header-oriented solutions using custom or standard headers.  
As different solutions also provide different advantages and disadvantages, **URL-based versioning** has been selected as the most suitable method for the AAS API. Among other advantages, implementation complexity on clients as well as servers is rather low and different versions can be easily accessed through browsers without the need for specific development tools or extensions.

Ein Bild, das Dunkel enthält.

Automatisch generierte Beschreibung

Figure 4 - Generic URL Scheme for AAS API Versioning

Upcoming implementations of AAS related servers need to implement the version prefix “**api/v<X.Y>/**” to provide information of the specific major version regarding AAS Part 2 version, where <X> denotes the implemented major version and <Y> denotes the minor version, e.g. “api/v3.0/” (see Figure 4).

**Note:** all URLs mentioned in this document regarding the REST mapping of the AAS APIs have to be understood with this prefix in mind.

The versioning scheme for AAS API related services follows semantic versioning[[7]](#footnote-8). Very briefly, this defines version numbers as a format following: <MAJOR>.<MINOR>.<PATCH>.

The major version changes in case of breaking or incompatible changes that need to be addressed by clients. Minor versions add (new) functionality in a backwards compatible way and allow clients with lower minor versions to keep their existing functionality. Patch versions only include backwards compatible bug fixes.

AAS API versioning uses the major and minor version as described above. A specific AAS API version uses specific related versions of the metamodel as defined in Clause 1.2. AAS API versions with the same major version must remain compatible, i.e. a client written for an older or a newer minor version must still work. This requires corresponding testing of clients and servers.

Additionally, “Release candidates” are variants of the implementation of the denoted major version. For example, “3.1.0 RC2” should be interpreted as the second (alternative) release candidate for version 3.1.0. This will still result in the version prefix “/api/v3.1/”.

As multiple versions will be supported in the future, an AAS ecosystem consisting of Registry / Discovery services as well as AAS Repository, Submodel (standalone), or AAS (standalone) services should share a consistent version. Therefore, a consistent interface description in the form of OpenAPI documents shall be provided with each major version.

Upcoming compatibility constraints regarding newer versions will be elaborated in further iterations of this document and related technical descriptions (OpenAPI specification).

Finally, it is recommended to include an additional "/description” endpoint into each service to further denote information about APIs / servers capabilities. This endpoint provides further information about the API and its supported profiles. The “/description” will be extended with additional information in later versions.

## Addressing Resources

The API allows to address each referable element, either by its global identifier or by its idShort-path depending on the object type.

If the referable element is an identifiable, it can only be addressed by the global identifier of the object. All other referable elements are addressable by the idShort-path.  
The idShort-path is a chain of idShorts or SubmodelElementList-indexes, which points to an element within a hierarchy of elements. The root of the idShort-path is always a submodel and the first element in an idShort-path is always an idShort of a first level SubmodelElement within a Submodel. Technically, the idShort path is a string and the idShorts are separated by a dot while the SubmodelElementList-indexes are written in brackets.

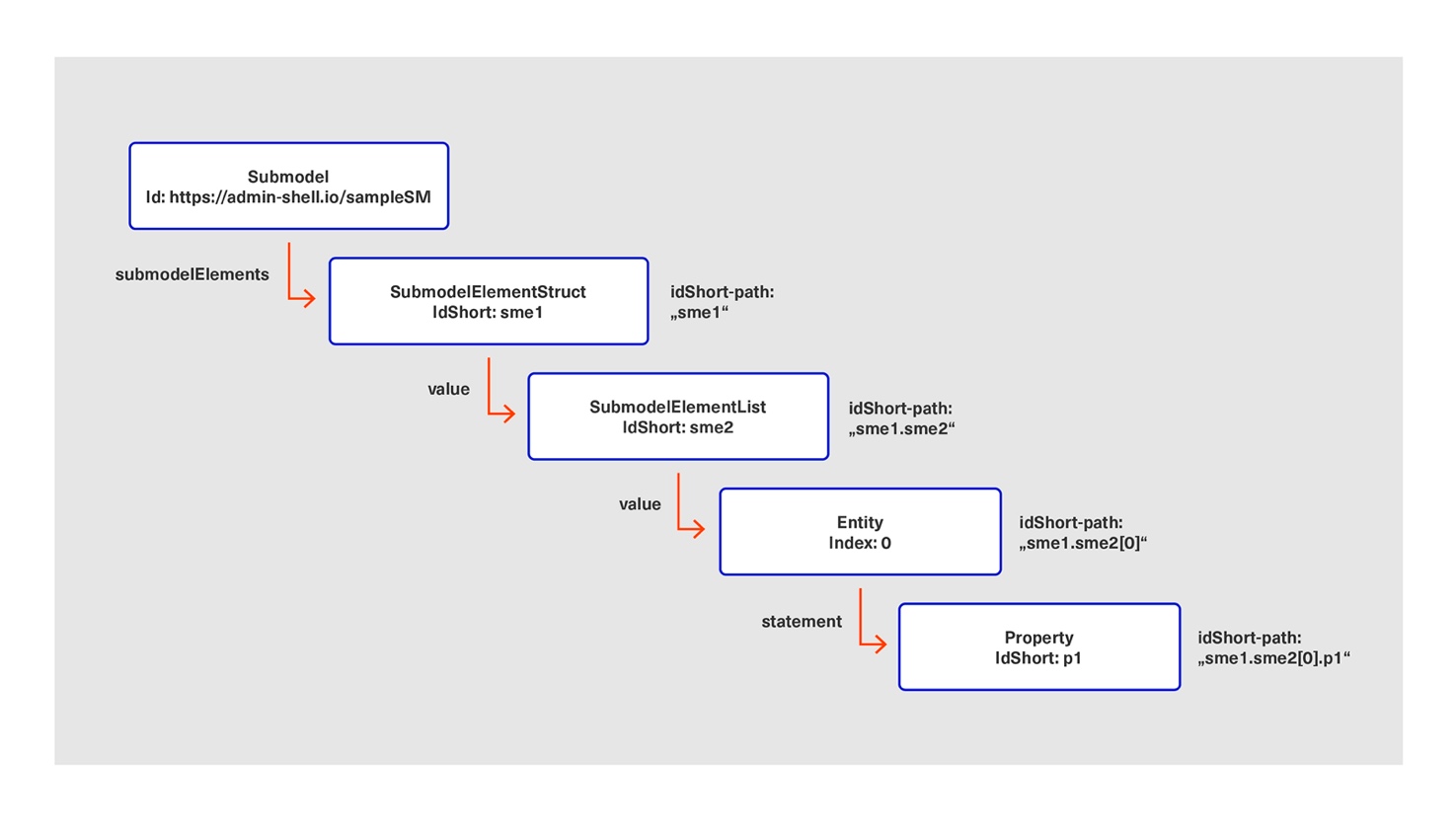


Figure 5 Example Hierarchy of Submodel Elements

The example hierarchy in Figure 5 shows a Submodel with a hierarchical structure of SubmodelElements. The submodel can be addressed by its global identifier “https://admin-shell.io/sampleSM”. The other elements in the figure do not have a global identifier; they are, however, uniquely identifiable and addressable by the submodel identifier and the idShort-path. The idShort-path in this example pointing to the Property p1 is “sme1.sme2[0].p1”. The hierarchy is built on parent-child relations between the elements. There are four elements which can aggregate SubmodelElements and create deeper hierarchal structures. The elements are Submodel, SubmodelElementCollection, SubmodelElementList, and Entity. The fields used to navigate to a deeper level of the hierarchy can be seen in the following table.

Table 10 Children of certain objects

|  |  |
| --- | --- |
| **Element Name** | **Child aggregation field name** |
| Submodel | SubmodelElement |
| SubmodelElementCollection | value |
| SubmodelElementList | value |
| AnnotatedRelationshipElement | annotations |
| Entity | statements |

**Example requests:**

GET /submodels/aHR0cHM6Ly9hZG1pbi1zaGVsbC5pby9zYW1wbGVTTQ/submodel/submodelElements/ sme1.sme2%5B0%5D.p1

Add a new Property to the Entity statements:

POST /submodels/aHR0cHM6Ly9hZG1pbi1zaGVsbC5pby9zYW1wbGVTTQ/submodel/submodelElements/ sme1.sme2%5B0%5D

Note 1: to avoid problems with IRI values in URLs, the identifiers shall be base64url-encoded before using them as parameters in the HTTP-APIs. IdshortPaths are base64url-encoded to also allow square brackets.

Note 2: in the example above, “aHR0cHM6Ly9hZG1pbi1zaGVsbC5pby9zYW1wbGVTTQ” is the base64url-encoding of “https://admin-shell.io/sampleSM”, “sme1.sme2%5B0%5D.p1” is the URL-encoding of “sme1.sme2[0].p1”, and “sme1.sme2%5B0%5D” is the URL-encoding of “sme1.sme2[0]”.

## Metadata Objects

Metadata objects are defined for scenarios where a client only wants to access the metadata of an object, but not the value. **Metadata objects are only part of HTTP/REST and do not change the metamodel.** Metadata objects are used to reduce the payload response to a minimum and to avoid the recursive traversing through the data model when not needed. In many cases, a client is not interested in each child element or value of a resource, but only in the resource itself.

A metadata object does not contain any additional fields in relation to its full object representation, only some fields are left off. The left off fields are fields which could be requested by an own API call and may consist of a recursive or potentially large substructure. The serialization of a metadata object is the same as for the original full object, but without the left off fields.

Table 11 Metadata Attributes

|  |  |
| --- | --- |
| **Class Name** | **Fields not available in metadata representation** |
| **Identifiables** | |
| AssetAdministrationShell | assetInformation, submodels |
| Submodel | submodelElements |
| **SubmodelElements** | |
| SubmodelElementCollection | value |
| SubmodelElementList | value |
| Entity | statements, globalAssetId, specificAssetId |
| BasicEventElement | observed |
| Capability | -- |
| Operation | -- |
| **DataElements** | |
| Property | value, valueId |
| MultilanguageProperty | value, valueId |
| Range | min, max |
| ReferenceElement | value |
| RelationshipElement | first, second |
| AnnotatedRelationshipElement | first, second, annotations |
| Blob | value, contentType |
| File | value, contentType |

**Example**

The example shows a JSON serialization of an AssetAdministrationShell object in its full representation and how it looks like in a metadata representation.

Note: for editorial reasons, some fields which are the same for both representations are omitted.

Table 12 AssetAdministrationShell JSON Serialization Example

* {
* "idShort": "TestAssetAdministrationShell",
* "description": […],
* "id": {…},
* **…**
* "derivedFrom": {…}
* "assetInformation": {…},
* "submodels": […]
* }

Table 13 AssetAdministrationShell Metadata JSON Serialization Example

* {
* "idShort": "TestAssetAdministrationShell",
* "description": […],
* "id": {…}
* **…**
* "derivedFrom": {…}
* }
* }

## Pagination

Pagination is a commonly used pattern to break down potentially long result lists into smaller pieces for a better control of the network and computational load on both the server and the client side. For instance, the OData protocol [8] provides guidelines for parameters and behavior on the client and server side. In addition, the proposals of the RFC 8977[[8]](#footnote-9) present a best practice for web APIs. In the scope of the AAS HTTP/REST API, the query parameter “cursor” controls, which part of a longer result set is returned.

The AAS client may decide on the appropriate size of the result list through the limit parameter. If it is not specified, the server must comply to the default value or explicitly indicate it in the response object.

Pagination is currently only defined for the HTTP/REST API. Other APIs might introduce different patterns to control the response content.

Pagination is controlled by the client via the query parameters “cursor” and “limit”. They can be combined with all other query parameters as defined in this document and listed in the following table:

Table 14 Parameters for Pagination

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Values** | **Default** | **Explanation** |
| Cursor | string | - | The position from which to resume a result listing. The value may be base64url-encoded and contain additional information which helps the server to respond more efficiently. However, the client must not expect any meaning and treat the cursor value as an arbitrary character sequence.  The server must interpret a missing cursor as if the client wants to retrieve the first part of the result set. |
| Limit | nonNegativeInteger | 100 | The maximum size of the result list. |

Constraint AASa-001: The value of the cursor query parameter must not be empty. If the client does not know the cursor value, it must omit the whole query parameter in the request.

Note 1: this constraint prohibits that an empty cursor value is sent by the client, e.g. *…?cursor=""*.

Note 2: if the client sends a request without a cursor query parameter, the server must interpret it as if the client wants to retrieve the results from the very beginning. A client may send the query parameter “limit" without any cursor. In that case, the server must return at max the specified number of result items from the beginning.

Pagination requires a defined and consistent sorting. The server implementation must ensure a deterministic ordering of the result set. For instance, a server must not return an element A before another element C and in any later request return C before A. This applies in particular if any attribute of either A or C has been changed between the two requests. However, in case a new element B was created (or deleted), the client must expect that B and then C are returned after A.

Nevertheless, the inherent order of the result set must stay the same. Implementations may maintain an internal sorting attribute to ensure this behavior or implement it in any other appropriate manner. The server is not obligated to inform the client about its ordering schema.

The server informs the client about pagination attributes through the Result object in the request response. In particular, the Result contains the cursor value for the next page. Additional information, e.g. the overall number of result items, may also be part of it.

| **Class Name** | Result | | |
| --- | --- | --- | --- |
| **Explanation** | An object connecting the actual list of returned items with metadata information to, e.g. fetch the next part of the result set. | | |
| **Inherits from** | -- | | |
| **Attribute** | **Explanation** | **Type** | **Card.** | |
| result | List of returned items. Any kind of Referables is possible, depending on the endpoint which has been requested. | Referable | 0..\* | |
| paging\_metadata | Additional information for the client to, e.g. fetch the next part of the result set. | PagingMetadata | 1 | |

| **Class Name** | PagingMetadata | | |
| --- | --- | --- | --- |
| **Explanation** | Additional information for the client to, e.g. fetch the next part of the result set.  Note: more attributes may be added to this class in future versions. | | |
| **Inherits from** | -- | | |
| **Attribute** | **Explanation** | **Type** | **Card.** | |
| cursor | The cursor for the next part of the result set. No cursor attribute means that the end of the result set has been reached. | string | 0..1 | |

## Payload

The payload is generated from the technology-neutral specification as described in Part 1 of the Asset Administration Shell Series for JSON [1].

The serialization of JSON values is described in Clause 11.4.2.

Additional classes needed for payload of the HTTP/REST API specification can be found in Clause 10.2.

## Modifier Constraints

To use metadata objects as described in Clause 12.5., modifiers are implemented as HTTP query parameters or path suffixes. For example, a request for a specific submodel may look like:   
GET /submodel/$value?level=deep&extent=withBlobValue

The following constraints apply for the combination of modifiers:

* If Level=Core and Content=Value, only the requested object and the direct children without their value (empty value) will be returned in value serialization. If a direct child is a SubmodelElementCollection, "<SubmodelElementCollection/idShort>": {} will be returned. If a direct child is a SubmodelElementList, "<SubmodelElementList/idShort>": [] will be returned.
* The combination of Content=Metadata and Extent=WithBLOBValue is not allowed.
* The combination of Level=Deep and Content=Reference is not allowed.
* Modifiers cannot be used for POST operations.

In addition, the modifiers can also be used for PUT operations. They define how the request content is delivered and have the same semantics as in the related GET operation. Only Content=Reference and Content=Path are not possible for PUT.

## Mapping of Operations

The following

Table 15 shows the mapping of the generic operations to the HTTP/REST API.

The black entries correspond to the corresponding generic operations.

The blue entries are operations which only exist in the HTTP/REST API.

Table 15 Mapping of the generic Interface Operations to HTTP API Operations

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation Name** | **HTTP Verb** | **REST-Path** | **Comment (e.g. optional query parameters)** |
|  |  |  |  |
| **Asset Administration Shell Interface** |  |  |  |
| GetAssetAdministrationShell | GET | /aas | content-suffix: $reference |
| PutAssetAdministrationShell | PUT |  |  |
| GetAllSubmodelReferences | GET | /aas/submodel-refs | Pagination |
| PostSubmodelReference | POST | /aas/submodel-refs |  |
| DeleteSubmodelReference | DELETE | /aas/submodel-refs/{submodelIdentifier} | use base64url-encoded identifier |
| GetAssetInformation | GET | /aas/asset-information |  |
| PutAssetInformation | PUT | /aas/asset-information |  |
| GetThumbnail | GET | /aas/asset-information/thumbnail |  |
| PutThumbnail | PUT | /aas/asset-information/thumbnail |  |
| DeleteThumbnail | DELETE | /aas/asset-information/thumbnail |  |
|  | \* | /aas/submodels/{submodel-identifier}/\* | superpath as defined in service specification or profile |
|  |  |  |  |
| **Submodel Interface** |  |  |  |
| GetSubmodel | GET | /submodel | ?level=deep/core  path-suffix= $metadata/$value/$reference/$path or no suffix for normal  ?extent=WithoutBLOBValue/WithBLOBValue |
| PutSubmodel | PUT | /submodel |  |
| PatchSubmodel | PATCH | /submodel | path-suffix=$metadata/$value or no path for normal |
| GetAllSubmodelElements | GET | /submodel/submodel-elements | ?level=deep/core  path-suffix= $metadata/$value/$reference/$path or no suffix for nomal  ?extent=WithoutBLOBValue/WithBLOBValue  Pagination |
| GetSubmodelElementByPath | GET | /submodel/submodel-elements/{idShortPath} | use separated idshort path of this element  ?level=deep/core  path-suffix= $metadata/$value/$reference/$path or no suffix for nomal ?extent=WithoutBLOBValue/WithBLOBValue  URL-encoded IdShortPath |
| GetFileByPath | GET | /submodel/submodel-elements/{idShortPath}/attachment | use separated idShort path of this element  URL-encoded IdShortPath |
| PutFileByPath | PUT | /submodel/submodel-elements/{idShortPath}/attachment | use separated idShort path of this element  URL-encoded IdShortPath |
| DeleteFileByPath | DELETE | /submodel/submodel-elements/{idShortPath}/attachment | use separated idShort path of this element  URL-encoded IdShortPath |
| PostSubmodelElement | POST | /submodel/submodel-elements | SerializationModifiers are not used with POST |
| PostSubmodelElementByPath | POST | /submodel/submodel-elements/{idShortPath} | use separated idShort path of the parent element  SerializationModifiers are not used with POST |
| PutSubmodelElementByPath | PUT | /submodel/submodel-elements/{idShortPath} | use separated idShort path of this element  URL-encoded IdShortPath |
| PatchSubmodelElementByPath | PATCH | /submodel/submodel-elements/{idShortPath} | use separated idShort path of this element  path-suffix=$metadata/$value or no suffix for normal  URL-encoded IdShortPath  Note: values remain unchanged with content=metadata |
| PatchSubmodelElementValueByPath | PATCH | /submodel/submodel-elements/{idShortPath}/$value | use separated idShort path of this element; see Clause 11.4.2 for values  path-suffix=$value  URL-encoded IdShortPath |
| DeleteSubmodelElementByPath | DELETE | /submodel/submodel-elements/{idShortPath} | use separated idshort path of this element  URL-encoded IdShortPath |
| InvokeOperationSync | POST | /submodel/submodel-elements/{idShortPath}/invoke | path-suffix=$value or no suffix for normal  URL-encoded IdShortPath |
| InvokeOperationAsync | POST | /submodel/submodel-elements/{idShortPath} /invoke-async | get operationHandle  path-suffix=$value or no suffix for normal  URL-encoded IdShortPath |
| GetOperationAsyncResult | GET | /submodel/submodel-elements/{idShortPath} /operation-results/ {handleId} | handleId=operationHandle  path-suffix=$value or no suffix for normal  URL-encoded IdShortPath |
|  |  |  |  |
| **Shell Repository Interface** |  |  |  |
| GetAllAssetAdministrationShells | GET | /shells | path-suffix=$reference or no suffix normal  Pagination |
| GetAllAssetAdministrationShellsByAssetId | GET | /shells | base64url-encoded JSON-serialized key-value-pairs  ?assetids=…  Pagination |
| GetAllAssetAdministrationShellsByIdShort | GET | /shells | Pagination  ?idShort=<idShort to query for> |
| GetAssetAdministrationShellById | GET | /shells/{aasIdentifier} | base64url-encoded identifier  path-suffix=$reference or no suffix normal |
| PostAssetAdministrationShell | POST | /shells |  |
| PutAssetAdministrationShellById | PUT | /shells/{aasIdentifier} | base64url-encoded identifier |
| DeleteAssetAdministrationShellById | DELETE | /shells/{aasIdentifier} | base64url-encoded identifier |
| AasInterface | \* | /shells/{aasIdentifier}/\* | superpath as defined in Service Specification or Profile |
|  |  |  |  |
| **Submodel Repository Interface** |  |  |  |
| GetAllSubmodels | GET | /submodels | path-suffix= $metadata/$value/$reference/$path or no suffix for normal  Pagination |
| GetAllSubmodelsBySemanticId | GET | /submodels | ?semanticId=<base64url-encoded value of the semanticId>  path-suffix= $metadata/$value/$reference/$path or no suffix for normal  Constraint AASa-002: The base64url-encoded identifier of the semanticId shall have a length of maximum 3072 characters.  Pagination |
| GetAllSubmodelsByIdShort | GET | /submodels | path-suffix= $metadata/$value/$reference/$path or no suffix for normal  Pagination |
| GetSubmodelById | GET | /submodels/{submodelIdentifier} | path-suffix=$metadata or no suffix for normal  base64url-encoded identifier |
| PostSubmodel | POST | /submodels |  |
| PutSubmodelById | PUT | /submodels/{submodelIdentifier} | base64url-encoded identifier |
| PatchSubmodelById | PATCH | /submodels/{submodelIdentifier} | path-suffix=$metadata/$value or no suffix for normal |
| DeleteSubmodelById | DELETE | /submodels/{submodelIdentifier} | base64url-encoded identifier |
| SubmodelInterface | \* | /submodels/{submodelIdentifier}/\* | superpath as defined in service specification or profile |
|  |  |  |  |
| **Concept Description Repository Interface** |  |  |  |
| GetAllConceptDescriptions | GET | /concept-descriptions | Pagination |
| GetConceptDescriptionById | GET | /concept-descriptions/{cdIdentifier} | base64url-encoded identifier  Pagination |
| GetAllConceptDescriptionsByIdShort | GET | /concept-descriptions | Pagination |
| GetAllConceptDescriptionsByIsCaseOf | GET | /concept-descriptions | base64url-encoded identifier  Pagination |
| GetAllConceptDescriptionsByDataSpecificationReference | GET | /concept-descriptions | base64url-encoded identifier  Pagination |
| PostConceptDescription | POST | /concept-descriptions/ |  |
| PutConceptDescriptionById | PUT | /concept-descriptions/{cdIdentifier} | base64url-encoded identifier |
| DeleteConceptDescriptionById | DELETE | /concept-descriptions/{cdIdentifier} | base64url-encoded identifier |
|  |  |  |  |
| **AASX File Server Interface** |  |  |  |
| GetAllAASXPackageIds | GET | /packages | base64url-encoded identifier  Pagination |
| PostAASXPackage | POST | /packages |  |
| GetAASXByPackageId | GET | /packages/{packageId} | base64url-encoded identifier |
| PutAASXByPackageId | PUT | /packages/{packageId} | base64url-encoded identifier |
| DeleteAASXByPackageId | DELETE | /packages/{packageId} | base64url-encoded identifier |
|  |  |  |  |
| **Serialization Interface** |  |  |  |
| GenerateSerializationByIds | GET | /serialization | base64url-encoded identifier; AcceptHeader: application/aasx+xml or application/json oder application/xml |
|  |  |  |  |
| **AAS Basic Discovery Interface** |  |  |  |
| GetAllAssetAdministrationShellIdsByAssetLink | GET | /lookup/shells | base64url-encoded JSON-serialized key-value-pairs  ?assetids=…  Pagination |
| GetAllAssetLinksById | GET | /lookup/shells/{aasIdentifier} | base64url-encoded identifier |
| PostAllAssetLinksById | POST | /lookup/shells/{aasIdentifier} | base64url-encoded identifier |
| DeleteAllAssetLinksById | DELETE | /lookup/shells/{aasIdentifier} | base64url-encoded identifier |
|  |  |  |  |
| **AAS Registry Interface** |  |  |  |
| GetAllAssetAdministrationShellDescriptors | GET | /shell-descriptors | Pagination  assetKind=type|instance  assetType= base64url-encoded identifier |
| GetAssetAdministrationShellDescriptorById | GET | /shell-descriptors/{aasIdentifier} | base64url-encoded identifier |
| PostAssetAdministrationShellDescriptorById | POST | /shell-descriptors/{aasIdentifier} | base64url-encoded identifier |
| PutAssetAdministrationShellDescriptorById | PUT | /shell-descriptors/{aasIdentifier} | base64url-encoded identifier |
| DeleteAssetAdministrationShellDescriptorById | DELETE | /shell-descriptors/{aasIdentifier} | base64url-encoded identifier |
| Submodel Registry Interface | \* | /shell-descriptors/{aasIdentifier}/submodelDescriptors/\* | superpath as defined in Service Specification or Profile |
|  |  |  |  |
| **Submodel Registry Interface** |  |  |  |
| GetAllSubmodelDescriptors | GET | /submodel-descriptors | Pagination |
| GetSubmodelDescriptorById | GET | /submodel-descriptors/{submodelIdentifier} | base64url-encoded identifier |
| PostSubmodelDescriptor | POST | /submodel-descriptors/{submodelIdentifier} | base64url-encoded identifier |
| PutSubmodelDescriptorById | PUT | /submodel-descriptors/{submodelIdentifier} | base64url-encoded identifier |
| DeleteSubmodelDescriptorById | DELETE | /submodel-descriptors/{submodelIdentifier} | base64url-encoded identifier |
|  |  |  |  |
| **Descriptor Interface** |  |  |  |
| GetDescription | GET | /description | Provide additional information on interface endpoint; may also be used at a server endpoint to list all descriptions available on that server |

### Asynchronous Invocation of the SubmodelElement “Operation”

The invocation of the SubmodelElement “Operation” is the only call that can appear either synchronously or asynchronously in the current version of the specification. The expected behavior is therefore explained in detail.

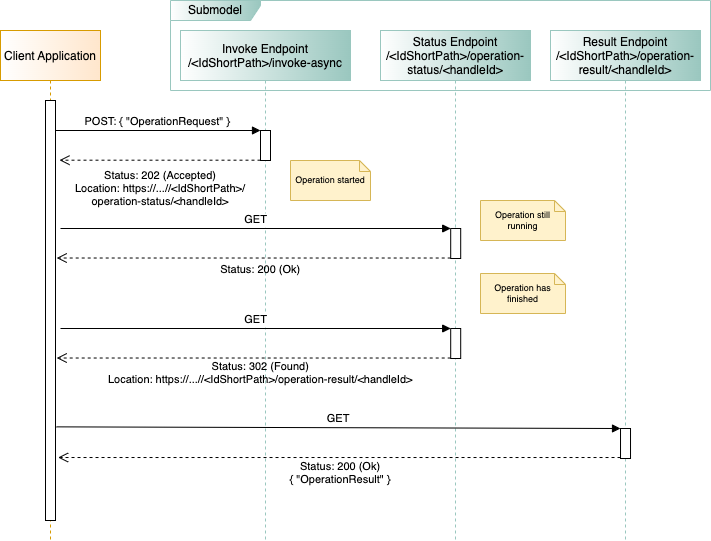


Figure 6 Sequence for asynchronous invocations of the SubmodelElement 'Operation'

The client informs the server whether it is interested in a synchronous (asynchronous) call by targeting the /invoke (/invoke-async) endpoint. In case of a synchronous interaction, the communication channel is kept open until the server has processed the request and responds with an OperationResult object, or a timeout or other kind of error occurs.

In the asynchronous pattern, the server immediately responds with an Accepted (status code: 202) message containing the link to an endpoint where the client can fetch status information about his request (see Figure 6). This status endpoint is also located at the same SubmodelElement “Operation”, followed by the path segments "/operation-status/{handleId}”.

In case the request is incorrect and the server already recognizes it, the server responds directly with the according status code, e.g. 400. If the server can only recognize the error during later processing and not at the time it receives the request, it responds with an Accepted (202) message at first. Hence, a received Accepted message does not guarantee the client that its request is valid in every case.

If the server has not finished processing the request, the status endpoint responds with an BaseOperationResult object with the attribute “executionState” set to “Running”. As soon as the processing is finished, the status endpoints deliver a Found (HTTP status code 302) response with the location of the result in the Location response header. The result is, similar to the status information, provided at the same SubmodelElement “Operation”, followed by the path segments "/operation-result/{handleId}”.

In case incorrect inputs have been provided by the client but the server was only able to recognize this during processing, or if the server perceived any other error during processing, the server must still provide the OperationResult object with status code 200 and set the attribute “executionState” to “Failed”.

Note: the invocation of the SubmodelElement “Operation” may also be conducted in the “ValueOnly” content. In this case, the "/$value” path segment is added to the previously mentioned endpoints.

## Mapping of Status Codes

The following table shows the mapping of the generic status codes to HTTP status codes according to IETF RFC 7231 (see Clause 6.1 <https://datatracker.ietf.org/doc/html/rfc7231#section-6>)

Table 16 Status Code Mapping for HTTP

|  |  |  |  |
| --- | --- | --- | --- |
| **Generic status code** | **Meaning** | **HTTP status code** | **Explanation** |
| Success | Success | 200 (OK) | Standard response for successful requests |
| SuccessCreated | Successful creation of a new resource | 201 (Created) | Successful request resulting in the creation of a new resource, e.g. SubmodelElement |
| SuccessAccepted | The reception of the request was successful | 202 (Accepted) | The server has accepted the request, but the result will be supplied later |
| SuccessNoContent | Success with explicitly no content in the payload | 204 (No Content) | Successful request with no content in return, e.g. used for updating existing resources |
| ClientErrorBadRequest | Bad or malformed request | 400 (Bad Request) | The server does not / cannot process the request due to a general client error, e.g. a malformed request |
| ClientNotAuthorized | Wrong or missing authorization credentials | 401 (Unauthorized) | The client missed or provided invalid credentials |
| ClientForbidden | Authorization has been refused | 403 (Forbidden) | The request content is basically valid and understood by the server, but the server refuses the action due to certain restrictions, e.g. profiles or roles |
| ClientErrorResourceNotFound | Resource not found | 404 (Not Found) | The requested resource was not found |
| ClientMethodNotAllowed | Operation request is not allowed | 405 (Method Not Allowed) | The server rejected the request for the requested resource, e.g. /invoke only for the operation submodel element |
| ClientResourceConflict | Conflict-creating resource (resource already exists) | 409 (Conflict) | A resource already exists; might occur if a Submodel or SubmodelElement with the same Identifier or ShortId is contained in a POST request. |
| ServerInternalError | Unexpected error | 500 (Internal Server Error) | General server-internal error due to an unexpected condition |
| ServerNotImplemented | Not implemented | 501 (Not Implemented) | The server does not support the functionality to fulfill the request |
| ServerErrorBadGateway | Bad Gateway | 502 (Bad Gateway) | The primarily addressed server that was acting as gateway or proxy received an invalid response from subsequent systems/servers |

## Additional Data Types for Payload for HTTP/REST

In addition to the data types used in the technology-neutral specification, the HTTP/REST API uses the data types as defined in this clause.

### PackageDescription

| **Class Name** | PackageDescription | | |
| --- | --- | --- | --- |
| **Explanation** | The package description consists of a system-wide unique packageId and its corresponding Asset Administration Shell identifiers. The packageId is used to identify the AASX package at the AASX file server. The package description is used to list the Asset Administration Shells in a given AASX package. This class is not part of the metamodel. | | |
| **Inherits from** | -- | | |
| **Attribute** | **Explanation** | **Type** | **Card.** | |
| packageId | File server specific package id | ShortIdType | 1 | |
| aasId | Asset Administration Shell unique identifier | Identifier | 0..\* | |

## Service Specifications and Profiles



Figure 2 defines that a service specification contains at least one API and that an API contains at least one API Operation.

The profiles defined in this clause present complete service specifications and their subsets.

For instance, the profile “RepositoryServiceSpecification/V3.0\_SSP-002” contains the API Operation “GetAllSubmodels” but not “PostSubmodelElementByPath”, while the more comprehensive “RepositoryServiceSpecification/V3.0\_SSP-001” contains both. Furthermore, profiles also define which of the SerializationModifiers (content, extent, level) or serialization formats (JSON) can be used or whether pagination or asynchronous operations are available.

Table 17 Overview of Service Specifications and the Contained APIs

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Contained APIs:**  **Service Specifications:** | Asset Administration Shell API | Submodel API | AASX File Server API | Asset Administration Shell Registry API | Submodel Registry API | Asset Administration Shell Repository API | Submodel Repository API | Concept Description Repository API | Asset Administration Shell Basic Discovery API | Serialization API | Description API |
| Asset Administration Shell Service Specification | x | s |  |  |  |  |  |  |  | x | x |
| Submodel Service Specification |  | x |  |  |  |  |  |  |  | x | x |
| AASX File Server Service Specification |  |  | x |  |  |  |  |  |  |  | x |
| Asset Administration Shell Registry Serv. Spec. |  |  |  | x | s |  |  |  |  |  | x |
| Submodel Registry Service Specification |  |  |  |  | x |  |  |  |  |  | x |
| Discovery Service Specification |  |  |  |  |  |  |  |  | x |  | x |
| Asset Administration Shell Repository Serv. Spec. | s | s |  |  |  | x | s |  |  | x | x |
| Submodel Repository Service Specification |  | s |  |  |  |  | x |  |  | x | x |
| ConceptDescription Repository Service Spec. |  |  |  |  |  |  |  | x |  | x | x |

x: Service Specification contains API at the root

s: Service Specification contains API through superpaths as introduced in Clause 0

### Profiles

Service specifications are further refined in profiles, governing which API operations, modifiers, and path combinations are supported. The following clauses describe each service specification and present their predefined profiles. Each profile is unambiguously identified and represented through a normative OpenAPI document. The different OpenAPI profiles of one ServiceSpecification share the same *title* attribute but with different *versions*. The version attribute contains both the major and minor version as well as the profile identifier. A profile identifier is defined as:

https://admin-shell.io/aas/API/3/0/<service specification name>/SSP-<profile number>

The name of the service specification ends with "ServiceSpecification".

The supported service specification or profile can be discovered at the /description endpoint. This endpoint will return the related profile string.

Additional profiles might be introduced in future versions of this document.

Note: in the following, only the last part (<name of service specification>/SSP-<profile number>) is used in the text for better readability, e.g. “AssetAdministrationShellServiceSpecification/SSP-001” instead of “https://admin-shell.io/aas/API/3/0/AssetAdministrationShellServiceSpecification/SSP-001”.

### Asset Administration Shell Service Specification

|  |  |
| --- | --- |
| **Service Specification / Profiles** | **Description** |
| AssetAdministrationShellServiceSpecification/ SSP-001 | Full feature set |
| AssetAdministrationShellServiceSpecification/ SSP-002 | Only read operations; is included in the profile AssetAdministrationShellServiceSpecification/SSP-001. |

#### Asset Administration Shell Service Specification – Full Profile

The Asset Administration Shell service specification with all its features and endpoints is represented through the profile identifier **AssetAdministrationShellServiceSpecification/SSP-001**:

|  |  |
| --- | --- |
| **Name:** | AAS Full Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/AssetAdministrationShellServiceSpecification/SSP-001 |
| **Feature** | Appearance |
| APIs and API Operations | *Asset Administration Shell API:* GetAssetAdministrationShell PutAssetAdministrationShell GetAllSubmodelReferences PostSubmodelReference DeleteSubmodelReference GetAssetInformation PutAssetInformation GetThumbnail PutThumbnail DeleteThumbnail  *Submodel API as superpath:* GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath PutFileByPath DeleteFileByPath PutSubmodel PatchSubmodel PostSubmodelElement PostSubmodelElementByPath PutSubmodelElementByPath PatchSubmodelElementByPath  DeleteSubmodelElementByPath InvokeOperationSync InvokeOperationAsync  GetOperationAsyncStatus GetOperationAsyncResult  *Serialization API:* GenerateSerializationByIds  *Description API:* GetDescription |
| SerializationModifier | Level: Core, Deep  Content: Normal, Metadata, Value, Reference, Path  Extent: WithBLOBValue, WithoutBLOBValue |
| SerializationFormat | JSON |
| Pagination | supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellServiceSpecification/V3.0_SSP-001>

#### Asset Administration Shell Service Specification – Read Profile

The Asset Administration Shell Service specification with the minimal feature set is represented through **AssetAdministrationShellServiceSpecification/SSP-002**:

|  |  |
| --- | --- |
| **Name:** | AAS Read Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/AssetAdministrationShellServiceSpecification/V3.0\_SSP-002 |
| **Feature** | Appearance |
| API Operations | *Asset Administration Shell API:* GetAssetAdministrationShell  GetAllSubmodelReferences  GetAssetInformation GetThumbnail  *Submodel API as superpath:* GetSubmodel  GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath  *Description API:* GetDescription |
| SerializationModifier | Level: Core, Deep  Content: Normal, Metadata, Value, Reference, Path  Extent: WithBLOBValue, WithoutBLOBValue |
| SerializationFormat | JSON |
| Pagination | supported |

See: [https://app.swaggerhub.com/apis/Plattform\_i40/AssetAdministrationShellServiceSpecification/V3.0\_SSP-002](https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellServiceSpecification/V3.0_SPP-002)

### Submodel Service Specification

|  |  |
| --- | --- |
| **Service Specification / Profiles** | Description |
| SubmodelServiceSpecification/SSP-001 | Full feature set |
| SubmodelServiceSpecification/SSP-002 | Only reads operations; is included in the profile SubmodelServiceSpecification/SSP-001. |
| SubmodelServiceSpecification/SSP-003 | Limitation on the basic capabilities plus the option to execute synchronous operations and to read the submodel in the ValueOnly-serialization format to reduce required bandwidth; is included in the profile SubmodelServiceSpecification/SSP-001. |

#### Submodel Service Specification – Full Profile

The submodel service specification with all its features and endpoints is represented through **SubmodelServiceSpecification/SSP-001**:

|  |  |
| --- | --- |
| **Name:** | Submodel Full Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/SubmodelServiceSpecification/SSP-001 |
| **Feature** | Appearance |
| APIs and API Operations | *Submodel API:* GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath PutFileByPath DeleteFileByPath PutSubmodel PatchSubmodel PostSubmodelElement PostSubmodelElementByPath PutSubmodelElementByPath PatchSubmodelElementByPath  DeleteSubmodelElementByPath InvokeOperationSync InvokeOperationAsync  GetOperationAsyncStatus GetOperationAsyncResult  *Serialization API:* GenerateSerializationByIds  *Description API:* GetDescription |
| SerializationModifier | Level: Core, Deep  Content: Normal, Metadata, Value, Reference, Path  Extent: WithBLOBValue, WithoutBLOBValue |
| SerializationFormat | JSON |
| Pagination | supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/SubmodelServiceSpecification/V3.0_SSP-001>

#### Submodel Service Specification – Read Profile

The submodel service specification with its minimal feature set is represented through **SubmodelServiceSpecification/SSP-002**:

|  |  |
| --- | --- |
| **Name:** | Submodel Read Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/AssetAdministrationShellServiceSpecification/SSP-002 |
| **Feature** | Appearance |
| API and API Operations | *Submodel API:* GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath  *Serialization API:* GenerateSerializationByIds  *Description API:* GetDescription |
| SerializationModifier | Level: Core, Deep  Content: Normal, Metadata, Value, Reference, Path  Extent: WithBLOBValue, WithoutBLOBValue |
| SerializationFormat | JSON |
| Pagination | supported |

See: [https://app.swaggerhub.com/apis/Plattform\_i40/SubmodelServiceSpecification/V3.0\_SSP-002](https://app.swaggerhub.com/apis/Plattform_i40/SubmodelServiceSpecification/V3.0_SPP-002)

#### Submodel Service Specification – Value Profile

The submodel service specification with a reduced feature set is represented through **SubmodelServiceSpecification/SSP-003**:

|  |  |
| --- | --- |
| **Name:** | Submodel Value Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/SubmodelServiceSpecification/SSP-003 |
| **Feature** | Appearance |
| APIs and API Operations | *Submodel API:* GetSubmodel  InvokeOperationSync  *Description API:* GetDescription |
| SerializationModifier | Level: Deep  Content: Normal, Value  Extent: WithBLOBValue, WithoutBLOBValue |
| SerializationFormat | JSON |
| Pagination | not supported |

See: [https://app.swaggerhub.com/apis/Plattform\_i40/SubmodelServiceSpecification/V3.0\_SSP-003](https://app.swaggerhub.com/apis/Plattform_i40/SubmodelServiceSpecification/V3.0_SPP-003)

### AASX File Server Service Specification

|  |  |
| --- | --- |
| **Service Specification / Profiles** | **Description** |
| AasxFileServerServiceSpecification/SSP-001 | The full feature set of the AASX File Server Service Specification |

#### AASX File Server Service Specification – Full Profile

|  |  |
| --- | --- |
| **Name:** | AASX File Server Full Profile |
| **Profile Identifier** | https://admin-shell.io /aas/API/3/0/AasxFileServerServiceSpecification/SSP-001 |
| **Feature** | Appearance |
| APIs and API Operations | *File Server API:* GetAllAASXPackageIds GetAASXByPackageId PostAASXPackage PutAASXByPackageId DeleteAASXByPackageId  *Description API:* GetDescription |
| SerializationModifier | not applicable |
| SerializationFormat | JSON for descriptions and error messages  AASX for packages |
| Pagination | supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/AasxFileServerServiceSpecification/V3.0_SSP-001>

### Asset Administration Shell Registry Service Specification

|  |  |
| --- | --- |
| **Service Specification / Profiles** | **Description** |
| AssetAdministrationShellRegistryServiceSpecification/SSP-001 | Full profile |
| AssetAdministrationShellRegistryServiceSpecification/SSP-002 | Only reads operations; is included in the profile AssetAdministrationShellRegistryServiceSpecification/SSP-001. |

#### Asset Administration Shell Registry Service Specification – Full Profile

|  |  |
| --- | --- |
| **Name:** | Asset Administration Shell Registry Full Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/ AssetAdministrationShellRegistryServiceSpecification/SSP-001 |
| **Feature** | Appearance |
| APIs and API Operations | *AAS Registry API:* GetAllAssetAdministrationShellDescriptors GetAssetAdministrationShellDescriptorById PostAssetAdministrationShellDescriptor PutAssetAdministrationShellDescriptorById DeleteAssetAdministrationShellDescriptorById  *Submodel Registry API as superpath:* GetAllSubmodelDescriptors GetSubmodelDescriptorById PostSubmodelDescriptor PutSubmodelDescriptorById DeleteSubmodelDescriptorById  *Description API:* GetDescription |
| SerializationModifier | not applicable |
| SerializationFormat | JSON |
| Pagination | Supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRegistryServiceSpecification/V3.0_SSP-001>

#### Asset Administration Shell Registry Service Specification – Read Profile

|  |  |
| --- | --- |
| **Name:** | AAS Registry Read Profile |
| **Profile Identifier:** | https://admin-shell.io/aas/API/3/0/ AssetAdministrationShellRegistryServiceSpecification/SSP-002 |
| **Feature** | Appearance |
| APIs and API Operations | *AAS Registry API:* GetAllAssetAdministrationShellDescriptors GetAssetAdministrationShellDescriptorById  *Submodel Registry API as superpath:* GetAllSubmodelDescriptors GetSubmodelDescriptorById  *Description API:* GetDescription |
| SerializationModifier | not applicable |
| SerializationFormat | JSON |
| Pagination | supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRegistryServiceSpecification/V3.0_SSP-002>

### Submodel Registry Service Specification

|  |  |
| --- | --- |
| **Service Specification / Profiles** | **Description** |
| SubmodelRegistryServiceSpecification/SSP-001 | Full profile |
| SubmodelRegistryServiceSpecification/SSP-002 | Only reads operations; is included in the profile SubmodelRegistryServiceSpecification/SSP-001. |

#### Submodel Registry Service Specification – Full Profile

|  |  |
| --- | --- |
| **Name:** | Submodel Registry Full Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/SubmodelRegistryServiceSpecification/SSP-001 |
| **Feature** | Appearance |
| APIs and API Operations | *Submodel Registry API:* GetAllSubmodelDescriptors GetSubmodelDescriptorById PostSubmodelDescriptor PutSubmodelDescriptorById DeleteSubmodelDescriptorById  *Description API:* GetDescription |
| SerializationModifier | not applicable |
| SerializationFormat | JSON |
| Pagination | supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/SubmodelRegistryServiceSpecification/V3.0_SSP-001>

#### Submodel Registry Profile – Read Profile

|  |  |
| --- | --- |
| **Name:** | Submodel Registry Read Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/ SubmodelRegistryServiceSpecification/SSP-002 |
| **Feature** | Appearance |
| APIs and API Operations | *Submodel Registry API:* GetAllSubmodelDescriptors GetSubmodelDescriptorById  *Description API:* GetDescription |
| SerializationModifier | not applicable |
| SerializationFormat | JSON |
| Pagination | Supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/SubmodelRegistryServiceSpecification/V3.0_SSP-002>

### Discovery Service Specification

|  |  |
| --- | --- |
| **Service Specification / Profiles** | **Description** |
| DiscoveryServiceSpecification/SSP-001 | Full feature set |

#### Discovery Service Specification – Full Profile

|  |  |
| --- | --- |
| **Name:** | Discovery Service Full Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/DiscoveryServiceSpecification/SSP-001 |
| **Feature** | Appearance |
| API and API Operations | *AAS Basic Discovery API:* GetAllAssetAdministrationShellIdsByAssetLink GetAllAssetLinksById PostAllAssetLinksById DeleteAllAssetLinksById  *Description API:* GetDescription |
| SerializationModifier | not applicable |
| SerializationFormat | JSON |
| Pagination | Not supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/DiscoveryServiceSpecification/V3.0_SSP-001>

### Asset Administration Shell Repository Service Specification

|  |  |
| --- | --- |
| **Service Specification / Profiles** | **Description** |
| AssetAdministrationShellRepository ServiceSpecification/SSP-001 | Full feature set |
| AssetAdministrationShellRepository ServiceSpecification/SSP-002 | Only read operations; is included in the profile AssetAdministrationShellRepositoryServiceSpecification/SSP-001 |

#### Asset Administration Shell Repository Service Specification – Full Profile

|  |  |
| --- | --- |
| **Name:** | AAS Repository Full Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/AssetAdministrationShellRepositoryServiceSpecification/SSP-001 |
| **Feature** | Appearance |
| API and API Operations | *AAS Repository API:*  GetAllAssetAdministrationShells GetAssetAdministrationShellById GetAllAssetAdministrationShellsByAssetId GetAllAssetAdministrationShellsByIdShort PostAssetAdministrationShell PutAssetAdministrationShellById DeleteAssetAdministrationShellById  *AAS API by superpath:* GetAssetAdministrationShell PutAssetAdministrationShell GetAllSubmodelReferences PostSubmodelReference DeleteSubmodelReference GetAssetInformation PutAssetInformation GetThumbnail PutThumbnail DeleteThumbnail  *Submodel Repository API by superpath:* GetAllSubmodels GetSubmodelById GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort PostSubmodel PutSubmodelById DeleteSubmodelById  *Submodel API by superpath:* GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath PutFileByPath DeleteFileByPath PutSubmodel PatchSubmodel PostSubmodelElement PostSubmodelElementByPath PutSubmodelElementByPath PatchSubmodelElementByPath  DeleteSubmodelElementByPath InvokeOperationSync InvokeOperationAsync  GetOperationAsyncStatus GetOperationAsyncResult  *AAS Serialization API:* GenerateSerializationByIds  *Description API:* GetDescription |
| SerializationModifier | Level: Core, Deep  Content: Normal, Metadata, Value, Reference, Path  Extent: WithBLOBValue, WithoutBLOBValue |
| SerializationFormat | JSON |
| Pagination | supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRepositoryServiceSpecification/V3.0_SSP-001>

#### Asset Administration Shell Repository Service Specification – Read Profile

|  |  |
| --- | --- |
| **Name:** | AAS Repository Read Profile |
| **Profile Identifier:** | https://admin-shell.io/aas/ API/3/0/AssetAdministrationShellRepositoryServiceSpecification/SSP-002 |
| **Feature** | Appearance |
| API and API Operations | *AAS Repository API:*  GetAllAssetAdministrationShells GetAssetAdministrationShellById GetAllAssetAdministrationShellsByAssetId GetAllAssetAdministrationShellsByIdShort  *AAS API by superpath:* GetAssetAdministrationShell  GetAllSubmodelReferences  GetAssetInformation  GetThumbnail  *Submodel Repository API by superpath:* GetAllSubmodels GetSubmodelById GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort  *Submodel API by superpath:* GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath  *Serialization API:* GenerateSerializationByIds  *Description API:* GetDescription |
| SerializationModifier | Level: Core, Deep  Content: Normal, Metadata, Value, Reference, Path  Extent: WithBLOBValue, WithoutBLOBValue |
| SerializationFormat | JSON |
| Pagination | supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRepositoryServiceSpecification/V3.0_SSP-002>

### Submodel Repository Service Specification

|  |  |
| --- | --- |
| **Service Specification / Profiles** | **Description** |
| SubmodelRepositoryServiceSpecification/ SSP-001 | Full feature set |
| SubmodelRepositoryServiceSpecification/ SSP-002 | Only read operations; is included in the profile SubmodelRepositoryServiceSpecification/SSP-001 |
| SubmodelRepositoryServiceSpecification/ SSP-003 | Profile for a Submodel Repository which only contains Submodels with kind=Template; is *not* included in the profile SubmodelRepositoryServiceSpecification/SSP-001 or the profile SubmodelRepositoryServiceSpecification/SSP-002 |
| SubmodelRepositoryServiceSpecification/ SSP-004 | Only read operations for a Submodel Repository which only contains Submodels with kind=Template; is included in the profile SubmodelRepositoryServiceSpecification/SSP-003 but *not* in the profile SubmodelRepositoryServiceSpecification/ SSP-001 or the profile SubmodelRepositoryService Specification/SSP-002 |

#### Submodel Repository - Full Profile

|  |  |
| --- | --- |
| **Name:** | Submodel Repository Full Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/SubmodelServiceSpecification/SSP-001 |
| **Feature** | Appearance |
| API and API Operations | *Submodel Repository API:* GetAllSubmodels GetSubmodelById GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort PostSubmodel  PutSubmodelById  PatchSubmodelById DeleteSubmodelById  *Submodel API by superpath:* GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath PutFileByPath DeleteFileByPath PutSubmodel PatchSubmodel PostSubmodelElement PostSubmodelElementByPath PutSubmodelElementByPath PatchSubmodelElementByPath  DeleteSubmodelElementByPath InvokeOperationSync InvokeOperationAsync  GetOperationAsyncStatus GetOperationAsyncResult  *AAS Serialization API:* GenerateSerializationByIds  *Description API:* GetDescription |
| SerializationModifier | Level: Core, Deep  Content: Normal, Metadata, Value, Reference, Path  Extent: WithBLOBValue, WithoutBLOBValue |
| SerializationFormat | JSON |
| Pagination | supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/SubmodelRepositoryServiceSpecification/V3.0_SSP-001>

#### Submodel Repository – Read Profile

|  |  |
| --- | --- |
| **Name:** | Submodel Repository Read Profile |
| **Profile Identifier** | https://admin-shell.io /aas/API/3/0/SubmodelServiceSpecification/SSP-002 |
| **Feature** | Appearance |
| API and API Operations | *Submodel Repository API:* GetAllSubmodels GetSubmodelById GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort  *Submodel API by superpath:* GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath  *Serialization API:* GenerateSerializationByIds  *Description API:* GetDescription |
| SerializationModifier | Level: Core, Deep  Content: Normal, Metadata, Value, Reference, Path  Extent: WithBLOBValue, WithoutBLOBValue |
| SerializationFormat | JSON |
| Pagination | supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/SubmodelRepositoryServiceSpecification/V3.0_SSP-002>

#### Submodel Repository - Template Profile

The Submodel Repository service specification that only provides and manages Submodel Templates is represented through the profile identifier **SubmodelRepositoryServiceSpecification/SSP-003**.

Constraint AASa-003: A service implementing the SubmodelServiceSpecification/SSP-003 must not accept or provide any Submodel with the attribute “kind=Instance”.

Note 1: due to Constraint AASa-003, SubmodelServiceSpecification/SSP-003 can not be combined with SubmodelServiceSpecification/SSP-001 or SubmodelServiceSpecification/SSP-002 as SubmodelService  
Specification/SSP-001 or SubmodelServiceSpecification/SSP-002-compliant services may contain Submodel instances but SubmodelServiceSpecification/SSP-003 not.

Note 2: future versions may introduce a Submodel Repository Instance Profile.

|  |  |
| --- | --- |
| **Name:** | Submodel Repository Template Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/SubmodelServiceSpecification/SSP-003 |
| **Feature** | Appearance |
| API and API Operations | *Submodel Repository API:* GetAllSubmodels GetSubmodelById GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort PostSubmodel PutSubmodelById  PatchSubmodelById DeleteSubmodelById  *Submodel API by superpath:* GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath PutFileByPath DeleteFileByPath PutSubmodel PatchSubmodel PostSubmodelElement PostSubmodelElementByPath PutSubmodelElementByPath PatchSubmodelElementByPath  DeleteSubmodelElementByPath  *AAS Serialization API:* GenerateSerializationByIds  *Description API:* GetDescription |
| SerializationModifier | Level: Core, Deep  Content: Normal, Metadata  Extent: WithoutBLOBValue |
| SerializationFormat | JSON |
| Pagination | supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/SubmodelRepositoryServiceSpecification/V3.0_SSP-003>

#### Submodel Repository - Template Read Profile

The Submodel Repository service specification that only provides Submodel Templates is represented through the profile identifier **SubmodelRepositoryServiceSpecification/SSP-004**.

Constraint AASa-004: A service implementing the SubmodelServiceSpecification/SSP-004 must not accept or provide any Submodel with the attribute “kind=Instance”.

Note: due to Constraint AASa-004, SubmodelServiceSpecification/SSP-004 can not be combined with SubmodelServiceSpecification/SSP-001 or SubmodelServiceSpecification/SSP-002 as SubmodelService  
Specification/SSP-001 or SubmodelServiceSpecification/SSP-002-compliant services may contain Submodel instances but SubmodelServiceSpecification/SSP-004 not.

|  |  |
| --- | --- |
| **Name:** | Submodel Repository Template Read Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/SubmodelServiceSpecification/SSP-004 |
| **Feature** | Appearance |
| API and API Operations | *Submodel Repository API:* GetAllSubmodels GetSubmodelById GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort  *Submodel API by superpath:* GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath  GetFileByPath  *Serialization API:* GenerateSerializationByIds  *Description API:* GetDescription |
| SerializationModifier | Level: Core, Deep  Content: Normal, Metadata  Extent: WithoutBLOBValue |
| SerializationFormat | JSON |
| Pagination | supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/SubmodelRepositoryServiceSpecification/V3.0_SSP-004>

### Concept Description Repository Service Specification

|  |  |
| --- | --- |
| **Service Specification / Profiles** | **Description** |
| ConceptDescriptionRepositoryServiceSpecification/SSP-001 | Full feature set |

#### Concept Description Repository Service Specification – Full Profile

|  |  |
| --- | --- |
| **Name:** | Concept Description Repository Full Profile |
| **Profile Identifier:** | https://admin-shell.io /aas/API/3/0/ConceptDescriptionRepositoryServiceSpecification/SSP-001 |
| **Feature** | Appearance |
| API and API Operations | *ConceptDescription Repository API* GetAllConceptDescriptions GetConceptDescriptionById GetAllConceptDescriptionsByIdShort GetAllConceptDescriptionsByIsCaseOf’ GetAllConceptDescriptionsByDataSpecificationReference PostConceptDescription PutConceptDescriptionById DeleteConceptDescriptionById  *Serialization API:* GenerateSerializationByIds  *Description API:* GetDescription |
| SerializationModifier | not applicable |
| SerializationFormat | JSON |
| Pagination | Supported |

See: <https://app.swaggerhub.com/apis/Plattform_i40/ConceptDescriptionRepositoryServiceSpecification/V3.0_SSP-001>

## Interactions

Interactions describe the sequence of calls of operations by a client application to achieve a defined goal in a use case. Future versions of the document will describe interactions for further use cases.

Currently, only the key use case “Access a submodel in a distributed system” with focus on a completely decentralized Industry 4.0 system is described.

Since the interaction diagram in the current version only describes a first subset of interactions, some constraints and assumptions are made according to the configuration and qualities of the system.

Constraints and assumptions for calling an AAS and a submodel operation by a client application:

* The calling application hast to be aware that endpoints may change at any time. If the application has cached an endpoint that is no longer vivid, the application needs to start the interaction to resolve the appropriate endpoint again from the beginning.
* Endpoints for infrastructure interfaces like AAS registries are known at design time of the client application or configured manually before start-up.
* The endpoint information of the submodel registry must be known to the client application. The following questions are subject to discussion for future interaction versions:
  1. Will it be accessible via the AAS interface and therefore become a mandatory part of a standard interaction?
  2. How much “control” of submodels is implemented in the AAS and how are distributed submodels handled, which are deployed in network areas not accessible by the AAS server application?
* The AAS server application itself is instantiated and registered by calling an AAS registry interface.
* The AAS-ID is known to the calling application.
* Access to any API is allowed only if authenticated (mechanisms for authentication are to be described separately) and response follows a defined access rights model for all calls.

In Figure 7 below, the interaction starts with a client application resolving the interface endpoint of an Asset Administration Shell registry with a known asset ID. The AAS registry provides the AAS Descriptor object belonging to this asset ID and containing the Submodel Descriptors of the Submodels, which are part of the related AAS. Both descriptor kinds, those of the Asset Administration Shell and those of the Submodels, contain endpoint information for the respective AAS and Submodel Repositories. AAS interface operations – independent of the underlying protocol – are used to retrieve the AAS from the AAS repository answering at one of the supplied endpoints. In the sequence shown in Figure 7, one submodel of this AAS is also provided through the same repository. For the second submodel, however, only the submodel identifier is provided by the AAS registry, while the endpoint information is empty. Therefore, the client application must execute another look-up at a dedicated Submodel Registry, which responds with one Submodel Descriptor object. Equipped with this information, the client application locates the stand-alone Submodel Repository and downloads the second submodel.

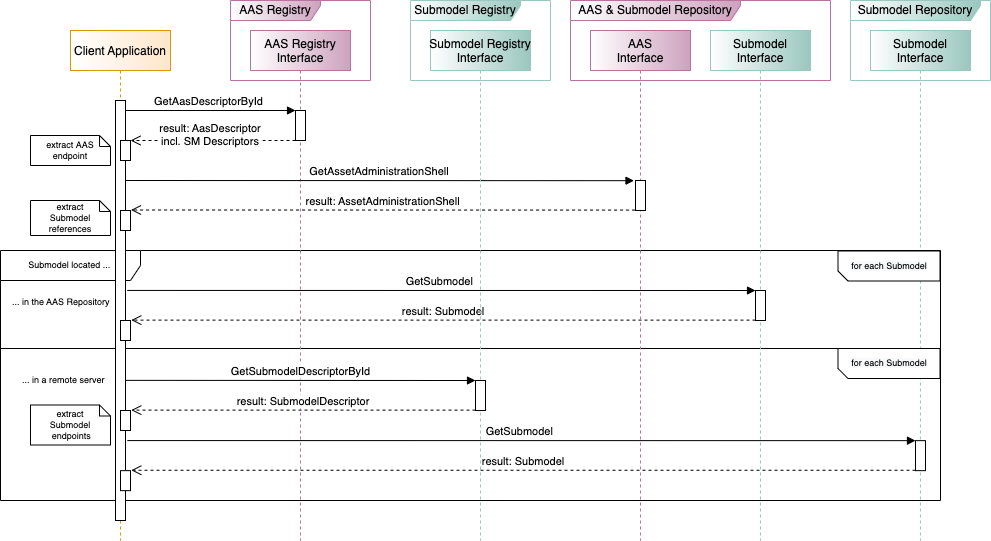


Figure 7 Interactions for Client Applications using AAS and Submodel Interfaces (independent Submodel Registry)

Figure 8 shows a slight variant of the scenario in Figure 7. The AAS Registry only returns the AAS Descriptor without any Submodel Descriptors. The client application retrieves the AAS from the discovered AAS Repository endpoint and learns about its submodel references. Using these references, the client can ask a Submodel Registry, in this case also hosted by the repository server, for the respective Submodel Descriptors. It understands that the Submodels are also available at the same server and downloads them by sending requests to the Submodel Interface.

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

Figure 8 Interaction for Client Application using AAS and Submodel Interfaces (included Submodel Registry)

The difference between Interface and API Operations is outlined in Figure 9. This sequence translates the interaction on the interface level of Figure 7, which is protocol-independent and therefore can be implemented in several different manners, to the specific HTTP API Operations. The generic operations are replaced with HTTP requests, e.g. “GetAasDescriptorById” with “GET /shell-descriptors/<aas-id>”. The returned objects are shortened for better readability. The first request to the Submodel API shows the concatenated path (“/shells/<aas-id>/aas” + “/submodels/<submodel-id>”) and illustrates how contained submodels can be provided natively through an AAS API.

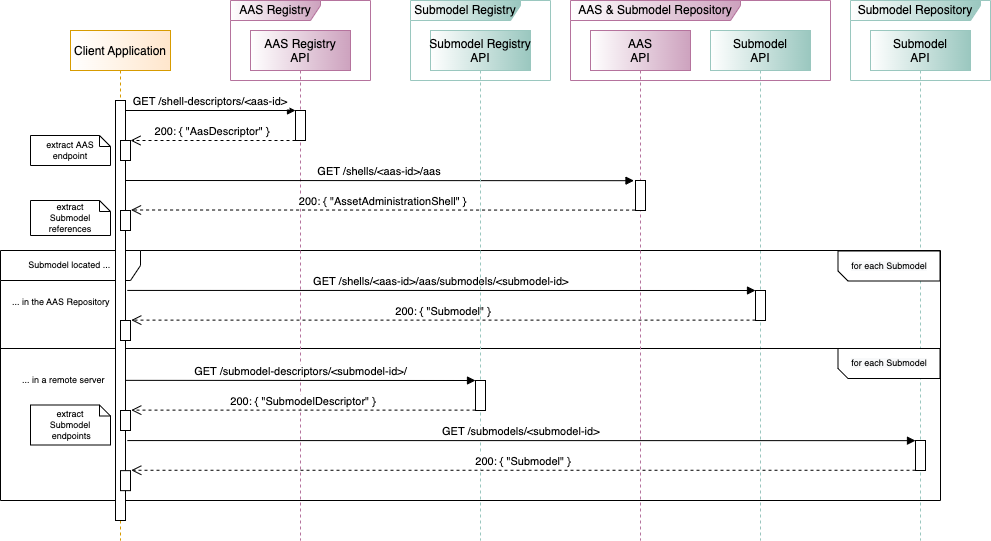


Figure 9 Interaction for Client Application using AAS and Submodels (for HTTP API Operations)

## Security

The AAS metamodel includes a security metamodel, which is defined in the chapter “Overview Metamodel of Administration Shell w.r.t. Security”. This chapter was part of “Details of the Asset Administration Shell Part 1” until version 3.0 RC02. Since Part 1 has been split for the release of version 3.0, this chapter will become the basis of “Specification of the Asset Administration Shell. Part 4: Security”. In addition to the security metamodel, Part 4 will define e.g. authentication, further details about authorization, and signature of data.

Authentication is mandatory. Depending on the ecosystem the AAS uses, different authentication mechanisms might be in place. This clause explains one exemplary authentication mechanisms, which has been developed by the security working group (AG3) of Plattform Industrie 4.0. Other authentication services (e.g. Username/Password, DID=Decentralized Identifiers, Verifiable Credentials, EDC=Eclipse Data Space Connector, or IDS=International Data Spaces) may also be used to receive an access token for authorization.

The following paragraphs describe the most important steps for token-based authentication of the HTTP/REST APIs. For more details, see “Secure Downloadservice” (<https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/secure_downloadservice.html>). Figure 10 gives an overview.

Ein Bild, das Diagramm enthält.

Automatisch generierte BeschreibungFigure 10 The private\_key\_certchain\_jwt Method [...download service]

A client application uses a client certificate to create a certificate chain. The certificate chain can be checked on the authentication server by the corresponding Root CA certificate, which is signed by a certification authority (CA). The client application sends the certificate chain to the authentication server as token request by a JSON Web Token (JWT). The JWT is signed by the client’s private key corresponding to the client certificate (JWT = Data + Signature).

If the authentication is approved, the client application receives an access token from the authentication server (not shown in Figure 10).

Such an access token contains attributes from the client certificate (e.g. username, email address) which will be sent as HTTP header bearer token to the AAS server application. The latter will check, whether the access token has been signed by a trusted authentication server and will make the authorization according to the AAS security metamodel.

A running demo is explained in “Secure Downloadservice”. A corresponding server can be seen on <https://admin-shell-io.com/5011/> with a related security AAS at the bottom.

The AAS security metamodel does not deal with authentication; it assumes that the user is already authenticated. The example security AAS is only created for demonstration purposes and is not standardized. Since the version of the AASX Package Explorer used does not yet support the AAS security metamodel, the required information in subsequent steps like the access permission rules for AAS are modelled as a submodel.

The different security and authentication steps are explained in the video <https://admin-shell-io.com/screencasts/security/Industrie_40_Security_with_AASX_Server.mp4>.

## API Code Generation

SwaggerHub includes the possibility to generate code from an API, e.g. for C# .NET:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

API code can be created for both servers and clients in various programming languages.

Known issues include the following:

* When generating the aspnetcore server stub for the AssetAdministrationShellEnvironmentApi, the operation PutFileByPath is not generated automatically and must be added manually.
* When generating the aspnetcore server stub for the AasxFileServerInterfaceApi, the operations PostAASXPackage and PutAASXPackageById are not generated automatically and must be added manually.

The SwaggerHub code generator development team is not part of the AAS activities and has been informed about these issues.

# Summary and Outlook

This document specifies the interfaces for a single Asset Administration Shell and its Submodels, as well as for a repository of Asset Administration Shells. Additionally, infrastructural interfaces like Registry and Discovery of a set of Asset Administration Shells are specified.

All interfaces are specified in a technology-neutral way before defining technology-specific APIs.

In this version of the specification, HTTP/REST APIs are defined and mapped to the technology-neutral specification as a frontrunner.

In subsequent versions of this specification, APIs using other technologies are planned to be supported, e.g. gRPC or MQTT.

Additionally, further interfaces, service specifications, and profiles may be defined. Querying will also be a topic.

Another very important topic that will be looked at in upcoming versions of the specification in more detail is the definition of access control based on the information provided by an Asset Administration Shell to ensure the trustworthiness of the contained information across different system boundaries.

1. Templates used for Specification

This Annex explains the table templates used for documentation of interfaces, operations, data types, etc.

*Card.* is the cardinality (or multiplicity) defining the lower and upper bound of the number of instances of the member element. "\*" denotes an arbitrary infinite number of elements of the corresponding Type. "0..1" means optional. "0..\*" or "0..3" etc. means that the list may be either not available (null object) or empty or has infinitely many / exactly three elements.

Note: attributes having a default value are always considered to be optional; there is always a value for the attribute because the default value is used for initialization in this case.

Table 18 Interface Description

|  |  |
| --- | --- |
| Interface: <Interface Name> | |
| **Operation Name** | **Description** |
| Oper1 | Human-understandable description of the operation of the interface. Only major input and output information shall be described, no individual request and result parameters.  Note: all words in the service operation name are written together in italics without a blank in between. The first letter of the first word is lower case, all other words are upper case. |
| … |  |
| operN (optional) | Human-understandable description of the operation n of the interface. Optional operations are to be marked by suffix (optional) after the operation name. |

Table 19 Operation Description

| **Operation Name:** | Name of the operation: all individual words in the operation name are capitalized | | | |
| --- | --- | --- | --- | --- |
| **Explanation:** | Human-understandable description of the functionality  The operation provides its functionality through the following input and output parameters:   Input parameter 1: human-understandable description of the purpose of the input parameter 1   …   Input parameter N: human-understandable description of the purpose of input parameter N   Output parameter 1: human-understandable description of the purpose of output parameter 1: human-understandable description of the purpose of the input parameter 1   …   Output Parameter N: human-understandable description of the purpose of output parameter N:  If ***payload*** is mentioned as output parameter, only the returned payload in case of a successful operation (status code: Success, SuccessCreated) is denoted in column *Type*. In case of failure see Clause 10.2.9.1.  If no ***payload*** is mentioned as output parameter, the status code shall be SuccessNoContent in case of success, otherwise see Clause 0.  Convention: all words in the interface name are written together in italics without a blank in between. The first letter of the first word and all other words are written in upper case letters. | | | |
| **semanticId** | The unique identifier of this operation | | | |
| **Name** | **Description** | **Mand.** | **Type** | **Card.** |
| Input Parameter | | | | |
| inputParameter1 | Human-understandable description of the input parameter 1 of the operation. | States whether the inputParameter1 is mandatory (“yes”) or optional (“no”) | Type of the input parameter 1 | The cardinality of type of the inputParameter1, e.g. zero-to-one (“0..1”) or at-least-one (“1..\*). |
| … |  |  |  |  |
| inputParameterN | Human-understandable description of the input parameter N of the operation. | States whether the inputParameterN is mandatory (“yes”) or optional (“no”) | Type of the input parameter N | The cardinality of type of the inputParameterN, e.g. zero-to-one (“0..1”) or at-least-one (“1..\*). |
| Output Parameter | | | | |
| outputParameter1 | Human-understandable description of the output parameter 1 of the operation. | States whether the outputParameter1 is mandatory (“yes”) or optional (“no”) | Type of the output  parameter 1 | The cardinality of type of the outputParameter1, e.g. zero-to-one (“0..1”) or at-least-one (“1..\*). |
| … |  |  |  |  |
| outputParameterN | Human-understandable description of the output parameter N of the operation. | States whether the outputParameterN is mandatory (“yes”) or optional (“no”) | Type of the output parameter N | The cardinality of type of the outputParameterN, e.g. zero-to-one (“0..1”) or at-least-one (“1..\*). |

Table 20 Data Types for Payload Description

| **Class Name** | Name of the class: all individual words in the class name are capitalized | | | |
| --- | --- | --- | --- | --- |
| **Explanation** | Human-understandable description of the class  The Class has following attributes:   Attribute 1: human-understandable description of the purpose of the attribute 1   …   Attribute N: human-understandable description of the purpose of the attribute N  Convention: all words in the class name are written together in italics without a blank in between. The first letter of the first word and all other words are written in upper case letters. | | | |
| **Inherits from** | Name of the class this class inherits from | | | |
| **semanticId** | The unique identifier of this class | | | |
| **Attribute (\* = mandatory)** | **Explanation** | **Type** | **Card.** |
| attribute1 | Human-understandable description of the attribute 1 of the class. | Type of the attribute 1 | Cardinality of the attribute 1 |
| … |  |  |  |
| attributeN | Human-understandable description of the attribute N of the class. | Type of the attribute N | Cardinality of the attribute N |

Table 21 Enumeration Description

|  |  |
| --- | --- |
| **Enumeration Name:** | Name of the enumeration: all individual words in the enumeration name are capitalized |
| **Explanation:** | Human-understandable description of the enumeration  The enumeration has following literals:   Literal 1: human-understandable description of the purpose of the literal 1   …   Literal N: human-understandable description of the purpose of the literal N  Convention: all words in the enumeration name are written together in italics without a blank in between. The first letter of the first word and all other words are written in upper case letters. |
| **semanticId** | The unique identifier of this enumeration |
| **Literal** | **Description** |
| Literal1 | Human-understandable description of the literal 1 of the enumeration. |
| … |  |
| LiteralN | Human understandable description of the literal N of the enumeration. |

1. ValueOnly-Serialization Example

The following example shows the ValueOnly-Serialization for an entire Submodel that validates against the JSON-schema specified in Clause 11.4.3.   
As mentioned in Clause 11.4.3, *SubmodelElementCollection*s cannot be validated within the same schema due to circularity reasons; instead they have their own specific validation schema. An exemplary *SubmodelElementCollection* is added to the following JSON for completeness. It is, however, not validatable against the schema in Clause 11.4.3 due to the reasons mentioned above.

|  |
| --- |
| {  "MyPropertyIdShortNumber": 5000,  "MyPropertyIdShortString": "MyTestStringValue",  "MyPropertyIdShortBoolean": **true**,  "MyMultiLanguageProperty": [  {  "de": "Das ist ein deutscher Bezeichner"  },  {  "en": "That's an English label"  }  ],  "MyRange": {  "min": 3,  "max": 15  },  "MyFile": {  "contentType": "application/pdf",  "value": "SafetyInstructions.pdf"  },  "MyBlob": {  "contentType": "application/octet-stream",  "value": "VGhpcyBpcyBteSBibG9i"  },  "MyEntity": {  "statements": {  "MaxRotationSpeed": 5000  },  "entityType": "SelfManagedEntity",  "globalAssetId": "http://customer.com/demo/asset/1/1/MySubAsset"  },  "MyReference": {  "type": "ModelReference",  "keys": [  {  "type": "Submodel",  "value": "http://customer.com/demo/aas/1/1/1234859590"  },  {  "type": "Property",  "value": "MaxRotationSpeed"  }  ]  },  "MyBasicEvent": {  "observed": {  "type": "ModelReference",  "keys": [  {  "type": "Submodel",  "value": "http://customer.com/demo/aas/1/1/1234859590"  },  {  "type": "Property",  "value": "CurrentValue"  }  ]  }  },  "MyRelationship": {  "first": {  "type": "ModelReference",  "keys": [  {  "type": "Submodel",  "value": "http://customer.com/demo/aas/1/1/1234859590"  },  {  "type": "Property",  "value": "PlusPole"  }  ]  },  "second": {  "type": "ModelReference",  "keys": [  {  "type": "Submodel",  "value": "http://customer.com/demo/aas/1/0/1234859123490"  },  {  "type": "Property",  "value": "MinusPole"  }  ]  }  },  "MyAnnotatedRelationship": {  "first": {  "type": "ModelReference",  "keys": [  {  "type": "Submodel",  "value": "http://customer.com/demo/aas/1/1/1234859590"  },  {  "type": "Property",  "value": "PlusPole"  }  ]  },  "second": {  "type": "ModelReference",  "keys": [  {  "type": "Submodel",  "value": "http://customer.com/demo/aas/1/0/1234859123490"  },  {  "type": "Property",  "value": "MinusPole"  }  ]  },  "annotation": [  {  "AppliedRule": "TechnicalCurrentFlowDirection"  }  ]  },  "MySubmodelElementIntegerPropertyList": [  1,  2,  30,  50  ],  "MySubmodelElementFileList": [  {  "contentType": "application/pdf",  "value": "MyFirstFile.pdf"  },  {  "contentType": "application/pdf",  "value": "MySecondFile.pdf"  }  ],  "MySubmodelElementCollection":  {  "myStringElement": "That’s a string",  "myIntegerElement": 5,  "myBooleanElement": true  }  } |

1. SerializationModifier Examples
   1. Description

SerializationModifiers are only allowed for GET and PATCH operations.

GET operations can use any combination of SerializationModifiers.

POST operations create new resources using the input content.

PUT operations replace existing resources using the input content.

POST and PUT use the regular serialization. The client creates the input content as needed, so that no further SerializationModifiers need to be used.

PATCH operations may use the regular serialization, the metadata serialization, or the ValueOnly- serialization. The SerializationModifier Core is not used. The resources in the input content must already exist on the server and are replaced one by one accordingly. If one of the resources in the input content does not exist, no changes will be made on the server. “Resource exists” means, that the type of a SubmodelElement is the same in the input content and on the server. For example, a property may only be replaced by a property; elements of a SubmodelElementCollection or SubmodelElementList can only be replaced if they already exist on the server. A SubmodelElementList with five elements cannot be patched with a SubmodelElementList with more than five elements. A SubmodelElementList with five elements can be patched with a SubmodelElementList with less than five elements since all required elements starting from index 0 already exist.

Note: values remain unchanged with content=metadata.

* 1. Examples for GET Operations

|  |  |  |
| --- | --- | --- |
|  | **Deep (default)** | **Core** |
| **Normal (default)** | If applied to the Submodel:  **{**  "modelType"**:** "Submodel"**,**  "id"**:** "http://i40.customer.com/type/1/1/7A7104BDAB57E184"**,**  "idShort"**:** "TechnicalData"**,**  "semanticId"**:** **{**  "keys"**:** **[ {**  "type"**:** "GlobalReference"**,**  "value"**:** "0173-1#01-AFZ615#016"  **} ],**  "type"**:** "ExternalReference"  **},**  "submodelElements"**:** **[ {**  "modelType"**:** "SubmodelElementCollection"**,**  "idShort"**:** "RotationSpeed"**,**  "semanticId"**: {**  "keys"**:** **[ {**  "type"**:** "GlobalReference"**,**  "value"**:** "http://purl.org/iot/vocab/iot-taxonomy-lite#RotationalSpeed"  **} ],**  "type"**:** "ExternalReference"  **},**  "value"**:** **[ {**  "modelType"**:** "Property"**,**  "idShort"**:** "MaxRotationSpeed"**,**  "category"**:** "PARAMETER"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "ConceptDescription",  "value"**:** "0173-1#02-BAA120#008"  **} ],**  "type"**:** "ExternalReference"  **},**  "valueType"**:** "xs:int"**,**  "value": "5000"  **} ]**  **} ]**  **}** | If applied to the Submodel:  {  "modelType": "Submodel",  "id"**:** "http://i40.customer.com/type/1/1/7A7104BDAB57E184"**,**  "idShort"**:** "TechnicalData"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "GlobalReference"**,**  "value"**:** "0173-1#01-AFZ615#016"  **} ],**  "type"**:** "ExternalReference"  **},**  "submodelElements"**: [ {**  "modelType"**:** "SubmodelElementCollection"**,**  "idShort"**:** "RotationSpeed"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "GlobalReference"**,**  "value"**:** "http://purl.org/iot/vocab/iot-taxonomy-lite#RotationalSpeed"  **} ],**  "type"**:** "ExternalReference"  **}**  **} ]**  **}** |
| **Metadata** | If applied to the Submodel:  {  "modelType"**:** "Submodel"**,**  "id"**:** "http://i40.customer.com/type/1/1/7A7104BDAB57E184"**,**  "idShort"**:** "TechnicalData"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "GlobalReference"**,**  "value"**:** "0173-1#01-AFZ615#016"  **} ],**  "type"**:** "ExternalReference"  **},**  "submodelElements"**: [ {**  "modelType"**:** "SubmodelElementCollection"**,**  "idShort"**:** "RotationSpeed"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "GlobalReference"**,**  "value"**:** "http://purl.org/iot/vocab/iot-taxonomy-lite#RotationalSpeed"  **} ],**  "type"**:** "ExternalReference"  **},**  "value"**: [ {**  "modelType"**:** "Property"**,**  "idShort"**:** "MaxRotationSpeed"**,**  "category"**:** "PARAMETER",  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "ConceptDescription"**,**  "value"**:** "0173-1#02-BAA120#008"  **} ],**  "type"**:** "ExternalReference"  **},**  "valueType"**:** "xs:int"  **} ]**  **} ]**  **}** | If applied to the Submodel:  {  "modelType"**:** "Submodel"**,**  "id"**:** "http://i40.customer.com/type/1/1/7A7104BDAB57E184"**,**  "idShort"**:** "TechnicalData"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "GlobalReference"**,**  "value"**:** "0173-1#01-AFZ615#016"  **} ],**  "type"**:** "ExternalReference"  **},**  "submodelElements"**: [ {**  "modelType"**:** "SubmodelElementCollection"**,**  "idShort"**:** "RotationSpeed"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "GlobalReference"**,**  "value"**:** "http://purl.org/iot/vocab/iot-taxonomy-lite#RotationalSpeed"  **} ],**  "type"**:** "ExternalReference"  **}**  **} ]**  **}** |
| If applied to the Property, i.e. idShortPath “RotationSpeed.MaxRotationSpeed”: {  "modelType"**:** "Property"**,**  "idShort": "DocumentId"**,**  "category"**:** "PARAMETER"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "GlobalReference",  "value"**:** "0173-1#02-BAA120#008"  **} ] ,**  "type"**:** "ExternalReference"  **},**  "valueType"**:** "xs:int"  **}** |
| **Value** | If applied to the Submodel:  {  "TechnicalData": **{**  "RotationSpeed": **{**  "MaxRotationSpeed"**:** "5000"  **}**  **}**  **}** | If applied to the Submodel:  {  "TechnicalData"**:** **{**  "RotationSpeed"**:** **{}**  **}**  **}** |
| **Reference** | Not allowed, see Clause 12.8:  “The combination of Level=Deep and Content=Reference is not allowed.” | If applied to the Submodel:  {  "keys"**: [ {**  "type"**:** "Submodel"**,**  "value"**:** "http://i40.customer.com/type/1/1/7A7104BDAB57E184"  **} ],**  "type"**:** "ModelReference"  **}** |
| If applied to the Property inside the SubmodelElementCollection, i.e. idShortPath “RotationSpeed.MaxRotationSpeed”:  {  "keys"**: [ {**  "type"**:** "Submodel"**,**  "value"**:** "http://i40.customer.com/type/1/1/7A7104BDAB57E184"  **}, {**  "type"**:** "SubmodelElementCollection"**,**  "value"**:** "RotationSpeed"  **}, {**  "type"**:** "Property"**,**  "value"**:** "MaxRotationSpeed"  **}],**  "type"**:** "ModelReference"  **}** |
| **Path** | If applied to the Submodel:  [  "RotationSpeed",  "RotationSpeed.MaxRotationSpeed"  **]** | If applied to the Submodel:  [  "RotationSpeed"  **]** |
| If applied to the Property inside the SubmodelElementCollection:  [ ] |

* 1. Examples for PATCH Operations

|  |  |
| --- | --- |
|  | **Deep (default)** |
| **Normal (default)** | If applied to the Submodel:  {  "modelType"**:** "Submodel"**,**  "id"**:** "http://i40.customer.com/type/1/1/7A7104BDAB57E184"**,**  "idShort": "TechnicalData"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "GlobalReference"**,**  "value"**:** "0173-1#01-AFZ615#016"  **} ],**  "type"**:** "ExternalReference"  **},**  "submodelElements"**: [ {**  "modelType"**:** "SubmodelElementCollection"**,**  "idShort"**:** "RotationSpeed"**,**  "semanticId": {  "keys": [ {  "type": "GlobalReference",  "value": "http://purl.org/iot/vocab/iot-taxonomy-lite#RotationalSpeed"  } ],  "type": "ExternalReference"  },  "value"**: [ {**  "modelType"**:** "Property"**,**  "idShort"**:** "MaxRotationSpeed"**,**  "category"**:** "PARAMETER"**,**  "semanticId"**: {**  "keys"**: [ {**  "type": "ConceptDescription",  "value": "0173-1#02-BAA120#008"  **} ],**  "type": "ExternalReference"  **},**  "valueType"**:** "xs:int"**,**  "value": "5000"  **} ]**  **} ]**  **}**  If applied to the SubmodelElementCollection, i.e. idShortPath “OperatingManual”:  {  "modelType"**:** "SubmodelElementCollection"**,**  "idShort"**:** "RotationSpeed"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "GlobalReference",  "value"**:** "http://purl.org/iot/vocab/iot-taxonomy-lite#RotationalSpeed"  **} ],**  "type"**:** "ExternalReference"  **},**  "value"**: [ {**  "modelType": "Property"**,**  "idShort"**:** "MaxRotationSpeed"**,**  "category"**:** "PARAMETER"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "ConceptDescription"**,**  "value"**:** "0173-1#02-BAA120#008"  **} ],**  "type"**:** "ExternalReference"  **},**  "valueType"**:** "xs:int"**,**  "value"**:** "5000"  **} ]**  **}**  If applied to the Property, i.e. idShortPath “OperatingManual.DocumentId”:  **{**  "modelType"**:** "Property"**,**  "idShort"**:** "MaxRotationSpeed"**,**  "category"**:** "PARAMETER"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "ConceptDescription"**,**  "value"**:** "0173-1#02-BAA120#008"  **} ],**  "type"**:** "ExternalReference"  **},**  "valueType"**:** "xs:int"**,**  "value"**:** "5000"  **}** |
| **Metadata** | If applied to the Submodel:  {  "modelType"**:** "Submodel"**,**  "id"**:** "http://i40.customer.com/type/1/1/7A7104BDAB57E184"**,**  "idShort"**:** "TechnicalData"  **}**  If applied to the SubmodelElementCollection, i.e. idShortPath “RotationSpeed”:  {  "modelType"**:** "SubmodelElementCollection"**,**  "idShort"**:** "RotationSpeed"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "GlobalReference"**,**  "value"**:** "http://purl.org/iot/vocab/iot-taxonomy-lite#RotationalSpeed"  **} ],**  "type"**:** "ExternalReference"  **}**  **}**  If applied to the Property, i.e. idShortPath “RotationSpeed.MaxRotationSpeed”:  **{**  "modelType"**:** "Property"**,**  "idShort"**:** "MaxRotationSpeed"**,**  "category"**:** "PARAMETER"**,**  "semanticId"**: {**  "keys"**: [ {**  "type"**:** "ConceptDescription"**,**  "value"**:** "0173-1#02-BAA120#008"  **} ],**  "type"**:** "ExternalReference"  **}**  **}** |
|
| **Value** | If applied to the Submodel:  {  "TechnicalData"**: {**  "RotationSpeed"**: {**  "MaxRotationSpeed"**:** "5000"  **}**  **}**  **}**  If applied to the SubmodelElementCollection, i.e. idShortPath “RotationSpeed”:  {  "RotationSpeed"**: {**  "MaxRotationSpeed"**:** "5000"  **}**  **}**  If applied to the Property, i.e. idShortPath “RotationSpeed.MaxRotationSpeed”:  {  "MaxRotationSpeed"**:** "5000"  **}** |

1. Backus-Naur-Form

The Backus-Naur form (BNF) – a meta-syntax notation for context-free grammars – is used to define grammars. For more information see Wikipedia[[9]](#footnote-10).

A BNF specification is a set of derivation rules, written as

<symbol> ::= \_\_expression\_\_

where:

* <[symbol](https://en.wikipedia.org/wiki/Symbol)> is a [nonterminal](https://en.wikipedia.org/wiki/Nonterminal) (variable) and the [\_\_expression\_\_](https://en.wikipedia.org/wiki/Expression_(mathematics)) consists of one or more sequences of either terminal or nonterminal symbols,
* ::= means that the symbol on the left must be replaced with the expression on the right,
* more sequences of symbols are separated by the [vertical bar](https://en.wikipedia.org/wiki/Vertical_bar) "|", indicating a [choice](https://en.wikipedia.org/wiki/Alternation_(formal_language_theory)), the whole being a possible substitution for the symbol on the left,
* symbols that never appear on a left side are [terminals](https://en.wikipedia.org/wiki/Terminal_symbol), while symbols that appear on a left side are [non-terminals](https://en.wikipedia.org/wiki/Nonterminal_symbol) and are always enclosed between the pair of angle brackets <>,
* terminals are enclosed with quotation marks: "text". "" is an empty string,
* optional items are enclosed in square brackets: [<item-x>],
* items existing 0 or more times are enclosed in curly brackets are suffixed with an asterisk (\*) such as <word> ::= <letter> {<letter>}\*,
* Items existing 1 or more times are suffixed with an addition (plus) symbol, +, such as <word> ::= {<letter>}+,
* round brackets are used to explicitly to define the order of expansion to indicate precedence, example: ( <symbol1> | <symbol2> ) <symbol3>,
* text without quotation marks is an informal explanation of what is expected; this text is cursive if grammar is non-recursive and vice versa.

Example:

<contact-address> ::= <name> "e-mail addresses:" <e-mail-Addresses>

<e-mail-Addresses> ::= {<e-mail-Address>}\*

<e-mail-Addresse> ::= <local-part> "@" <domain>

<name> ::= characters

<local-part> ::= characters conformant to local-part in RFC 5322

<domain> ::= characters conformant to domain in RFC 5322

Valid contact addresses:

Hugo Me e-mail addresses: Hugo@example.com

Hugo e-mail addresses: Hugo.Me@text.de

Invalid contact addresses:

Hugo

Hugo Hugo@ example.com

Hugo@example.com

1. Bibliography

[1] Specification of the Asset Administration Shell. Part 1: Metamodel, Version 3.0. Industrial Digital Twin Association (IDTA), March 2023. Online. Available: https://industrialdigitaltwin.org/en/content-hub

[2] Specification of the Asset Administration Shell. Part 3a: Data Specification – IEC 61360, Version 3.0. Industrial Digital Twin Association (IDTA), March 2023. Online. Available: https://industrialdigitaltwin.org/en/content-hub

[3] Specification of the Asset Administration Shell. Part 5: Package File Format, Version 3.0. Industrial Digital Twin Association (IDTA), March 2023. Online. Available: https://industrialdigitaltwin.org/en/content-hub

[4] [Tom Preston-Werner](file:///C:/Users/sbader/AppData/Local/Temp/Tom%20Preston-Werner). Semantic Versioning. Version 2.0.0. Online. Available: <https://semver.org/spec/v2.0.0.html>

[5] RFC 8820: URI Design and Ownership. Internet Engineering Task Force (IETF), 2020. Online. Available: https://tools.ietf.org/html/rfc8820

[6] DIN SPEC 91406: “Automatic identification of physical objects and information on physical objects in IT systems, particularly IoT systems”. December 2019. Online. Available: <https://www.beuth.de/de/technische-regel/din-spec-91406/314564057>

[7] Decentralized Identifiers (DIDs) v1.0. Edited by Manu Sporny, Amy Guy, Markus Sabadello, and Drummond Reed. W3C Recommendation. Online. Available: https://www.w3.org/TR/did-core/

[8] OData Version 4.01 Part 1: Protocol. Edited by Michael Pizzo, Ralf Handl, and Martin Zurmuehl. OASIS Standard. Online. Available: https://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part1-protocol.html

Change Notes

* 1. General
* \* Means not backward compatible
* (\*) means not backward compatible but just renaming
  1. Interface Changes w.r.t. V1.0RC03 to V3.0

Major Changes:

* Introduction of service specifications and profiles
* Introduction of pagination for "GetAll\*" API operations in http/REST
* Distinction between replace and update for operations
* SerializationModifier Content as path: $metadata, $value, $reference, $path
* Introduction of length constraints for string attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BWC** | **Interface Change** | | **Kind of Change** | **Comment** |
|  | | Submodel | New | PatchSubmodel and PatchSubmodelElementByPath  (PUT to completely replace and PATCH to update content) |
|  | | Asset Administration Shell,  Submodel,  AASX File Server,  AAS Repository,  Submodel Repository,  CD Repository,  AAS Registry,  Submodel Registry,  AAS Basic Discovery | Changed | Add Pagination:  GetAllAssetAdministrationShells  GetAllAssetAdministrationShellsByAssetId  GetAllAssetAdministrationShellsByIdShort  GetAllSubmodelReferences  GetAllSubmodels  GetAllSubmodelsBySemanticId  GetAllSubmodelsByIdShort  GetAllSubmodelElements  GetSubmodelElementByPath  GetAllConceptDescriptions  GetAllConceptDescriptionsByIdShort  GetAllConceptDescriptionsByIsCaseOf  GetAllConceptDescriptionsByDataSpecificationReference  GetAllAssetAdministrationShellDescriptors  GetAllSubmodelDescriptors  GetAllAssetAdministrationShellIdsByAssetLink  GetAllAASXPackageIds |
|  | | Submodel | Changed | SerializationModifier Content as path: $metadata, $value, $reference, $path |
|  | | Asset Administration Shell | New | GetThumbnail, PutThumbnail |
|  | | Submodel Repository | New | PatchSubmodelForId was missing |
|  | | Registry | New | Add extensions to descriptor |
|  | | AssetAdministrationShellDescriptor | New | Add the attributes assetKind and assetType |
|  | | SubmodelDescriptor | New | Add supplementalSemanticId |
|  | | \* | Changed | Rename GetDescriptor to GetDescription |
|  | | \* | Changed | API versioning with major + minor |
|  | | \* | New | Profiles |
|  | | \* | Changed | Clarify service specifications and APIs |
|  | | CD Registry | Changed | Renaming parameter ‘cdIdentifier’ in GetConceptDescriptionById to ‘id’. Parameter has not been changed in the HTTP API. |

* 1. Operation Changes w.r.t. V1.0RC03 to V3.0

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation Change Old** | **Operation Change New** | **Kind of Change** | **Comment** |
| GetDescriptor | GetDescription | Changed | Rename, get profiles |
|  |  |  |  |

* 1. Interface Changes w.r.t. V1.0RC02 to V1.0RC03

|  |  |  |  |
| --- | --- | --- | --- |
| **BWC** | **Interface Change** | **Kind of Change** | **Comment** |
| \* | Discovery | Changed | IndentifierKeyValuePair to SpecificAssetId |
| \* | Submodel | Changed | SubmodelElementStruct remains as SubmodelElementCollection |
| \* | Submodel | Changed | ModelReference and GlobalReference are combined back to Reference |
| \* | Submodel | Changed | Rename trimmed to metadata |
|  | Submodel | New | Add GetFileByPath |
|  | Submodel | New | Add PutFileByPath |
| \* | Submodel | Changed | InvokeOperationAsync |
|  | Registry | Changed | Endpoint |
| \* | Registry | Changed | Remove /registry from REST path |
| \* | All | New | API Versioning adds a prefix to all interfaces |

* 1. Operation Changes w.r.t. V1.0RC02 to V1.0RC03

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation Change Old** | **Operation Change New** | **Kind of Change** | **Comment** |
|  |  | Changed | inputArgument and inoutputArgument are OperationVariable |
| GetAllAssetAdminstrationShellsByAssetLink |  | Changed | IdentifierKeyValuePair to SpeicifcAssetId |
| GetAllAssetLinksById |  | Changed | IdentifierKeyValuePair to SpeicifcAssetId |
| PostAllAssetLinksById |  | Changed | IdentifierKeyValuePair to SpeicifcAssetId |

* 1. Interface Changes w.r.t. V1.0RC01 to V1.0RC02

|  |  |  |  |
| --- | --- | --- | --- |
| **BWC** | **Interface Change** | **Kind of Change** | **Comment** |
| \* | Asset Administration Shell | Changed | Renamed:  RemoveSubmodelReference to DeleteSubmodelReference  Removed:  PutSubmodelReference, PatchAssetAdministrationShell  New:  GetAssetInformation  PutAssetInformation  GetAllSubmodelReferences  PostSubmodelReference |
| \* | Submodel | Changed | Removed:  GetAllSubmodelElementsByParentPathAndSemanticId, GetAllSubmodelElementsBySemanticId  New:  PutSubmodel, PostSubmodelElement, PostSubmodelElementByPath |
| \* | Asset Administration Shell Serialization | Changed | Renamed:  GetSerializationByIds to GenerateSerializationByIds  Removed:  GetAASX |
|  | AASX File Server | New | New interface |
| (\*) | Asset Administration Shell Registry | Changed | Renamed: PutAssetAdministrationShellDescriptor to PutAssetAdministrationShellDescriptorById  New:  PostAssetAdministrationShellDescriptor |
| (\*) | Submodel Registry | Changed | Renamed:  PutSubmodelDescriptor to PutSubmodelDescriptorById  New: PostSubmodelDescriptor |
| (\*) | Asset Administration Shell Repository | Changed | Renamed:  GetAllAssetAdministrationShellsById to GetAssetAdministrationShellById,  PutAssetAdministrationShell to PutAssetAdministratioShellById  New:  PostAssetAdministrationShell |
| (\*) | Submodel Repository | Changed | Renamed:  PutSubmodel to PutSubmodelById  New:  PostSubmodel |
| (\*) | Asset Administration Shell Basic Discovery | Changed | Removed: GetAllAssetAdministrationShellIdsByAssetId,  PutAssetId  New: GetAllAssetAdministrationShellIdsByAssetLink, GetAllAssetLinksById, PutAllAssetLinksById, DeleteAllAssetLinksById |
| (\*) | Submodel Discovery Basic | Removed |  |
| (\*) | Concept Description Repository | Changed | Renamed: GetAllConceptDescriptionsWtihDataSpecificationReference to GetAllConceptDescriptionsByDataSpecificationReference, PutConceptDescription to PutConceptDescriptionById  New:  PostConceptDescription |

* 1. Operation Changes w.r.t. V1.0RC01 to V1.0RC02

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation Change Old** | **Operation Change New** | **Kind of Change** | **Comment** |
| PatchAssetAdministrationShell |  | Removed |  |
| PutSubmodelReference |  | Removed | Substituted by PostSubmodelReference |
|  | PostSubmodelReference | New | For PutSubmodelReference |
| RemoveSubmodelReference | DeleteSubmodelReference | Changed |  |
|  | GetAllSubmodelReferences | New |  |
|  | PostSubmodelReference | New |  |
|  | GetAssetInformation | New |  |
|  | PutAssetInformation | New |  |
|  | PutSubmodel | New |  |
|  | PostSubmodelElement | New |  |
|  | PostSubmodelElementByPath | New |  |
| GetAllSubmodelElementsByParentPathAndSemanticId |  | Removed |  |
| GetAllSubmodelElementsBySemanticId |  | Removed |  |
| GetAASX |  | Removed |  |
| GetSerializationByIds | GenerateSerializationByIds | Renamed |  |
|  | GetAllAASXPackageIds | New |  |
|  | GetAASXByPackageId | New |  |
|  | PostAASXPackage | New |  |
|  | PutAASXByPackageId | New |  |
|  | DeleteAASXByPackageId | New |  |
| PutAssetAdministrationShellDescriptor | PutAssetAdministrationShellDescriptorById | Changed | Naming pattern byId |
|  | PostAssetAdministrationDescriptor | New |  |
| PutSubmodelDescriptor | PutSubmodelDescriptorById | Changed | Naming pattern byId |
|  | PostSubmdeoDescriptor | New |  |
| GetAllAssetAdministrationShellsById | GetAssetAdministrationShellById | Changed | Naming pattern resource singular |
|  | PostAssetAdministrationShell | New |  |
| PutAssetAdministrationShell | PutAssetAdministrationShellById | Changed | Naming pattern byId |
| PutSubmodel | PutSubmodelById | Changed | Naming pattern byId |
|  | PostSubmodel | New |  |
| GetAllAssetAdministrationShellIdsByAssetId |  | Removed | substituted by GetAllAssetAdministrationShellIdsByAssetLink and GetAllAssetLinksById |
| PutAssetId |  | Removed | Substituted by PutAllAssetLinksById and DeleteAllAssetLinksById |
|  | GetAllAssetAdministrationShellIdsByAssetLink | New | Before: GetAllAssetAdministrationShellIdsByAssetId |
|  | GetAllAssetLinksById | New |  |
|  | PutAllAssetLinksById | New |  |
|  | DeleteAllAssetLinksById | New |  |
| GetAllSubmodelIdsBySemanticId |  | Removed |  |
| GetAllConceptDescriptionsWithDataSpecificationReference | GetAllConceptDescriptionsByDataSpecificationReference | Renamed | Renaming With  By |
| PutConceptDescription | PutConceptDescriptionById | Changed | Naming pattern byId |
|  | PostConceptDescription | New |  |

www.industrialdigitaltwin.org

1. <https://github.com/admin-shell-io/id> [↑](#footnote-ref-2)
2. For easier reading only the standard paths are shown in the following: $metadata, $value, $reference and $path parameter paths are additionally contained in the OpenAPI file. [↑](#footnote-ref-3)
3. Related OpenAPI file: <https://app.swaggerhub.com/apis/Plattform_i40/AasxFileServerServiceSpecification/V3.0_SSP-001> [↑](#footnote-ref-4)
4. <https://www.w3.org/TR/xmlschema-2/> [↑](#footnote-ref-5)
5. cf. https://openmanufacturingplatform.github.io/sds-documentation/bamm-specification/2.0.0/datatypes.html [↑](#footnote-ref-6)
6. cf. <https://openmanufacturingplatform.github.io/sds-documentation/bamm-specification/v1.0.0/datatypes.html> (with adjustments for +/-INF, NaN, and language-typed literal support) [↑](#footnote-ref-7)
7. <http://semver.org> [↑](#footnote-ref-8)
8. see Chapter 2.4 of RFC 8977 [↑](#footnote-ref-9)
9. https://en.wikipedia.org/wiki/Backus%E2%80%93Naur\_form [↑](#footnote-ref-10)