Specification of the Asset Administration Shell

Part 2: Application Programming Interfaces

SPECIFICATION

IDTA Number: 01002
Version 3.1

Specification of the Asset Administration Shell

This specification is part of the Asset Administration Shell Specification series.

Version

This is version 3.1 of the specification IDTA-01002.

Previous version: 3.0.3

Metamodel Versions

This document (version 3.1) uses the following parts of the “Specification of the Asset Administration Shell” series:

- IDTA-01001 Part 1: Metamodel in version 3.1 [1]
- IDTA-01003-a Part 3a: Data Specification – IEC 61360 in version 3.1 [2]
- IDTA-01004 Part 4: Security in version 3.1 [3]
- IDTA-01005 Part 5: Package File Format (AASX) in version 3.1 [4]

Notice

IDTA Document Number: IDTA-01002-3-1

DOI: https://doi.org/10.62628/IDTA.01002-3-1

May 2025

How to Get in Contact

Contact: IDTA

Sources: GitHub

Feedback:

- New issues, bugs
- Questions and Answers

Editorial Notes

History

This document (version 3.1) was produced by the Work Stream "Specification of the Asset Administration Shell" of the Working Group "Open Technology" of the Industrial Digital Twin Association (IDTA). It is the first version published as html document and maintained completely open source.

Version 3.0, the first release of the API, was finalized 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).

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”.

Versioning

This specification is versioned using Semantic Versioning 2.0.0 (semver) and follows the semver specification [9].

Conformance

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 RFC2119 RFC8174[1]:

- MUST word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- MUST NOT This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
- SHOULD This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

Terms and Definitions

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 63278-1:2023.

API

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

API Operation

specification of a single operation (procedure) 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 63278-1:2023, note added]

interface

shared boundary between two entities defined by functional characteristics, signal characteristics, or other characteristics as appropriate

- [SOURCE: IEC 61800-7-1: 2015]

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

distinct part of the functionality that is provided by an entity through interfaces

- [SOURCE: IEC 63278-1:2023; IEC 60050-741:2020, 741-01-28]

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 63278-1:2023]

SubmodelElement

element of a Submodel

- [SOURCE: IEC 63278-1:2023]

Abbreviations

Abbreviation	Description
AAS	Asset Administration Shell
AASX	Package file format for the AAS
AML	AutomationML, Automation Markup Language
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

Abbreviation	Description
OPC UA	OPC Unified Architecture maintained by the OPC Foundation
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	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UUID	Universally Unique Identifier
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.

[1] https://www.ietf.org/rfc/rfc2119.txt

Preamble

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

Introduction gives an introduction to the topic.

General topics are discussed in *General*.

The query language is sepcified in *Query Language*.

The technology-neutral specification of the interfaces of the Asset Administration Shell can be found in Clauses:

- API Interfaces
- Data Types for Payload
- Basic Operation Parameters

Following Clauses define the API specification for HTTP/REST:

- HTTP/REST API
- Service Profiles
- Interactions
- API Code Generation

Summary and Outlook provides a summary and outlook.

Annex SerializationModifier Examples contains examples for the serialization modifiers.

Annex Backus Naur Form defines the grammar language used in the specification. Annex UML contains information about UML, while Annex Class Table Templates provides the tables used to specify UML classes etc. as used in this specification. Annex API Table Templates explains the templates used to specify operations and interfaces.

Handling Constraints explains the numbering of constraints used in the specification. Overview Constraints gives an (non-normative) overview of all constraints used in the documents.

The *Change Log* describes metamodel changes compared to previous versions.

Finally, a *Bibliography* is given.

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 [4] mainly considered file exchange, this specification focuses on the API that allows online access to information provided by the AAS (see Figure 1).

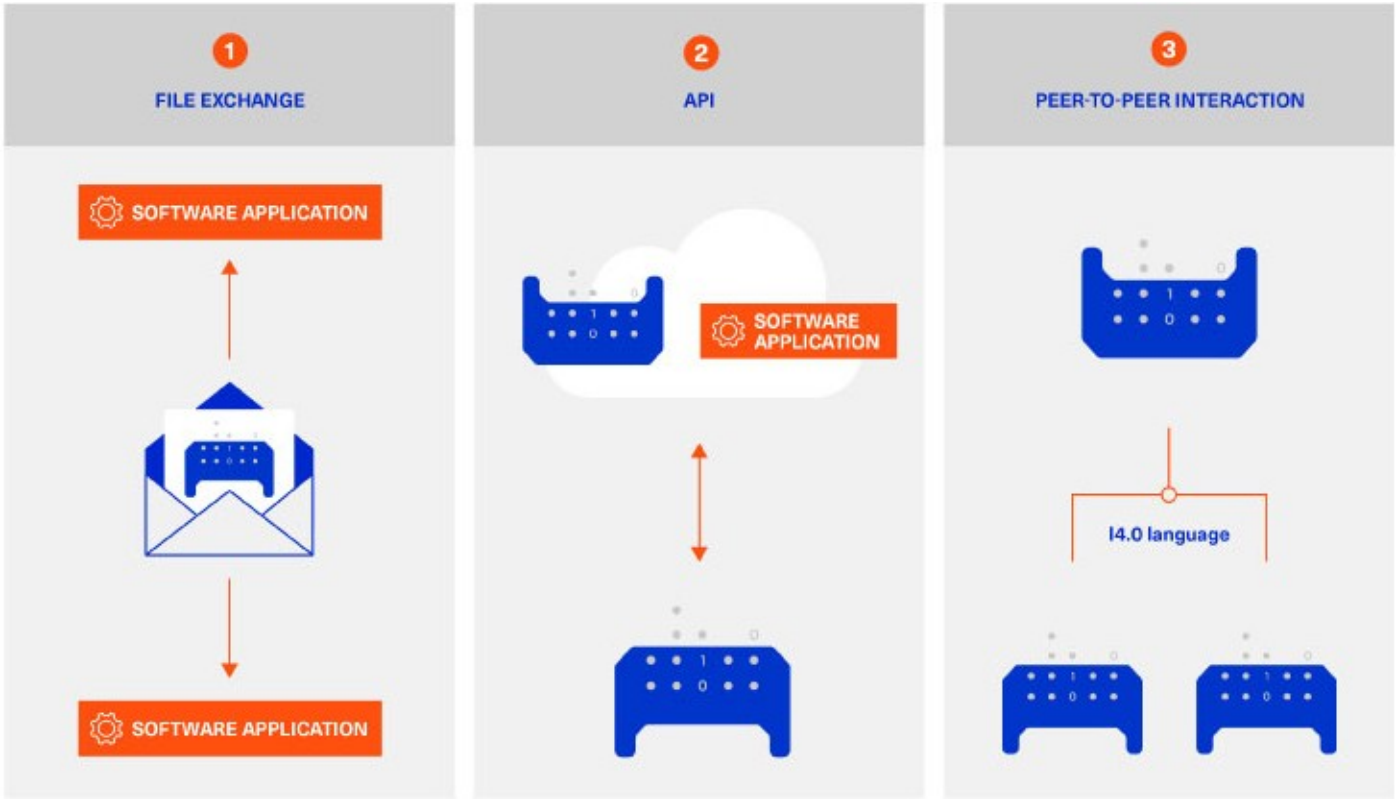


Figure 1. Types of Information Exchange via Asset Administration Shells

General

Services, Interfaces and Interface Operations

This document uses the Industrie 4.0 Service Model illustrated in [general:::i40-service-model] 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 of 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.
- **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 a single operation (procedure) 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.

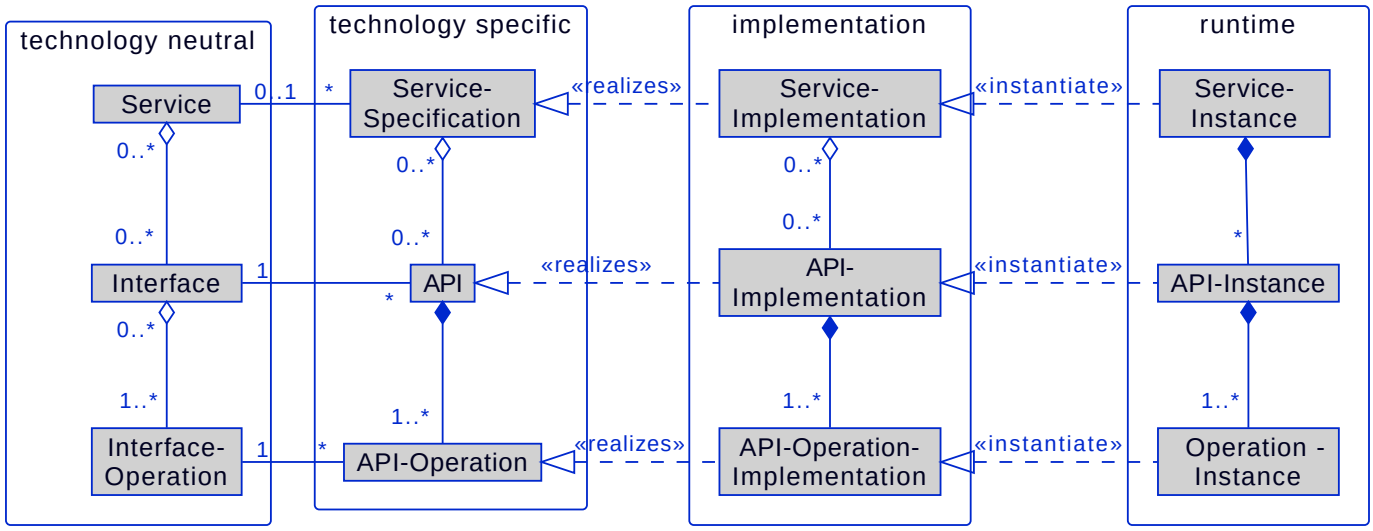


Figure 2. Services, Interfaces & APIs and Operations

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 Asset Administration Shell Interfaces.

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. Metamodel Specification Details 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.

- QUERY: returns a list of resources based on conditions expressed through the AAS Query Language.
- PUT: replaces an existing resource or creates it, when not found.
- PUTBULK: bulk replacement.
- 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.
- POSTBULK: bulk creation.
- 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 1: values remain unchanged with content=metadata.

- DELETE: deletes a resource based on a given identifier.
- DELETEBULK: bulk deletion.
- INVOKE: invokes an operation at a specified path.
- INVOKEASYNC: asyncronous invocation of an operation.
- SEARCHALL: returns a list fo resources based on asset links.

Note 2: 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" | "Query" |"Put" | "PutBulk" | "Post" | "PostBulk" |
"Patch" | "Delete" | "DeleteBulk" | "Invoke" | "InvokeAsync" | "SearchAll"

<Model Element Name> ::= "AssetAdministrationShell"["Ids"|"s"] |
"AssetAdministrationShellDescriptor"["Ids"|"s"] | "SubmodelReference"["s"] |
"AssetInformation" | "Submodel"["Ids"|"s"] | "SubmodelDescriptor"["Ids"|"s"] |
"SubmodelElement"["s"] | "ConceptDescription"["Ids"|"s"]

<Modifier> ::= "Value" | "IdShortPath" | "Reference"

<By-Qualifier> ::= | "Id" | "SemanticId" | "ParentPathAndSemanticId" | "Path" | "AssetId" |
"IdShort" | "IsCaseOf" | "DataSpecificationReference"
```

Examples:

GetSubmodel has method verb "Get" and model element name "Submodel".

GetAllSubmodelElementsByPath has method verb "GetAll" and model 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 EN IEC 61406 [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 Industrial Digital Twin Association (IDTA). 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 https://github.com/admin-shell-io/id. 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.).

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.

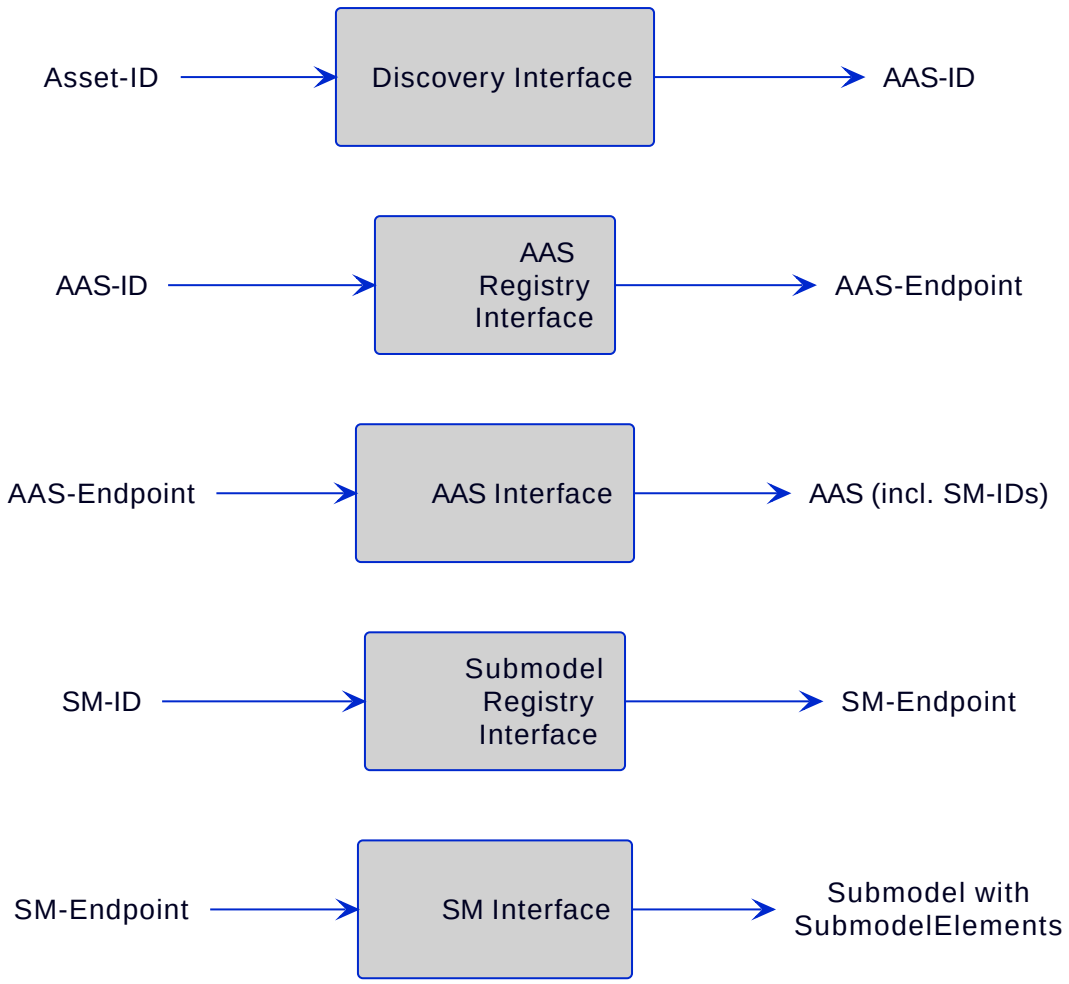


Figure 3. Retrieval of Asset-related Information by AAS and Submodels

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 AssetAdministrationShellRepositoryServiceSpecification/V3.1_SSP-001.[1]:

/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 as described in the related OpenAPI file https://app.swaggerhub.com/apis/Plattform_i40/AasxFileServerServiceSpecification/V3.1_SSP-001].

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 AssetAdministrationShellServiceSpecification/V3.1_SSP-001:

/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.

[1] 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.

Query Language

Many use cases of the Asset Administration Shell require the involvement of a high number of Asset Administration Shells at the same time. Executing the business logic on all potentially involved Asset Administration Shells solely by the client application requires a huge amount of transferred data objects and bandwidth. It is therefore necessary to send parts of the selection conditions to the AAS hosting systems. The AAS Query Language enables AAS clients to describe and handover their interests and AAS servers to only respond with the needed data objects.

The Asset Administration Shell propagates a Query Language inspired by the so-called Resource Query Language (RQL [11]). This language follows a simplified grammar and expressiveness compared with RQL and other languages with a similar scope, e.g. SPARQL [12], GQL [13], or JsonPath [14]. The same language shall be used independent of the communication protocol, therefore, it is not limited to the HTTP API of the AAS. However, different communication protocols may treat aspects differently, e.g., by requiring different query or result set serialisations, variations in the order of constructs, or endpoint patterns. Nevertheless, all share the same expressiveness, and under same conditions a query shall lead to the same results independently of the chosen communication protocol.

> Note: The AAS Query Language and the AAS Access Rules [3] share the same BNF grammar. Therefore, the result is a superset, with query as the entrypoint for the Query Language.

Use Case Examples

Examples for Use Cases are search queries in the submodels *DigitalNameplate*, *TechnicalData* or *HandoverDocumentation*. *DigitalNameplate* has a fixed structure, only the bottom part with Markings and AssetSpecificProperties is variable. Queries in *DigitalNameplate* can be well done by providing idShortPaths to the elements in the query, e.g. searching for all DigitalNameplates with a certain ManufacturerName and CountryOfOrigin. TechnicalData has its major variable part on the bottom and a small fixed structure on the top. *TechnicalData* includes ProductClassifications with ProductClassId, so that e.g. by ECLASS ClassId a certain product type can be queried. *TechnicalData* includes all the technical properties, e.g. width of the product. An example is to search for motor starters (*ProductClassifications[].ProductClassId* = 27-37-09-05) with a width less than 100 mm (*semanticId* = 0173-1#02-BAF016#006, *value* < 100). HandoverDocumentation consists basically of a list of document information according VDI 2770 (*DocumentedEntity{00}*). It is expected, that this will be changed to SubmodelElementList like other V3 Submodels. An example is to search for Documents of a certain ClassId (*semanticId* = 0173-1#02-ABH996#001, *ClassId* = "03-01", i.e. Commissioning) in a certain language (*semanticId* = 0173-1#02-AAN468#006, e.g. *value* = "nl").

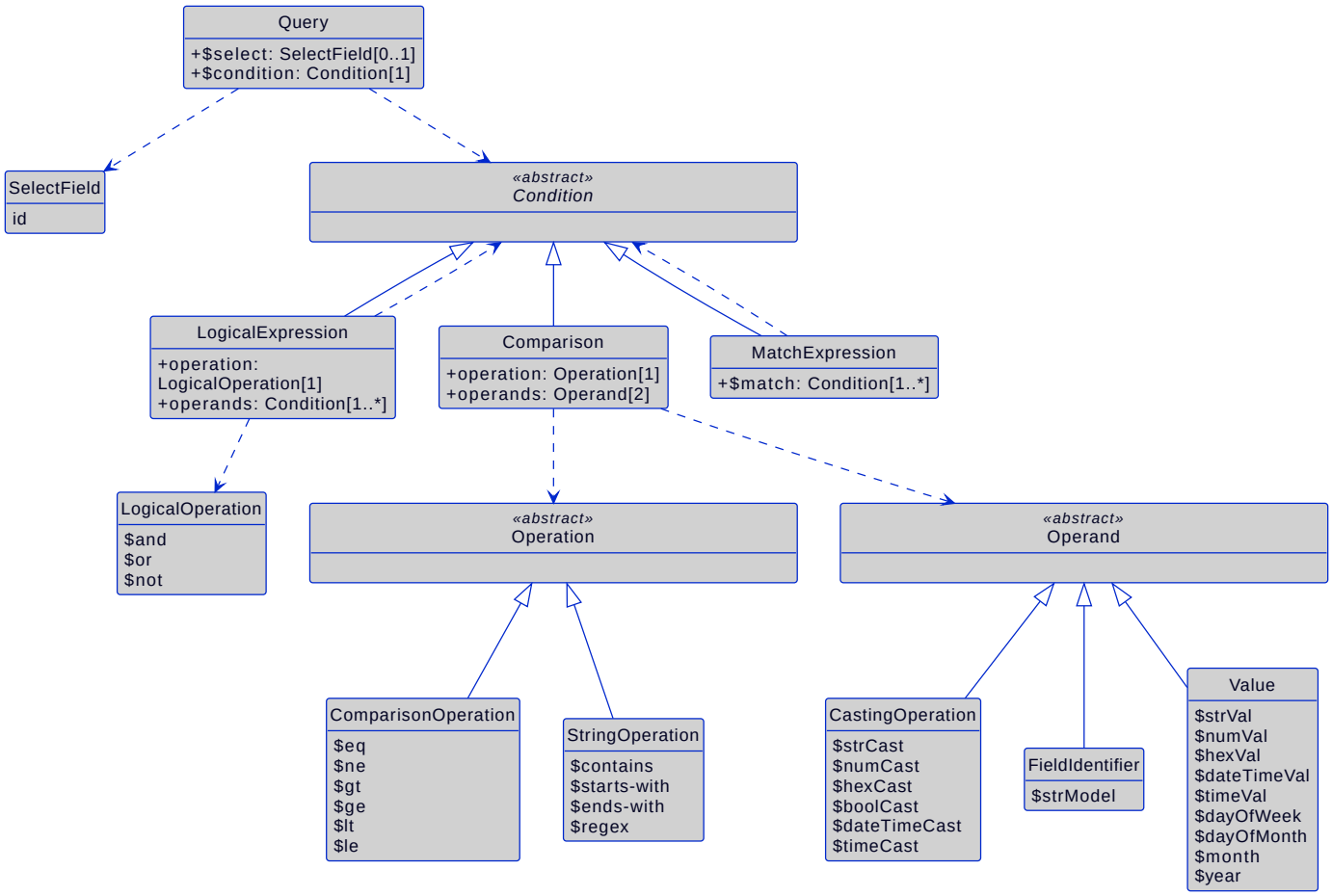


Figure 4. Main elements of the AAS Query Language

Limitations

The AAS Query Language is not intended as feature-comparable to existing query languages like SPARQL or GQL. In case the expressiveness of such technologies is needed, a software vendor might additionally extend his service accordingly. However, in order to keep the additional overhead for AAS Query Language implementers as small as possible, the following limitations apply:

1. The AAS Query Language focusses on Identifiables: It is only possible to formulate queries for Asset Administration Shells, Submodels, and their according Descriptors, as well as ConceptDescription objects. In addition, also their identifiers (values of the "id" attributes) can be queried. This means in particular that not every Referable can be queried.
2. Only Repository and Registry Services have specified query functionalities: Querys on Asset Administration Shell Services or Submodel Services are not defined.
3. Only selected attributes defined by the AAS data model [1] can be used as conditions for returned objects: Usage of custom or vendor-specific attributes outside of the AAS specifications is generally not recommended and may lead to a rejections of the query by the receiving systems.
4. It is not possible to traverse through the specific fields of the SubmodelElements Entity (e.g. Entity/statements), AnnotatedRelationshipElement (e.g. AnnotatedRelationshipElement/annotations), or Operation (e.g. Operation/outputVariables). Only the fields semanticId idShort, value, and valueType are available for SubmodelElements.
5. The rendering of the "oneOf" clause of logicalExpressions does not work in SwaggerHub. This issue has no effect on the functionality of the requirement.

Search in AAS Hierarchy

AAS consists of a hierarchical structure. An AAS references its "contained" submodels. A submodel contains a list of SubmodelElements, which may include further SubmodelElements, and so on. The query language allows to both address a specific instance of the data or to search all data in a repository. A specific instance is addressed by

specifying the id of an Identifiable and the idShortPath to a Referable in a Submodel. If such explicit information is not specified, a hierarchical search is made. By the $aas FieldIdentifier conditions for all the relevant AAS are defined. By the $sm FieldIdentifier conditions for all the relevant submodels are defined. $sm can be used with and without $aas. If $aas and $sm are used together, only the submodels referenced by the matching $aas are searched by the $sm expression. The same search principle is used, when combining $sm and $sme. In such case only the SubmodelElements which are part of matching submodels by $sm expression are searched by the $sme expression. Several such hierarchical conditions may even be combined by $match expressions.

Grammar

The content and structure of the AAS Query Language is defined in the context-free Backus-Naur form (BNF). See Appendix Backus Naur Form for more details on BNF. The detailed serialisation and interaction patterns are defined by the different technology mappings, e.g., the AAS HTTP API represents AAS Queries as JSON objects (see Clause Serialisation). Other mappings may define different serialisations, however, all have to follow the general structure of queries defined in the following grammar.

This is the combined grammar for the AAS Query Language and the AAS Access Rules defined by the AAS Security specification [3].

```
<grammar> ::= <query> | <AllAccessPermissionRules>

<query> ::= <selectStatement>? <logicalExpression>
<selectStatement> ::= "$select" <ws> "id" <ws>

<logicalExpression> ::= <logicalNestedExpression> | <logicalOrExpression> |
<logicalAndExpression> |
<logicalNotExpression> | <matchExpression> | <BoolLiteral> | <castToBool> |
<singleComparison>
<logicalNestedExpression> ::= "(" <ws> <logicalExpression> ")" <ws>
<logicalOrExpression> ::= "$or" <ws> "(" <ws> <logicalExpression> ( "," <ws>
<logicalExpression> )+ ")" <ws>
<logicalAndExpression> ::= "$and" <ws> "(" <ws> <logicalExpression> ( "," <ws>
<logicalExpression> )+ ")" <ws>
<logicalNotExpression> ::= "$not" <ws> "(" <ws> <logicalExpression> ")" <ws>

<matchExpression> ::= ( "$match" <ws> "(" <ws> ( <singleComparison> | <matchExpression> ) (
"," <ws> ( <singleComparison> | <matchExpression> ) )* ")" <ws> )

<singleComparison> ::=
    <stringComparison> |
    <numericalComparison> |
    <hexComparison> |
    <boolComparison> |
    <dateTimeComparison> |
    <timeComparison>

<allComparisons> ::= ( "$eq" | "$ne" | "$gt" | "$lt" | "$ge" | "$le" )

<stringComparison> ::=
    ( ( "$starts-with" | "$ends-with" | "$contains" | "$regex") <ws> "(" <ws>
<stringOperand> <ws> "," <ws> <stringOperand> <ws> ")" <ws> ) |
    ( <stringOperand> <ws> <allComparisons> <ws> <stringOperand> <ws> ) |
    ( <stringOperand> <ws> <allComparisons> <ws> <FieldIdentifierString> <ws> ) |
    ( <FieldIdentifierString> <ws> <allComparisons> <ws> <stringOperand> <ws> )
```

```
<numericalComparison> ::=
    ( <numericalOperand> <ws> <allComparisons> <ws> <numericalOperand> <ws> ) |
    ( <numericalOperand> <ws> <allComparisons> <ws> <FieldIdentifierString> <ws> ) |
    ( <FieldIdentifierString> <ws> <allComparisons> <ws> <numericalOperand> <ws> )

<hexComparison> ::=
    <hexOperand> <ws> <allComparisons> <ws> <hexOperand> <ws>

<boolComparison> ::=
    <boolOperand> <ws> ( "$eq" | "$ne" ) <ws> <boolOperand> <ws>

<dateTimeComparison> ::=
    <dateTimeOperand> <ws> <allComparisons> <ws> <dateTimeOperand> <ws>

<dateTimeToNum> ::=
    ( "$dayOfWeek" | "$dayOfMonth" | "$month" | "$year" ) <ws> "(" <ws> <dateTimeOperand>
<ws> ")" <ws>

<timeComparison> ::=
    <timeOperand> <ws> <allComparisons> <ws> <timeOperand> <ws>

<operand> ::= <stringOperand> | <numericalOperand> | <hexOperand> | <boolOperand> |
<dateTimeOperand> | <timeOperand>

<stringOperand> ::=
    <FieldIdentifierString> | <StringLiteral> | <castToString> | <SingleAttribute>

<numericalOperand> ::=
    <NumericalLiteral> | <castToNumerical> | <dateTimeToNum>

<hexOperand> ::=
    <HexLiteral> | <castToHex>

<boolOperand> ::=
    <BoolLiteral> | <castToBool>

<dateTimeOperand> ::=
    <DateTimeLiteral> | <castToDateTime> | <GlobalAttribute>

<timeOperand> ::=
    <TimeLiteral> | <castToTime>

<castToString> ::=
    "str" <ws> "(" <ws> <operand> <ws> ")" <ws>

<castToNumerical> ::=
    "num" <ws> "(" <ws> <operand> <ws> ")" <ws>

<castToHex> ::=
    "hex" <ws> "(" <ws> <operand> <ws> ")" <ws>
```

```
<castToBool> ::=
    "bool" <ws> "(" <ws> <operand> <ws> ")" <ws>

<castToDateTime> ::=
    "dateTime" <ws> "(" <ws> <stringOperand> <ws> ")" <ws>

<castToTime> ::=
    "time" <ws> "(" <ws> ( <stringOperand> | <dateTimeOperand> ) <ws> ")" <ws>

<AllAccessPermissionRules> ::=
    ( "DEFATTRIBUTES" <ws> <StringLiteral> <ws> <AttributeGroup> <ws> )*
    ( "DEFACLS" <ws> <StringLiteral> <ws> <ACL> <ws> )*
    ( "DEFOBJECTS" <ws> <StringLiteral> <ws> <ObjectGroup> <ws> )*
    ( "DEFFORMULAS" <ws> <StringLiteral> <ws> <Condition> <ws> )*
    ( <AccessPermissionRule> <ws> )*

<AccessPermissionRule> ::=
    "ACCESSRULE:" <ws>
    ( <ACL> | <UseACL> ) <ws>
    "OBJECTS:" <ws>
    ( <SingleObject> <ws> )*
    ( <UseObjectGroup> <ws> )*
    "FORMULA:" <ws>
    ( <Condition> | <UseFormula> ) <ws>
    ( "FILTER:" <ws> <FragmentObject> <ws> ( <Condition> | <UseFormula> ) <ws> )?

<ACL> ::=
    "ATTRIBUTES:" <ws>
    ( <SingleAttribute> <ws> )*
    ( <UseAttributeGroup> <ws> )*
    "RIGHTS:" <ws> <Right> <ws> ( <Right> <ws> )*
    "ACCESS:" <ws> <Access> <ws>

<UseACL> ::=
    "USEACLS" <ws> <StringLiteral> <ws>

<Right> ::=
    "CREATE" | "READ" | "UPDATE" | "DELETE" | "EXECUTE" | "VIEW" | "ALL" | "TREE"

<Access> ::=
    "ALLOW" | "DISABLED"

<SingleAttribute> ::=
    <ClaimAttribute> | <GlobalAttribute> | <ReferenceAttribute>

<ClaimAttribute> ::=
    "CLAIM" <ws> "(" <ws> <ClaimLiteral> <ws> ")"

<GlobalAttribute> ::=
    "GLOBAL" <ws> "(" <ws> ( "LOCALNOW" | "UTCNOW" | "CLIENTNOW" | "ANONYMOUS" ) <ws> ")"

<ReferenceAttribute> ::=
```

```
    "REFERENCE" <ws> "(" <ws> <ReferenceLiteral> <ws> ")"

<AttributeGroup> ::=
    ( <SingleAttribute> <ws> )*
    ( <UseAttributeGroup> <ws> )*

<UseAttributeGroup> ::=
    "USEATTRIBUTES" <ws> <StringLiteral> <ws>

<SingleObject> ::=
    <RouteObject> | <IdentifiableObject> | <ReferableObject> | <FragmentObject> |
<DescriptorObject>

<RouteObject> ::=
    "ROUTE" <ws> <RouteLiteral> <ws>

<IdentifiableObject> ::=
    "IDENTIFIABLE" <ws> <IdentifiableLiteral> <ws>

<ReferableObject> ::=
    "REFERABLE" <ws> <ReferableLiteral> <ws>

<FragmentObject> ::=
    "FRAGMENT" <ws> <FragmentLiteral> <ws>

<DescriptorObject> ::=
    "DESCRIPTOR" <ws> <DescriptorLiteral> <ws>

<ObjectGroup> ::=
    ( <SingleObject> <ws> )*
    | ( <UseObjectGroup> <ws> )*

<UseObjectGroup> ::=
    "USEOBJECTS" <ws> <StringLiteral> <ws>

<UseFormula> ::=
    "USEFORMULAS" <ws> <StringLiteral> <ws>

<Condition> ::= <logicalExpression> <ws>

<DateTimeLiteral> ::= <datetime> <ws>
<TimeLiteral> ::= <time> <ws>
<datetime> ::= <date> <ws> ( "T" | " " ) <ws> <time> <ws> ( <timezone> <ws> )?
<date> ::= <year> <ws> "-" <ws> <month> <ws> "-" <ws> <day> <ws>
<year> ::= <digit> <ws> <digit> <ws> <digit> <ws> <digit> <ws>
<month> ::= <digit> <ws> <digit> <ws>
<day> ::= <digit> <ws> <digit> <ws>
<time> ::= <hour> <ws> ":" <ws> <minute> <ws> ( ":" <ws> <second> <ws> )? ( "." <ws>
<fraction> <ws> )?
<timezone> ::= ( "Z" | ( "+" | "-" ) <ws> <hour> <ws> ":" <ws> <minute> <ws> )
<hour> ::= <digit> <ws> <digit> <ws>
<minute> ::= <digit> <ws> <digit> <ws>
```

```
<second> ::= <digit> <ws> <digit> <ws>
<fraction> ::= <digit>+ <ws>

<digit> ::= [0-9] <ws>
<StringLiteral> ::= "\"" ( [A-Z] | [a-z] | [0-9] | "/" | "*" | "[" | "]" | "(" | ")" | " "
| "_" | "@" | "#" | "\\" | "+" | "-" | "." | "," | ":" | "$" | "^" | "*" )+ "\""
<ClaimLiteral> ::= <StringLiteral>
<ReferenceLiteral> ::= <StringLiteral>
<RouteLiteral> ::= <StringLiteral>
<IdentifiableLiteral> ::= <StringLiteral>
<ReferableLiteral> ::= <StringLiteral>
<FragmentLiteral> ::= <StringLiteral>
<DescriptorLiteral> ::= <StringLiteral>
<NumericalLiteral> ::= ( "+" | "-" )? ( [0-9]+ ( "." [0-9]* )? | "." [0-9]+ ) ( ( "e" | "E"
)? [0-9]+ )
<HexLiteral> ::= "16#" ( [0-9] | [A-F] )+
<BoolLiteral> ::= "true" | "false"
<FieldIdentifier> ::= <FieldIdentifierString>
<FieldIdentifierString> ::= <FieldIdentifierAAS> | <FieldIdentifierSM> |
<FieldIdentifierSME> | <FieldIdentifierCD> | <FieldIdentifierAasDescriptor> |
<FieldIdentifierSmDescriptor>
<FieldIdentifierAAS> ::= "$aas#" ( "idShort" | "id" | "assetInformation.assetKind" |
"assetInformation.assetType" | "assetInformation.globalAssetId" | "assetInformation."
<SpecificAssetIdsClause> | "submodels." <ReferenceClause> )
<FieldIdentifierSM> ::= "$sm#" ( <SemanticIdClause> | "idShort" | "id" )
<FieldIdentifierSME> ::= "$sme" ( "." <idShortPath> )? "#" ( <SemanticIdClause> | "idShort"
| "value" | "valueType" | "language" )
<FieldIdentifierCD> ::= "$cd#" ( "idShort" | "id" ) <ws>
<FieldIdentifierAasDescriptor> ::= "$aasdesc#" ( "idShort" | "id" | "assetKind" |
"assetType" | "globalAssetId" | <SpecificAssetIdsClause>  | "endpoints" ( "[" ( [0-9]* )
"]" ) "." <EndpointClause> | "submodelDescriptors" ( "[" ( [0-9]* ) "]" ) "."
<SmDescriptorClause> )
<FieldIdentifierSmDescriptor> ::= "$smdesc#" <SmDescriptorClause>
<SmDescriptorClause> ::= ( <SemanticIdClause> | "idShort" | "id" | "endpoints" ( "[" ( [0-
9]* ) "]" ) "." <EndpointClause> )
<EndpointClause> ::= "interface" | "protocolinformation.href"

<ReferenceClause> ::= ( "type" | "keys" ( "[" ( [0-9]* ) "]" ) ( ".type" | ".value" ) )
<SemanticIdClause> ::= ( "semanticId" | "semanticId." <ReferenceClause> )
<SpecificAssetIdsClause> ::=  ( "specificAssetIds" ( "[" ( [0-9]* ) "]" ) ( ".name" |
".value" | ".externalSubjectId" | ".externalSubjectId." <ReferenceClause> ) )
<idShortPath> ::= ( <idShort> ("[" ( [0-9]* ) "]" )? ( "." <idShortPath> )* )
<idShort> ::= ( ( [a-z] | [A-Z] ) ( [a-z] | [A-Z] | [0-9] | "_" )* )

<ws> ::= ( " " | "\t" | "\r" | "\n" )+
```

Select Expression

The default return type is a list containing the respective AAS objects. For an AAS Repository, it's a list of Asset Administration Shells, while a Submodel Registry returns a list of SubmodelDescriptors. The optional $select statement declares whether the response shall deviate from this default behavior, and only return a list of identifiers instead of a list of objects ($select id).

Identification of Fields

Elements relevant for comparisons are described using an adapted IdShortPath notation. The declaration starts with kind of element that shall serve as the root of the expression, followed with an optional '.' as the separator symbol and an optional IdShortPath, and a mandatory declaration of the AAS attribute that shall be examined, separated via '#'. A recursive search is made through the Submodel Elements, when the optional IdShortPath is left out.

```
FieldIdentifier ::= <RootDeclaration> ( "." <IdShortPath> | "" ) "#" <AttributeDeclaration>
```

Constraint AASa-005: Only the _SubmodelElements_ root delaration can be followed with IdShortPaths.

Table 1. Root Elements for Field Identifiers

Root Element	Definition	Example(s)
$aas	Starting point for fields available in Asset Administration Shells	$aas#assetInformation.assetKind
$sm	Starting point for fields available in Submodels	$sm#semanticId.keys[0].value
$sme	Starting point for fields available in Submodel Elements. Can be followed with an IdShortPath, see Constraint AASa-005. Can start at any possible Submodel Element, is **not** restricted to Submodel Elements directly in the Submodel/submodelElements list.	$sme.smeCollectionIdShort.propertyIdShort#value, $sme#value to search recursively
$cd	Starting point for fields available in Concept Descriptions	$cd#id
$aasdesc	Starting point for fields available in Asset Administration Shell Descriptors	$aasdesc#submodelDescriptors[0].endpoints[0].protocolinformation.href
$smdesc	Starting point for fields available in Submodel Descriptors	$smdesc#endpoints[0].protocolinformation.href

Attribute declarations point to literal values that provide the input for comparisons. It is not possible to point to objects or lists, only atomic values. Attribute declarations present a subset of the attributes defined by the AAS Metamodel and the extension classes of Clause Data Types for Payload.

Table 2. Attribute Elements for Field Identifiers. <index> *is an optional nonNegativeInteger value. No* <index> *shall be interpreted as 'anywhere in the list'.*

Root Element	Definition	Example
id	Identifier, e.g., of an AAS, Submodel, or Condept Description	$aas#id
idShort	Value of the idShort attribute	$aas#id
assetInformation.assetKind	Value of the assetKind attribute of an AAS	$aas#assetInformation.assetKind

assetInformation.assetType	Value of the assetKind attribute of an AAS	$aas#assetInformation.assetType
assetInformation.globalAssetId	Value of the globalAssetId attribute of an AAS	$aas#assetInformation.globalAssetId
assetInformation.assetInformation.specificAssetIds[<index>].name	Name of a SpecificAssetId of an AAS	$aas#assetInformation.assetInformation. specificAssetIds[].name
assetInformation.assetInformation.specificAssetIds[<index>].value	Value of a SpecificAssetId of an AAS	$aas#assetInformation.assetInformation. specificAssetIds[1].value
assetInformation.assetInformation.specificAssetIds[<index>].externalSubjectId. type	Type of a Reference used as an externalSubjectId in a SpecificAssetId of an AAS	$aas#assetInformation.assetInformation. specificAssetIds[0]. externalSubjectId.type
assetInformation.assetInformation.specificAssetIds[<index>].externalSubjectId. keys[<index>].value	Value of a key of a Reference used as an externalSubjectId in a SpecificAssetId of an AAS	$aas#assetInformation.assetInformation.specificAssetIds[0].externalSubjectId. keys[].value
assetInformation.assetInformation.specificAssetIds[<index>].externalSubjectId. keys[<index>].type	Type of a key of a Reference used as an externalSubjectId in a SpecificAssetId of an AAS	$aas#assetInformation.assetInformation.specificAssetIds[0].externalSubjectId. keys[].type
submodels	Shortcut for Submodels referenced by an AAS, see Clause References	$aas#submodels
submodels.type	Type of a Reference that associates an AAS with a Submodel	$aas#submodels.type
submodels.keys[<index>].value	Value of a key used in a Reference that associates an AAS with a Submodel	$aas#submodels.keys[].value
submodels.keys[<index>].type	Value of a key used in a Reference that associates an AAS with a Submodel	$aas#submodels.keys[0].type
semanticId	Shortcut for semanticIds, see Clause References	$sm#semanticId
semanticId.type	ReferenceType of a semanticId Reference	$sm#semanticId.type
semanticId.keys[<index>].type	KeyType of a semanticId Reference	$sm#semanticId.keys[].type
semanticId.keys[<index>].value	Value of a key of a semanticId Reference	$sme#semanticId.keys[].value
value	Value of a Submodel Element	$sme#value

valueType	ValueType of a Submodel Element	$sme.someIdShort#valueType
language	Language of a Multilanguage Property	$sme#language
assetKind	Value of the assetKind attribute of an AAS Descriptor	$aasdesc#assetKind
assetType	Value of the assetKind attribute of an AAS Descriptor	$aasdesc#assetType
globalAssetId	Value of the globalAssetId attribute of an AAS Descriptor	$aasdesc#globalAssetId
specificAssetIds[<index>].name	Name of a SpecificAssetId of an AAS Descriptor	$aasdesc#specificAssetIds[].name
specificAssetIds[<index>].value	Value of a SpecificAssetId of an AAS Descriptor	$aasdesc#specificAssetIds[1].value
specificAssetIds[<index>].externalSubjectId.type	Type of a Reference used as an externalSubjectId in a SpecificAssetId of an AAS Descriptor	$aasdesc#specificAssetIds[<index>]. externalSubjectId.type
specificAssetIds[<index>].externalSubjectId.keys[<index>].value	Value of a key of a Reference used as an externalSubjectId in a SpecificAssetId of an AAS Descriptor	$aasdesc#specificAssetIds[0].externalSubjectId.keys[].value
specificAssetIds[<index>].externalSubjectId.keys[<index>].type	Type of a key of a Reference used as an externalSubjectId in a SpecificAssetId of an AAS Descriptor	$aasdesc#specificAssetIds[0].externalSubjectId.keys[].type
endpoints[<index>].interface	Interface of an endpoint of an AAS or Submodel Note: Can only be used with Asset Administration Shell Descriptors or Submodel Descriptors.	$aasdesc#endpoints[0].interface
endpoints[<index>].protocolinformation.href	Href of an endpoint of an AAS or Submodel Note: Can only be used with Asset Administration Shell Descriptors or Submodel Descriptors.	$smdesc#endpoints[0].protocolinformation.href

submodelDescriptors.semanticId	Shortcut for semanticIds, see Clause References	$aasdesc#submodelDescriptors.semanticId
submodelDescriptors.semanticId.type	ReferenceType of a semanticId Reference used in a referenced Submodel of an Asset Administration Shell Descriptor	$aasdesc#submodelDescriptors.semanticId.type
submodelDescriptors.semanticId.keys[<index>].type	KeyType of a semanticId Reference used in a referenced Submodel of an Asset Administration Shell Descriptor	$aasdesc#submodelDescriptors.semanticId. keys[].type
submodelDescriptors.semanticId.keys[<index>].value	Value of a key of a semanticId Reference used in a referenced Submodel of an Asset Administration Shell Descriptor	$aasdesc#submodelDescriptors.semanticId. keys[].value
submodelDescriptors.id	Identifier of a referenced Submodel as available in an Asset Administration Shell Descriptor	$aasdesc#submodelDescriptors.id
submodelDescriptors.idShort	idShort of a referenced Submodel as available in an Asset Administration Shell Descriptor	$aasdesc#submodelDescriptors.idShort
submodelDescriptors.endpoints[<index>].interface	Endpoint interface of a referenced Submodel as available in an Asset Administration Shell Descriptor	$aasdesc#submodelDescriptors.endpoints[0]. interface
submodelDescriptors.endpoints[<index>].protocolinformation.href	Endpoint href of a referenced Submodel as available in an Asset Administration Shell Descriptor	$aasdesc#submodelDescriptors.endpoints[]. protocolinformation.href

Comparison Operators

The following comparison operators are part of the query language. The result of a comparison shall always be (a) of type xs:boolean or (b) a comparison error, e.g., due to invalid inputs.
In case of an error the comparison result is invalid. Any further combinations with the invalid result will also have an invalid result. In that case all condition expressions will be ignored for the evaluated element and no result will be returned. Only the elements with valid results will be returned.

The comparisons $eq $ne $gt $lt $ge $le are overloaded in the BNF grammar, which means that the same comparison can deal with several input types. For instance, $eq can be used both for values of type xs:string and xs:int. The comparisons $contains $starts-with $ends-with $regex only allow xs:string.

Operator	Description	Definition
$eq	Compares two values if they are identical.	Operator 'A eq B' in https://www.w3.org/TR/xpath-30/#mapping

$ne	Compares two values if they are not identical.	Operator 'A eq B' in https://www.w3.org/TR/xpath-30/#mapping
$gt	Checks whether one parameter is greater than another.	Operator 'A gt B' in https://www.w3.org/TR/xpath-30/#mapping
$lt	Checks whether one parameter is lower than another.	Operator 'A lt B' in https://www.w3.org/TR/xpath-30/#mapping
$ge	Checks whether one parameter is greater or equal than another.	Operator 'A ge B' in https://www.w3.org/TR/xpath-30/#mapping
$le	Checks whether one parameter is lower or equal than another.	Operator 'A ne B' in https://www.w3.org/TR/xpath-30/#mapping
$contains	Compares two string expressions whether the second parameter appears as a substring inside of the first.	Defined as fn:contains in https://www.w3.org/TR/xpath-functions-30/#func-contains
$starts-with	Compares two string expressions whether the second parameter appears character-equal at the beginning of the first.	Defined as fn:starts-with in https://www.w3.org/TR/xpath-functions-30/#func-starts-with
$ends-with	Compares two string expressions whether the second parameter appears character-equal at the end of the first.	Defined as fn:starts-with in https://www.w3.org/TR/xpath-functions-30/#func-ends-with
$regex	Compares two string expressions whether the first parameter matches the regex of the second.	Defined as fn:starts-with in https://www.w3.org/TR/xpath-functions-30/#func-matches

Example

The following example is used to illustrate the comparisons. The following Asset Administration Shell is used for the evaluations of the different operators.

```
{
    "modelType": "AssetAdministrationShell",
    "id": "https://example.com/asset-administration-shell-1",
    "assetInformation": {
        "assetKind": "Instance",
        "globalAssetId": "urn:asset-administration-shell-1",
        "specificAssetIds": [
            {
                "name": "supplierId",
                "value": "aas-1"
            },
            {
                "name": "customerId",
                "value": "aas-2"
            }
        ],
    },
```

```
    "submodels": [
        {
            "type": "ModelReference",
            "keys": [
                {
                    "type": "Submodel",
                    "value": "https://example.com/submodel-1"
                }
            ]
        },
        {
            "type": "ModelReference",
            "keys": [
                {
                    "type": "Submodel",
                    "value": "https://example.com/submodel-2"
                }
            ]
        }
    ]
}
```

Comparison	Result	Comment
$aas#idShort $eq $aas#assetInformation.assetType	true	Both items are empty, therefore, they are equal.
$aas#idShort $le $aas#assetInformation.assetType	true	$eq implies $le
$aas#idShort $ne $aas#assetInformation.assetType	false	Both items are empty, therefore, they are equal.
1 $le 2	true	Numeric comparison
1 $gt 2	false	Numeric comparison
13 $eq '13'	false	Type mismatch
"a" $lt "b"	true	String comparison
"1" $gt "2"	false	String comparison
"11" $gt "2"	false	String comparison is executed character-wise: The first character of the left parameter ("1") comes before the first character of the right parameter ("2").

$aas#assetInformation.assetKind $eq $aas#submodels	false	$aas#submodels returns the strings https://example.com/submodel-1 and https://example.com/submodel-2, neither is equal to the value of the assetKind = Instance.
$aas#assetInformation.assetKind $ne $aas#submodels	true	$aas#submodels returns the strings https://example.com/submodel-1 and https://example.com/submodel-2, neither is equal to the value of the assetKind = Instance.
$aas#assetInformation.assetKind $eq $aas#assetInformation.assetKind	true	Comparison of identical values
$aas#assetInformation.assetKind $ne $aas#assetInformation.assetKind	false	Comparison of identical values
$aas#submodels $eq $aas#submodels	true	By definition.
$aas#assetInformation.assetKind $eq 17	false	17 is implicetly casted to the string "17", however, this is different to the value of the first paramter "Instance".
$aas#assetInformation.assetKind $ne 17	true	17 is implicetly casted to the string "17", which is not equal to the value of the first paramter "Instance".
$aas#assetInformation.assetKind $le $aas#assetInformation.assetKind	true	$eq implies $le
bool("true") $ge bool("true")	true	$eq implies $ge
bool("true") $gt bool("true")	false	Booleans do not offer $lt/$gt comparison
$aas#id $contains "https://example.com/asset-administration"	true	"https://example.com/asset-administration" is a substring of the id value ("https://example.com/asset-administration-shell-1")

Logical Expressions

Logical expressions allow the combination of two or more single comparisons through AND or OR relations, and to negate the result of an expression. Furthermore, logical expressions can also be used to combine other logical expressions. Combinations with invalid results will be ignored, as explained above.

Logical Operator	Description	Definition
$and	Connects two or more expressions through a logical AND.	Defined by https://www.w3.org/TR/xpath-30/#doc-xpath30-AndExpr

$or	Connects two or more expressions through a logical OR.	Defined by https://www.w3.org/TR/xpath-30/#doc-xpath30-OrExpr
$not	Negates an expression.	The "not" operator inverts the truth value of its operand. If the operand is true, the result is false, and if the operand is false, the result is true. Defined by https://www.w3.org/TR/xpath-functions-30/#func-not

Paranthesises

Paranthesises ( ) allow to combine logical expressions to define precedence. LogicalExpressions with paranthesises can be nested.

Casting

For explicite casting the following casting operators are used:

Casting Operator	Description	Definition
$str(<value>)	Casts the value to xs:string.	Defined by https://www.w3.org/TR/xpath-functions-30/#func-string
$num(<value>)	Casts the value to xs:integer.	Defined by https://www.w3.org/TR/xpath-functions-30/#func-number
$dateTime(<value>)	Casts the value to xs:dateTime.	Defined by https://www.w3.org/TR/xpath-functions-30/#func-dateTime
$bool(<value>)	Casts the value to xs:boolean.	Defined by https://www.w3.org/TR/xpath-functions-30/#func-boolean
$hex(<value>)	Casts the value to xs:hexBinary.	Defined by https://www.w3.org/TR/xpath-functions/#casting-to-binary
$dateTime(<value>)	Casts the value to xs:dateTime.	Defined by https://www.w3.org/TR/xpath-functions/#casting-to-datetimes
$time(<value>)	Casts the value to xs:time.	Defined by https://www.w3.org/TR/xpath-functions/#casting-to-datetimes

Implicite casting is used together with FieldIdentifiers. FieldIdentifiers are generally treated as xs:string in the query language.
If a FieldIdentifier is used in a logicalExpression, it will be implicitely casted to xs:boolean, which can only create a valid result for the values true and false.
If a FieldIdentifier is used in a comparison, the second parameter decides implicite casting. If the second parameter is a constant (string, number, hex, boolean, dateTime, time) or a corresponding explicite casting operator, the value of the FieldIdentifier will be implicitely casted to the corresponding data type.

Referring to Elements in Lists and Arrays

The AAS Metamodel defines several objects with lists as child elements, e.g., SubmodelElementList/value, AssetInformation/specificAssetIds, or Reference/keys. The AAS Query Language contains two different patterns to refer to elements inside these lists:

Specific Index Notation

In case the position inside the list is known, the query can directly leverage the index number inside square brackets.

```
<fieldIdentifier>[<index>]
```

For the first element of a SubmodelElementList with the idShort "smlIdShort", this notation may lead to a following declaration:

```
$sme.<someIdShortPath>.smlIdShort[0]
```

For the first key of a semanticId reference, this notation may lead to a following declaration:

```
$sme.<someIdShortPath>#semanticId.keys[0]
```

Any Element Notation

In case the position in the list is not known, the index inside the square brackets can be skipped. This means that at least one element has to fulfill the comparison to make the expression evaluate to true.

```
<fieldIdentifier>[]
```

To refer to any element inside a SubmodelElementList with the idShort "smlIdShort", this notation may lead to a following declaration:

```
$sme.<somePath>.smlIdShort[]
```

For any key of a semanticId reference, this notation may lead to a following declaration:

```
$sme.<somePath>#semanticId.keys[]
```

References

References include a list of keys, i.e. .keys[]. Very often the value of the first key in the list is needed, e.g. for semanticId.
To ease writing, .keys[0].value can be left off for References.

semanticId is defined as the "value" of the first key of the semanticId Reference object. The following two expressions are equivilant:

```
<somePath>.semanticId
<somePath>.semanticId.keys[0].value
```

A comparison using the semanticId as a shortcut could look like the following:

```
$sme#semanticId $eq "https://example.com/a/semantic/id"
```

Match of Elements in Lists

The $match operator signals that the following clauses (a) contain at least 1 list of elements with [] syntax, and that (b) all conditions shall be evaluated on the same element of this list. The common path prefix left to a [] determines the list to be searched.

FieldIdentifiers in a $match may contain several [], which define nested lists. In such case the path prefix of the first [] must always be the same and the path prefixes of the following [] accordingly.

$match is similar to $and, since all conditions inside must result to true. But as explained above, [] can be specified in addition.
$match can include $match and comparisions. logicalExpressions are not allowed inside $match.
$match inside $match further restricts the subset of possible elements inside nested [].

> Note 1: If the comparisons inside an $match clause contains expressions that are not pointing to the list under consideration, an error shall be returned.

The table illustrates the behavior using the example Asset Administration Shell above.

Comparison	Result	Comment

`$match(`
 	
`$aas#assetInformation.specificAssetIds[].name`
`  $eq "supplierId",`
 	
`$aas#assetInformation.specificAssetIds[].value`
`  $eq "aas-1"`
`)`
	`[`
	`  {`
	`  "modelType": "AssetAdministrationShell",`
	`  "id": "https://example.com/asset-administration-shell-1",`
	`  ...`
	`  }`
	`]`
		The values for 'supplierId' and 'aas-1' exist in the same SpecificAssetId object.

`$match(`
 	
`$aas#assetInformation.specificAssetIds[].name`
`  $eq "supplierId",`
 	
`$aas#assetInformation.specificAssetIds[].value`
`  $eq "aas-2"`
`)`
	`[]`	The values for 'supplierId' and 'aas-2' do not exist in the same SpecificAssetId object.

`$and (`
 	
`$aas#assetInformation.specificAssetIds[].name`
`  $eq "supplierId",`
 	
`$aas#assetInformation.specificAssetIds[].value`
`  $eq "aas-2"`
`)`
	`[`
	`  {`
	`  "modelType": "AssetAdministrationShell",`
	`  "id": "https://example.com/asset-administration-shell-1",`
	`  ...`
	`  }`
	`]`
		The values for 'supplierId' and 'aas-2' exist in the example AAS, even though in *different* SpecificAssetId objects. As no $match enforces the evaluation on the same SpecificAssetId, the AAS from the example fulfils the condition.

`$or (`
`  $match (`
 	
`$aas#assetInformation.specificAssetIds[]`
`  .name  $eq  "supplierId",`
 	
`$aas#assetInformation.specificAssetIds[]`
`  .value  $eq  "aas-1"`
`  ),`
`  $match (`
 	
`$aas#assetInformation.specificAssetIds[]`
`  .name  $eq  "customerId",`
 	
`$aas#assetInformation.specificAssetIds[]`
`  .value  $eq  "aas-2"`
`  )`
`)`
	`[`
	`  {`
	`  "modelType": "AssetAdministrationShell",`
	`  "id": "https://example.com/asset-administration-shell-1",`
	`  ...`
	`  }`
	`]`
		Both $match conditions are fulfilled, even though for different SpecificAssetIds. Therefore the OR clause also evaluates to true for both items.

Sorting and Pagination

The AAS Query Language does not introduce additional functionalities to control the pagination or sorting of the result sets. The general capabilities available for Operations apply as well for queries. See for instance Pagination for the pagination mechanism for the HTTP APIs, which also define the pagination and sorting behavior for AAS queries that are exchanged via HTTP.

JSON Schema

The AAS HTTP API represents AAS Queries as JSON objects. A JSON schema according to the grammar above has been defined. This JSON schema allows immediate validation of queries but also automatic code generation which has been validated with jsonschema2pojo. To allow such automatic code generation, polymorhism by the use of oneOf in the JSON schema must be avoided.
In addition, several constructs introduced in BNF form require a slightly differing representation. For instance, a basic comparison follows the pattern <operand> <operator> <operand> but appears as "<operator>": [<operand>, <operand>] in the JSON representation. JSON schema allows to use regular expressions. Such regular expressions are used to validate the syntax of FieldIdentifiers. To be able to use common JSON schema validators, the depth and details have been limited in the corresponding regular expressions.

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Common JSON Schema for AAS Queries and Access Rules",
  "description": "This schema contains all classes that are shared between the AAS Query
Language and the AAS Access Rule Language.",
  "definitions": {
    "standardString": {
      "type": "string",
      "pattern": "^(?!\\$).*"
    },
    "modelStringPattern": {
      "type": "string",
      "pattern":
"^((?:\\$aas#(?:idShort|id|assetInformation\\.assetKind|assetInformation\\.assetType|assetI
nformation\\.globalAssetId|assetInformation\\.(?:specificAssetIds(\\[[0-
9]*\\])(?:\\.(?:name|value|externalSubjectId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(?:type|value
))?)?)?)|submodels\\.(?:type|keys\\[\\d*\\](?:\\.(?:type|value))?))|submodels\\.(type|keys\
\[\\d*\\](?:\\.(type|value))?)))|(?:\\$sm#(?:semanticId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(t
ype|value))?)?|idShort|id))|(?:\\$sme(?:\\.[a-zA-Z][a-zA-Z0-9_]*(\\[[0-9]*\\])?(?:\\.[a-zA-
Z][a-zA-Z0-9_]*(\\[[0-
9]*\\])?)*)?#(?:semanticId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(type|value))?)?|idShort|value|
valueType|language))|(?:\\$cd#(?:idShort|id))|(?:\\$aasdesc#(?:idShort|id|assetKind|assetTy
pe|globalAssetId|specificAssetIds(\\[[0-
9]*\\])?(?:\\.(name|value|externalSubjectId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(type|value))?
)?)?)|endpoints(\\[[0-
9]*\\])\\.(interface|protocolinformation\\.href)|submodelDescriptors(\\[[0-
9]*\\])\\.(semanticId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(type|value))?)?|idShort|id|endpoint
s(\\[[0-
9]*\\])\\.(interface|protocolinformation\\.href))))|(?:\\$smdesc#(?:semanticId(?:\\.type|\\
.keys\\[\\d*\\](?:\\.(type|value))?)?|idShort|id|endpoints(\\[[0-
9]*\\])\\.(interface|protocolinformation\\.href))))$"
    },
    "hexLiteralPattern": {
      "type": "string",
      "pattern": "^16#[0-9A-F]+$"
    },
    "dateTimeLiteralPattern": {
      "type": "string",
      "format": "date-time"
    },
    "timeLiteralPattern": {
      "type": "string",
      "pattern": "^[0-9][0-9]:[0-9][0-9](:[0-9][0-9])?$"
    },
    "Value": {
      "type": "object",
      "properties": {
        "$field": {
          "$ref": "#/definitions/modelStringPattern"
        },
        "$strVal": {
```

```
        "$ref": "#/definitions/standardString"
      },
      "$attribute": {
        "$ref": "#/definitions/attributeItem"
      },
      "$numVal": {
        "type": "number"
      },
      "$hexVal": {
        "$ref": "#/definitions/hexLiteralPattern"
      },
      "$dateTimeVal": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      },
      "$timeVal": {
        "$ref": "#/definitions/timeLiteralPattern"
      },
      "$boolean": {
        "type": "boolean"
      },
      "$strCast": {
        "$ref": "#/definitions/Value"
      },
      "$numCast": {
        "$ref": "#/definitions/Value"
      },
      "$hexCast": {
        "$ref": "#/definitions/Value"
      },
      "$boolCast": {
        "$ref": "#/definitions/Value"
      },
      "$dateTimeCast": {
        "$ref": "#/definitions/Value"
      },
      "$timeCast": {
        "$ref": "#/definitions/Value"
      },
      "$dayOfWeek": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      },
      "$dayOfMonth": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      },
      "$month": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      },
      "$year": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      }
    },
    "oneOf": [
```

```
    {
      "required": [
        "$field"
      ]
    },
    {
      "required": [
        "$strVal"
      ]
    },
    {
      "required": [
        "$attribute"
      ]
    },
    {
      "required": [
        "$numVal"
      ]
    },
    {
      "required": [
        "$hexVal"
      ]
    },
    {
      "required": [
        "$dateTimeVal"
      ]
    },
    {
      "required": [
        "$timeVal"
      ]
    },
    {
      "required": [
        "$boolean"
      ]
    },
    {
      "required": [
        "$strCast"
      ]
    },
    {
      "required": [
        "$numCast"
      ]
    },
    {
      "required": [
```

```
          "$hexCast"
        ]
      },
      {
        "required": [
          "$boolCast"
        ]
      },
      {
        "required": [
          "$dateTimeCast"
        ]
      },
      {
        "required": [
          "$timeCast"
        ]
      },
      {
        "required": [
          "$dayOfWeek"
        ]
      },
      {
        "required": [
          "$dayOfMonth"
        ]
      },
      {
        "required": [
          "$month"
        ]
      },
      {
        "required": [
          "$year"
        ]
      }
    ],
    "additionalProperties": false
  },
  "stringValue": {
    "type": "object",
    "properties": {
      "$field": {
        "$ref": "#/definitions/modelStringPattern"
      },
      "$strVal": {
        "$ref": "#/definitions/standardString"
      },
      "$strCast": {
        "$ref": "#/definitions/Value"
```

```
      },
      "$attribute": {
        "$ref": "#/definitions/attributeItem"
      }
    },
    "oneOf": [
      {
        "required": [
          "$field"
        ]
      },
      {
        "required": [
          "$strVal"
        ]
      },
      {
        "required": [
          "$strCast"
        ]
      },
      {
        "required": [
          "$attribute"
        ]
      }
    ],
    "additionalProperties": false
  },
  "comparisonItems": {
    "type": "array",
    "minItems": 2,
    "maxItems": 2,
    "items": {
      "$ref": "#/definitions/Value"
    }
  },
  "stringItems": {
    "type": "array",
    "minItems": 2,
    "maxItems": 2,
    "items": {
      "$ref": "#/definitions/stringValue"
    }
  },
  "matchExpression": {
    "type": "object",
    "properties": {
      "$match": {
        "type": "array",
        "minItems": 1,
        "items": {
```

```
          "$ref": "#/definitions/matchExpression"
        }
      },
      "$eq": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$ne": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$gt": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$ge": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$lt": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$le": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$contains": {
        "$ref": "#/definitions/stringItems"
      },
      "$starts-with": {
        "$ref": "#/definitions/stringItems"
      },
      "$ends-with": {
        "$ref": "#/definitions/stringItems"
      },
      "$regex": {
        "$ref": "#/definitions/stringItems"
      },
      "$boolean": {
        "type": "boolean"
      }
    },
    "oneOf": [
      {
        "required": [
          "$eq"
        ]
      },
      {
        "required": [
          "$ne"
        ]
      },
      {
        "required": [
          "$gt"
        ]
```

```
      },
      {
        "required": [
          "$ge"
        ]
      },
      {
        "required": [
          "$lt"
        ]
      },
      {
        "required": [
          "$le"
        ]
      },
      {
        "required": [
          "$contains"
        ]
      },
      {
        "required": [
          "$starts-with"
        ]
      },
      {
        "required": [
          "$ends-with"
        ]
      },
      {
        "required": [
          "$regex"
        ]
      },
      {
        "required": [
          "$boolean"
        ]
      },
      {
        "required": [
          "$match"
        ]
      }
    ],
    "additionalProperties": false
  },
  "logicalExpression": {
    "type": "object",
    "properties": {
```

```
        "$and": {
          "type": "array",
          "minItems": 2,
          "items": {
            "$ref": "#/definitions/logicalExpression"
          }
        },
        "$match": {
          "type": "array",
          "minItems": 1,
          "items": {
            "$ref": "#/definitions/matchExpression"
          }
        },
        "$or": {
          "type": "array",
          "minItems": 2,
          "items": {
            "$ref": "#/definitions/logicalExpression"
          }
        },
        "$not": {
          "$ref": "#/definitions/logicalExpression"
        },
        "$eq": {
          "$ref": "#/definitions/comparisonItems"
        },
        "$ne": {
          "$ref": "#/definitions/comparisonItems"
        },
        "$gt": {
          "$ref": "#/definitions/comparisonItems"
        },
        "$ge": {
          "$ref": "#/definitions/comparisonItems"
        },
        "$lt": {
          "$ref": "#/definitions/comparisonItems"
        },
        "$le": {
          "$ref": "#/definitions/comparisonItems"
        },
        "$contains": {
          "$ref": "#/definitions/stringItems"
        },
        "$starts-with": {
          "$ref": "#/definitions/stringItems"
        },
        "$ends-with": {
          "$ref": "#/definitions/stringItems"
        },
        "$regex": {
```

```
        "$ref": "#/definitions/stringItems"
      },
      "$boolean": {
        "type": "boolean"
      }
    },
    "oneOf": [
      {
        "required": [
          "$and"
        ]
      },
      {
        "required": [
          "$or"
        ]
      },
      {
        "required": [
          "$not"
        ]
      },
      {
        "required": [
          "$eq"
        ]
      },
      {
        "required": [
          "$ne"
        ]
      },
      {
        "required": [
          "$gt"
        ]
      },
      {
        "required": [
          "$ge"
        ]
      },
      {
        "required": [
          "$lt"
        ]
      },
      {
        "required": [
          "$le"
        ]
      },
```

```
        {
          "required": [
            "$contains"
          ]
        },
        {
          "required": [
            "$starts-with"
          ]
        },
        {
          "required": [
            "$ends-with"
          ]
        },
        {
          "required": [
            "$regex"
          ]
        },
        {
          "required": [
            "$boolean"
          ]
        },
        {
          "required": [
            "$match"
          ]
        }
      ],
      "additionalProperties": false
    },
    "attributeItem": {
      "oneOf": [
        {
          "required": [
            "CLAIM"
          ]
        },
        {
          "required": [
            "GLOBAL"
          ]
        },
        {
          "required": [
            "REFERENCE"
          ]
        }
      ],
      "properties": {
```

```
        "CLAIM": {
          "type": "string"
        },
        "GLOBAL": {
          "type": "string",
          "enum": [
            "LOCALNOW",
            "UTCNOW",
            "CLIENTNOW",
            "ANONYMOUS"
          ]
        },
        "REFERENCE": {
          "type": "string"
        }
      },
      "additionalProperties": false
    },
    "objectItem": {
      "oneOf": [
        {
          "required": [
            "ROUTE"
          ]
        },
        {
          "required": [
            "IDENTIFIABLE"
          ]
        },
        {
          "required": [
            "REFERABLE"
          ]
        },
        {
          "required": [
            "FRAGMENT"
          ]
        },
        {
          "required": [
            "DESCRIPTOR"
          ]
        }
      ],
      "properties": {
        "ROUTE": {
          "type": "string"
        },
        "IDENTIFIABLE": {
          "type": "string"
```

```
        },
        "REFERABLE": {
          "type": "string"
        },
        "FRAGMENT": {
          "type": "string"
        },
        "DESCRIPTOR": {
          "type": "string"
        }
      },
      "additionalProperties": false
    },
    "rightsEnum": {
      "type": "string",
      "enum": [
        "CREATE",
        "READ",
        "UPDATE",
        "DELETE",
        "EXECUTE",
        "VIEW",
        "ALL",
        "TREE"
      ],
      "additionalProperties": false
    },
    "ACL": {
      "type": "object",
      "properties": {
        "ATTRIBUTES": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/attributeItem"
          }
        },
        "USEATTRIBUTES": {
          "type": "string"
        },
        "RIGHTS": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/rightsEnum"
          }
        },
        "ACCESS": {
          "type": "string",
          "enum": [
            "ALLOW",
            "DISABLED"
          ]
        }
```

```
    },
    "required": [
      "RIGHTS",
      "ACCESS"
    ],
    "oneOf": [
      {
        "required": [
          "ATTRIBUTES"
        ]
      },
      {
        "required": [
          "USEATTRIBUTES"
        ]
      }
    ],
    "additionalProperties": false
  },
  "AccessPermissionRule": {
    "type": "object",
    "properties": {
      "ACL": {
        "$ref": "#/definitions/ACL"
      },
      "USEACL": {
        "type": "string"
      },
      "OBJECTS": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/objectItem"
        },
        "additionalProperties": false
      },
      "USEOBJECTS": {
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "FORMULA": {
        "$ref": "#/definitions/logicalExpression",
        "additionalProperties": false
      },
      "USEFORMULA": {
        "type": "string"
      },
      "FRAGMENT": {
        "type": "string"
      },
      "FILTER": {
```

```
          "$ref": "#/definitions/logicalExpression",
          "additionalProperties": false
        },
        "USEFILTER": {
          "type": "string"
        }
      },
      "oneOf": [
        {
          "required": [
            "ACL"
          ]
        },
        {
          "required": [
            "USEACL"
          ]
        }
      ],
      "oneOf": [
        {
          "required": [
            "OBJECTS"
          ]
        },
        {
          "required": [
            "USEOBJECTS"
          ]
        }
      ],
      "oneOf": [
        {
          "required": [
            "FORMULA"
          ]
        },
        {
          "required": [
            "USEFORMULA"
          ]
        }
      ],
      "additionalProperties": false
    }
  },
  "type": "object",
  "properties": {
    "Query": {
      "type": "object",
      "properties": {
        "$select": {
```

```
        "type": "string",
        "pattern": "^id$"
      },
      "$condition": {
        "$ref": "#/definitions/logicalExpression"
      }
    },
    "required": [
      "$condition"
    ],
    "additionalProperties": false
  },
  "AllAccessPermissionRules": {
    "type": "object",
    "properties": {
      "DEFATTRIBUTES": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "name": {
              "type": "string"
            },
            "attributes": {
              "type": "array",
              "items": {
                "$ref": "#/definitions/attributeItem"
              }
            }
          },
          "required": [
            "name",
            "attributes"
          ],
          "additionalProperties": false
        }
      },
      "DEFACLS": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "name": {
              "type": "string"
            },
            "acl": {
              "$ref": "#/definitions/ACL"
            }
          },
          "required": [
            "name",
            "acl"
```

```
        ],
        "additionalProperties": false
      }
    },
    "DEFOBJECTS": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "name": {
            "type": "string"
          },
          "objects": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/objectItem"
            }
          },
          "USEOBJECTS": {
            "type": "array",
            "items": {
              "type": "string"
            }
          }
        },
        "required": [
          "name"
        ],
        "oneOf": [
          {
            "required": [
              "objects"
            ]
          },
          {
            "required": [
              "USEOBJECTS"
            ]
          }
        ],
        "additionalProperties": false
      }
    },
    "DEFFORMULAS": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "name": {
            "type": "string"
          },
          "formula": {
```

```
                "$ref": "#/definitions/logicalExpression"
              }
            },
            "required": [
              "name",
              "formula"
            ],
            "additionalProperties": false
          }
        },
        "rules": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/AccessPermissionRule"
          }
        }
      },
      "required": [
        "rules"
      ],
      "additionalProperties": false
    }
  },
  "oneOf": [
    {
      "required": [
        "Query"
      ]
    },
    {
      "required": [
        "AllAccessPermissionRules"
      ]
    }
  ],
  "additionalProperties": false
}
```

The table below explains the mapping between the grammar and the JSON schema. The table follows the structure of the JSON schema top to bottom.

JSON Schema	Grammar	Comment
modelStringPattern	<FieldIdentifier>	AAS model elements are strings which start with a $
standardString		All other value strings
hexLiteralPattern	<HexLiteral>	
dateTimeLiteralPattern	<DateTimeLiteral>	

timeLiteralPattern	<TimeLiteral>	
Value	<operand>	Comparisons eq, ne, gt, ge, lt, le; explicit properties for automatic code generation: strModel etc.
stringValue	<stringOperand>	String operations contains, starts-with, ends-with, regex; explicit properties for automatic code generation: strModel etc.
$field	-	string following the modelStringPattern
$strVal	-	string following the standardString
$attribute	-	explained in Security Access Rules; not used for query language
$numVal	<NumericalLiteral>	Number constant
$hexVal	<HexLiteral>	Hex number constant
$dateTimeVal	<DateTimeLiteral>	DateTime constant
$timeVal	<TimeLiteral>	Time constant
$boolean	<BoolLiteral>	Boolean constant
$strCast	<castToString>	any Value can be used as input
$numCast	<castToNumerical>	any Value can be used as input
$hexCast	<castToHex>	any Value can be used as input
$boolCast	<castToBool>	any Value can be used as input
$dateTimeCast	<castToDateTime>	any Value can be used as input
$timeCast	<castToTime>	any Value can be used as input
$dayOfWeek	<dateTimeToNum>	extract day of week from DateTime; needed for Security Access Rules
$dayOfMonth	<dateTimeToNum>	extract day of month from DateTime; needed for Security Access Rules
$month	<dateTimeToNum>	extract month from DateTime; needed for Security Access Rules
$year	<dateTimeToNum>	extract year from DateTime; needed for Security Access Rules
comparisonItems	-	Comparisons eq, ne, gt, ge, lt, le
stringItems	-	String operations contains, starts-with, ends-with, regex
matchExpression	<matchExpression>	nested match and comparisons and string operations

logicalExpression	<logicalExpression>	nested logicalExpression (including match) and comparisons and string operations
attributeItem	-	explained in Security Access Rules; not used for query language
ACL	-	explained in Security Access Rules; not used for query language
rightsEnum	-	explained in Security Access Rules; not used for query language
AccessPermissionRule	-	explained in Security Access Rules; not used for query language
AllAccessPermissionRules	-	explained in Security Access Rules; not used for query language
$condition	<logicalExpression>	Root object for the query condition expression
$select	<selectStatement>	Optional expresion to control the returned fields. Only 'id' is possible.

Examples for Grammar and JSON Schema

Single comparison

Grammar	JSON Schema

`$aas#idShort`
 	
`$eq`
 	
`$aas#assetInformation.assetType`
	`{`
	`  "$condition": {`
	`  "$eq": [`
	`  { "$field": "$aas#idShort" },`
	`  {`
	`  "$field":`
	
	`"$aas#assetInformation.assetType"`
	`  }`
	`  ]`
	`  }`
	`}`

HandoverDocumentation with VDI 2770 Class 03-01 Commissioning and language NL (as expected with SubmodelElementList)+

Grammar	JSON Schema

```
$match(
  $sme.Documents[].DocumentClassification
    .Class#value   $eq  "03-01",
  $sme.Documents[].DocumentVersion.
    SMLLanguages[]#language  $eq  "nl"
)
```

```
{
  "$condition": {
    "$match": [
      { "$eq": [
          { "$field": "$sme.Documents[].
            DocumentClassification.Class
            #value" },
          { "$strVal": "03-01" }
        ]
      },
      { "$eq": [
          { "$field": "$sme.Documents[].

DocumentVersion.SMLLanguages[]
            #language" },
          { "$strVal": "nl" }
        ]
      }
    ]
  }
}
```

TechnicalData with motor starter (ECLASS ClassId = 27-37-09-05) and width less than 100 mm

Grammar	JSON Schema

```
$and(
    $match(
        $sm#idShort $eq
"TechnicalData",

$sme.ProductClassifications[]
            .ProductClassId#value
        $eq   "27-37-09-05"
    ),
    $match(
        $sm#idShort $eq
"TechnicalData",
        $sme#semanticId $eq
            "0173-1#02-BAF016#006",
        $sme#value $lt 100
    )
)
```

```
{
  "$condition": {
    "$and": [
      { "$match": [
          { "$eq": [
              { "$field": "$sm#idShort" },
              { "$strVal": "TechnicalData" }
            ]
          },
          { "$eq": [
              {
                "$field":
                "$sme.ProductClassifications[]
                  .ProductClassId#value"
              },
              { "$strVal": "27-37-09-05" }
            ]
          }
        ]
      },
      { "$match": [
          {
            "$eq": [
              { "$field": "$sm#idShort" },
              { "$strVal": "TechnicalData" }
            ]
          },
          {
            "$eq": [
              { "$field": "$sme#semanticId" },
              { "$strVal": "0173-1#02-
BAF016#006" }
            ]
          },
          {
            "$lt": [
              { "$field": "$sme#value" },
              { "$numVal": 100 }
            ]
          }
        ]
      }
    ]
  }
}
```

Match specificAssetIDs

Grammar	JSON Schema

```
$or (
  $match (

$aas#assetInformation.specificAssetIds[]
      .name     $eq   "supplierId",

$aas#assetInformation.specificAssetIds[]
      .value    $eq   "aas-1"
  ),

  $match (

$aas#assetInformation.specificAssetIds[]
      .name     $eq   "customerId",

$aas#assetInformation.specificAssetIds[]
      .value    $eq   "aas-2"
  )
)
```

```
{
  "$condition": {
    "$or": [
      {
        "$match": [
          { "$eq": [
              { "$field":
"$aas#assetInformation.
              specificAssetIds[].name" },
              { "$strVal": "supplierId" }
            ]
          },
          { "$eq": [
              { "$field":
"$aas#assetInformation
              .specificAssetIds[].value"
},
              { "$strVal": "aas-1" }
            ]
          }
        ]
      },
      {
        "$match": [
          {
            "$eq": [
              { "$field":
"$aas#assetInformation
              .specificAssetIds[].name"
},
              { "$strVal": "customerId" }
            ]
          },
          {
            "$eq": [
              { "$field":
"$aas#assetInformation
              .specificAssetIds[].value"
},
              { "$strVal": "aas-2" }
            ]
          }
        ]
      }
    ]
  }
}
```

API Specification

API Interfaces

Asset Administration Shell Interfaces

General

The Asset Administration Shell and Submodel interface 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 of 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

The Asset Administration Shell interface enables the access of the elements of a single Asset Administration Shell.

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	AssetAdministrationShell object	1
Output Parameter				
statusCode	StatusCode	yes	StatusCode	1
payload	AssetAdministrationShell	no	Replaced AssetAdministrationShell	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

Interface: *Submodel*	
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 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 new submodel element must be set in the payload for non-identifiable Referables not being a direct child of a SubmodelElementList. For Elements being a direct child of a SubmodelElementList the input parameter index must be specified.
PutSubmodelElementByPath	Creates a new or replaces an existing submodel element at a specified path within the submodel element hierarchy
PatchSubmodelElementByPath	Updates an existing submodel element or creates a new 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	idShortPath 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	idShortPath 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	idShortPath 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	idShortPath 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	no	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	no	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	
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 for non-identifiable Referables not being a direct child of a SubmodelElementList. For Elements being a direct child of a SubmodelElementList the input parameter index must be specified. 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	idShortPath via relative Reference/Keys to a submodel element under which the new SubmodelElement shall be added	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 or creates a new 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	idShortPath via relative Reference/Keys to a submodel element which shall be replaced	yes	Key	1..*
submodelElement	Submodel element object	yes	SubmodelElement	1
Output Parameter				
StatusCode	Status code	yes	StatusCode	1
payload	New state of the submodel element	no	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	idShortPath 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	no	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 [1]			
semanticId	https://admin-shell.io/aas/API/GetSubmodelElementValueByPath/3/0			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
path	idShortPath via relative Reference/Keys to a submodel element	yes	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 [1]			
semanticId	https://admin-shell.io/aas/API/PatchSubmodelElementValueByPath/3/0			
Name	**Description**	**Mand.**	**Type**	**Card.**

Input Parameter				
path	idShortPath 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	idShortPath 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/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
path	idShortPath 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	idShortPath 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	Duration indicating when the client suggests the server to have finished execution of the invoked operation. The server may take this value into account to decide on its effective timeout, however, the server may or may not use by its own discretion.	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/3/1			
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	BaseOperationResult	1

Operation GetOperationAsyncResult

Operation Name	*GetOperationAsyncResult*			
Explanation	Returns the OperationResult of an asynchronously invoked operation			
semanticId	https://admin-shell.io/aas/API/GetOperationAsnycResult/3/1			
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 all concept descriptions that are referenced in the Submodels contained in the serialization and that are known to the server	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/aasx+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 or creates 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 PutAASXByPackageId

Operation Name	*PutAASXByPackageId*			
Explanation	Replaces or creates the AASX package at the server			
semanticId	https://admin-shell.io/aas/API/PutAASXByPackageId/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 DeleteAASXByPackageId

Operation Name	*DeleteAASXByPackageId*			
Explanation	Deletes a specific AASX package from the server			
semanticId	https://admin-shell.io/aas/API/DeleteAASXByPackageId/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

Registration 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 Asset Administration Shell Interfaces) 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 Publish and Discovery Interfaces.

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
CreateBulkAssetAdministrationShellDescriptors	Creates multiple new Asset Administration Shell Descriptors, i.e., registers multiple Asset Administration Shells
PutAssetAdministrationShellDescriptorById	Creates or replaces an existing Asset Administration Shell Descriptor, i.e., replaces registration information
PutBulkAssetAdministrationShellDescriptorsById	Creates or updates multiple existing Asset Administration Shell Descriptors
DeleteAssetAdministrationShellDescriptorById	Deletes an Asset Administration Shell Descriptor, i.e., de-registers an AAS
DeleteBulkAssetAdministrationShellDescriptorsById	Deletes multiple Asset Administration Shell Descriptors, i.e., de-registers multiple Asset Administration Shells
QueryAssetAdministrationShellDescriptors	Returns all Asset Administration Shell Descriptors that conform to a certain input query.

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, Role, NotApplicable)	no	AssetKind	1
assetType	The type of the assets to retrieve, encoded as unique ID	no	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/3/1			
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/3/1

Name	Description	Mand.	Type	Card.
Input Parameter				
shellDescriptor	Object containing 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 CreateBulkAssetAdministrationShellDescriptors

Operation Name	*CreateBulkAssetAdministrationShellDescriptors*
Explanation	Creates multiple new Asset Administration Shell Descriptors, i.e., registers multiple Asset Administration Shells
semanticId	https://admin-shell.io/aas/API/CreateBulkAssetAdministrationShellDescriptors/3/1

Name	Description	Mand.	Type	Card.
Input Parameter				
shellDescriptors	List of Asset Administration Shell Descriptor objects	yes	AssetAdministrationShellDescriptor	1..*
Output Parameter				
statusCode	Status code	yes	StatusCode	1

Operation PutAssetAdministrationShellDescriptorById

Operation Name	*PutAssetAdministrationShellDescriptorById*
Explanation	Creates or replaces an existing Asset Administration Shell Descriptor, i.e., replaces registration information
semanticId	https://admin-shell.io/aas/API/PutAssetAdministrationShellDescriptorById/3/1

Name	Description	Mand.	Type	Card.
Input Parameter				
shellDescriptor	Object containing the Asset Administration Shell's identification and endpoint information containing the Asset Administration Shell's identification and endpoint information	yes	AssetAdministrationShellDescriptor	1

Output Parameter				
statusCode	Status code	yes	StatusCode	1
payload	New state of the Asset Administration Shell Descriptor	no	AssetAdministrationShellDescriptor	1

Operation PutBulkAssetAdministrationShellDescriptorsById

Operation Name	*PutBulkAssetAdministrationShellDescriptorsById*			
Explanation	Updates multiple existing Asset Administration Shell Descriptors or creates them if they do not exist			
semanticId	https://admin-shell.io/aas/API/PutBulkAssetAdministrationShellDescriptorsById/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
shellDescriptors	List of Asset Administration Shell Descriptor objects	yes	AssetAdministrationShellDescriptor	1..*
Output Parameter				
statusCode	Status code	yes	StatusCode	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	Identifier	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1

Operation DeleteBulkAssetAdministrationShellDescriptorsById

Operation Name	*DeleteBulkAssetAdministrationShellDescriptorsById*
Explanation	Deletes multiple Asset Administration Shell Descriptors, i.e., de-registers multiple Asset Administration Shells

semanticId	https://admin-shell.io/aas/API/DeleteBulkAssetAdministrationShellDescriptorsById/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
shellDescriptors	List of Asset Administration Shell Descriptor objects	yes	AssetAdministrationShellDescriptor	1..*
Output Parameter				
statusCode	Status code	yes	StatusCode	1

Operation QueryAssetAdministrationShellDescriptors

Operation Name	*QueryAssetAdministrationShellDescriptors*			
Explanation	Returns all Asset Administration Shell Descriptors that conform to a certain input query.			
semanticId	https://admin-shell.io/aas/API/QueryAssetAdministrationShellDescriptors/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
query	Query conforming to the AAS Query Language	yes	AAS Query	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
payload	List of Asset Administration Shell Descriptors or AAS identifiers	no	QueryResult<AssetAdministrationShellDescriptor>	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

Interface: *Submodel Registry*	
PostBulkSubmodelDescriptors	Creates one or more new submodel descriptors, i.e., registers several submodels
PutSubmodelDescriptorById	Creates or replaces an existing submodel descriptor, i.e., replaces registration information
PutBulkSubmodelDescriptorsById	Creates or replaces one or more existing submodel descriptors, i.e., replaces registration information of several submodels
DeleteSubmodelDescriptorById	Deletes a submodel descriptor, i.e., de-registers a submodel
DeleteBulkSubmodelDescriptorsById	Deletes one or more submodel descriptors, i.e., de-registers several submodels
QuerySubmodelDescriptors	Returns all Submodel Descriptors that conform to a certain input query.

Operation GetAllSubmodelDescriptors

Operation Name	*GetAllSubmodelDescriptors*			
Explanation	Returns all submodel descriptors			
semanticId	https://admin-shell.io/aas/API/GetAllSubmodelDescriptors/3/1			
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	StatusCode	yes	StatusCode	1
payload	List of requested submodel descriptors	no	SubmodelDescriptor	1..*

Operation GetSubmodelDescriptorById

Operation Name	*GetSubmodelDescriptorById*			
Explanation	Returns a specific Submodel Descriptor			
semanticId	https://admin-shell.io/aas/API/GetSubmodelDescriptorById/3/1 =			
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/3/1			
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 PostBulkSubmodelDescriptors

Operation Name	*PostBulkSubmodelDescriptors*			
Explanation	Creates one or more new submodel descriptors, i.e., registers several submodels			
semanticId	https://admin-shell.io/aas/API/PostBulkSubmodelDescriptors/3/1			
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

Operation PutSubmodelDescriptorById

Operation Name	*PutSubmodelDescriptorById*			
Explanation	Replaces an existing submodel descriptor, i.e., replaces registration information			
semanticId	https://admin-shell.io/aas/API/PutSubmodelDescriptorById/3/1			
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	no	SubmodelDescriptor	1

Operation PutBulkSubmodelDescriptorsById

Operation Name	*PutBulkSubmodelDescriptorsById*			
Explanation	Replaces one or more existing submodel descriptors, i.e., replaces registration information of several submodels			
semanticId	https://admin-shell.io/aas/API/PutBulkSubmodelDescriptorsById/3/1			
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/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**

Input Parameter				
submodelIdentifier	The Submodel's unique id	yes	Identifier	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1

Operation DeleteBulkSubmodelDescriptorsById

Operation Name	*DeleteBulkSubmodelDescriptorsById*			
Explanation	Deletes one or more submodel descriptors, i.e., de-registers several submodels			
semanticId	https://admin-shell.io/aas/API/DeleteBulkSubmodelDescriptorsById/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
submodelIdentifier	The Submodel's unique ID	yes	Identifier	1..*
Output Parameter				
statusCode	Status code	yes	StatusCode	1

Operation QuerySubmodelDescriptors

Operation Name	*QuerySubmodelDescriptors*			
Explanation	Returns all Submodel Descriptors that conform to a certain input query.			
semanticId	https://admin-shell.io/aas/API/QuerySubmodelDescriptors/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
query	Query conforming to the AAS Query Language	yes	AAS Query	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
payload	List of Submodel Descriptors or Submodel identifiers	no	QueryResult<SubmodelDescriptor>	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 Asset Administration Shell Interfaces. 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 (Publish and Discovery Interfaces) and lookup (Registration Interfaces) 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 Repository*	
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	Creates or replaces an existing Asset Administration Shell
DeleteAssetAdministrationShellById	Deletes an Asset Administration Shell
QueryAssetAdministrationShells	Returns all Asset Administration Shells that conform to a certain input query

Operation GetAllAssetAdministrationShells

Operation Name	*GetAllAssetAdministrationShells*			
Explanation	Returns all Asset Administration Shells			
semanticId	https://admin-shell.io/aas/API/GetAllAssetAdministrationShells/3/1			
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	AssetAdministration Shell	1..*

Operation GetAssetAdministrationShellById

Operation Name	*GetAssetAdministrationShellById*			
Explanation	Returns a specific Asset Administration Shell			
semanticId	https://admin-shell.io/aas/API/GetAssetAdministrationShellById/3/1			
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	AssetAdministration Shell	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/3/1			
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	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	AssetAdministration Shell	1..*

Operation GetAllAssetAdministrationShellsByIdShort

Operation Name	*GetAllAssetAdministrationShellsByIdShort*			
Explanation	Returns all Asset Administration Shells with a specific *idShort*			
semanticId	https://admin-shell.io/aas/API/GetAllAssetAdministrationShellsByIdShort/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
idShort	The Asset Administration Shell'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 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/3/1			
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	Creates or replaces an existing Asset Administration Shell			
semanticId	https://admin-shell.io/aas/API/PutAssetAdministrationShellById/3/1			
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	no	AssetAdministrationShell	1

Operation DeleteAssetAdministrationShellById

Operation Name	*DeleteAssetAdministrationShellById*			
Explanation	Deletes an Asset Administration Shell			
semanticId	https://admin-shell.io/aas/API/DeleteAssetAdministrationShellById/3/1			
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

Operation QueryAssetAdministrationShells

Operation Name	*QueryAssetAdministrationShells*			
Explanation	Returns all Asset Administration Shells that conform to a certain input query.			
semanticId	https://admin-shell.io/aas/API/QueryAssetAdministrationShells/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
query	Query conforming to the AAS Query Language	yes	AAS Query	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
payload	List of Asset Administration Shells or AAS identifiers	no	QueryResult<AssetAdministrationShell>	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
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	Creates or replaces an existing Submodel
PatchSubmodelById	Updates an existing submodel
DeleteSubmodelById	Deletes a Submodel
QuerySubmodels	Returns all Submodels that conform to a certain input query

Operation GetAllSubmodels

Operation Name	*GetAllSubmodels*			
Explanation	Returns all Submodels			
semanticId	https://admin-shell.io/aas/API/GetAllSubmodels/3/1			
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/3/1			
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/3/1			
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/3/1			
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/3/1			
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	Creates a new or replaces an existing Submodel			
semanticId	https://admin-shell.io/aas/API/PutSubmodelById/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
submodel	Submodel object	yes	Submodel	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
payload	New state of the Submodel	no	Submodel	1

Operation PatchSubmodelById

Operation Name	*PatchSubmodelById*			
Explanation	Updates an existing Submodel			
semanticId	https://admin-shell.io/aas/API/PatchSubmodelById/3/1			
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	no	Submodel	1

Operation DeleteSubmodelById

Operation Name	*DeleteSubmodelById*
Explanation	Deletes a Submodel

semanticId	https://admin-shell.io/aas/API/DeleteSubmodelById/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
id	The Submodel's unique ID	yes	Identifier	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1

Operation QuerySubmodels

Operation Name	*QuerySubmodels*			
Explanation	Returns all Submodels that conform to a certain input query.			
semanticId	https://admin-shell.io/aas/API/QuerySubmodels/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
query	Query conforming to the AAS Query Language	yes	AAS Query	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
payload	List of Submodels or Submodel identifiers	no	QueryResult<Submodel>	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

Interface: *Concept Description Repository*	
GetAllConceptDescriptionsByDataSpecificationReference	Returns all Concept Descriptions with a specific *dataSpecification* reference
PostConceptDescription	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.
PutConceptDescriptionById	Creates or replaces an existing Concept Description
DeleteConceptDescriptionById	Deletes a Concept Description
QueryConceptDescriptions	Returns all Concept Descriptions that conform to a certain input query

Operation GetAllConceptDescriptions

Operation Name	*GetAllConceptDescriptions*			
Explanation	Returns all Concept Descriptions			
semanticId	https://admin-shell.io/aas/API/GetAllConceptDescriptions/3/1			
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/3/1			
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/3/1			
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/3/1			
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/3/1			
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/3/1			
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	Creates or replaces an existing Concept Description			
semanticId	https://admin-shell.io/aas/API/PutConceptDescriptionById/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
conceptDescription	Concept Description object	yes	ConceptDescription	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
payload	New state of the Concept Description	no	ConceptDescription	1

Operation DeleteConceptDescriptionById

Operation Name	*DeleteConceptDescriptionById*			
Explanation	Deletes a Concept Description			
semanticId	https://admin-shell.io/aas/API/DeleteConceptDescriptionById/3/1			
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

Operation QueryConceptDescriptions

Operation Name	*QueryConceptDescriptions*
Explanation	Returns a list of Concept Descriptions based on an input query.
semanticId	https://admin-shell.io/aas/API/QueryConceptDescriptions/3/1

Name	Description	Mand.	Type	Card.
Input Parameter				
query	The specific query conformant to the AAS Query Language, containing filter conditions for the required Concept Descriptions.	yes	AAS Query	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
payload	List of requested Concept Descriptions or identifiers	no	QueryResult<ConceptDescription>	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
SearchAllAssetAdministrationShellIdsByAssetLink	Returns a list of Asset Administration Shell IDs linked to specific asset identifiers or the global asset 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 <<Deprecated>>*
Explanation	Returns a list of Asset Administration Shell ids based on asset identifier key-value-pairs

semanticId	https://admin-shell.io/aas/API/GetAllAssetAdministrationShellIdsByAssetLink/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
assetIds	The specific assetId of an asset identifier, which could be the globalAssetId or specificAssetIds. Note: the name for the globalAssetId is defined in Contraint AASd-116 in IDTA-0001. It is the predefined key "*globalAssetId*" that would refer to the *AssetInformation/globalAssetId*.	no	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	no	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/3/1			
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 for the globalAssetId is defined in Contraint AASd-116 in IDTA-0001. It is the predefined name “*globalAssetId*” that would refer to the *AssetInformation/globalAssetId*.	no	SpecificAssetId	1..*

Operation SearchAllAssetAdministrationShellIdsByAssetLink

Operation Name	*SearchAllAssetAdministrationShellIdsByAssetLink*			
Explanation	Returns a list of Asset Administration Shell IDs linked to specific asset identifiers or the global asset ID			
semanticId	https://admin-shell.io/aas/API/SearchAllAssetAdministrationShellIdsByAssetLink/3/1			
Name	**Description**	**Mand.**	**Type**	**Card.**
Input Parameter				
assetLinks	A list of AssetLinks that all shall match. An AssetLink might be either derived from a SpecificAssetId ("name": "<specificAssetId.name>", "value": "<specificAssetId.value>") or a globalAssetId ("name": "globalAssetId", "value": "<globalAssetId-value>"). Note: The name for the globalAssetId is defined in Contraint AASd-116 in IDTA-0001. It is the predefined key “*globalAssetId*” that would refer to the *AssetInformation/globalAssetId*.	yes	AssetLink	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	no	Identifier	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/3/1			
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: "*globalAssetId*" is a predefined name (see Constraint AASd-116 in IDTA-0001). If a specific asset ID uses this predefined name "*globalAssetId*" then its value shall be used as the value for the *AssetInformation/globalAssetId* attribute of the associated AAS. It shall not be treated as an additional item of *AssetInformation/specificAssetIds*. Furthermore, a potentially included SpecificAssetId/externalSubjectId attribute for a specific asset ID with name "globalAssetId" shall be considered an invalid specific asset ID. Potentially included semanticId or supplementalSemanticIds are ignored.	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/3/1			
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/3/1			
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 “GetSelfDescription” 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 the Bibliography. The AAS package format and the AAS Package type are defined in Part 5 [4]. 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/3/1		
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/3/1		
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”, “Instance”, ”Role”, or "NotApplicable"	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/3/1		
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/3/1		
Attribute	**Explanation**	**Type**	**Card.**
protocolInformation	Protocol information of the network resource endpoint	ProtocolInformation	1
interface	Name of the offered interface at the endpoint	NameType	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

Which can be combined with the following serialisation modifiers:

Serialisation Modifier	serialisation-modifier-value
Metadata	METADATA
ValueOnly	VALUE
Path	PATH
Reference	REFERENCE

The value for the interface attribute shall be the concatenation of the interface-shortName, an optional serialisation modifier value, and the provided version. An non-existent serialisation modifier value represents the Normal serialisation.

- “{interface-shortName}[-{serialisation-modifier-value}]-{interface-version}”*

The interface-version of this specification is “3.1”, e.g. the entry for the Asset Administration Shell Interface is “AAS-3.1”.

See the following example for descriptor endpoints, that expose two submodel endpoints providing *Normal* serialisations but different versions of the API, and one endpoint providing the submodel in the ValueOnly serisalisation:

```
"endpoints": [
 {
    "protocolInformation": {
      "endpointAddress": "https://localhost:1234/api/3/submodel",
      "endpointProtocolVersion": "1.1"
    },
    "interface": "SUBMODEL-3.1"
 },
  {
    "protocolInformation": {
      "endpointAddress": "https://localhost:5678/api/2/submodel",
      "endpointProtocolVersion": "1.1"
    },
    "interface": "SUBMODEL-2.0"
 },
 {
    "protocolInformation": {
      "endpointAddress": "https://localhost:1234/api/3/submodel/$value",
      "endpointProtocolVersion": "1.1"
    },
    "interface": "SUBMODEL-VALUE-3.1"
 }
]
```

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 a URL	LocatorType	1
endpointProtocol	Either scheme of endpointAddress or scheme + further information. Scheme denotes the highest level of doubtless transmission.	SchemeType	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	NameType	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	ShortIdType	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	TextType	0..1
subprotocolBodyEncoding	If subprotocolBody is present, the encoding might be explicitly defined, otherwise it shall default to subprotocols encoding scheme	NameType	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	0..*

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	**ID**
	Explanation
NONE	https://admin-shell.io/aas/API/DataTypes/SecurityTypeEnum/NONE/3/0
	No predefined security type available
RFC_TLSA	https://admin-shell.io/aas/API/DataTypes/SecurityTypeEnum/RFC_TLSA/3/0
	TLSA according to RFC 6698
W3C_DID	https://admin-shell.io/aas/API/DataTypes/SecurityTypeEnum/W3C_DID/3/0
	Decentralized Identifiers according to the W3C Recommendation [7]

AssetLink

Class Name	*AssetLink*		
Explanation	Asset identifier derieved from either SpecificAssetId or GlobalAssetId		
Inherits from	-		
semanticId	https://admin-shell.io/aas/API/DataTypes/AssetLink/3/1		
Attribute	**Explanation**	**Type**	**Card.**
name	Name of the Asset identifier, i.e., "globalAssetId", a serial number, manufacturer part ID, or customer part IDs	LabelType	1
value	Value of the Asset Identifier	Identifier	1

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/3/1		
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 [http-rest-api::service-specifications-and-profiles:::service-specifications-and-profiles] for further details.
semanticId	https://admin-shell.io/aas/API/DataTypes/ServiceSpecificationProfileEnum/3/1

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/AssetAdministrationShellRegistryServiceSpecification/SSP-001	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/AssetAdministrationShellRegistryServiceSpecification/SSP-002	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/3/1/AssetAdministrationShellRegistryServiceSpecification/SSP-003	Indicates that the server implemented all details of the Asset Administration Shell Registry Service Specification Bulk Profile in version 3.1.

https://admin-shell.io/aas/API/3/0/SubmodelRegistryServiceSpecification/SSP-001	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/SubmodelRegistryServiceSpecification/SSP-002	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/3/1/SubmodelRegistryServiceSpecification/SSP-003	Indicates that the server implemented all details of the Submodel Registry Service Specification Bulk Profile in version 3.1.
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/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 Repository 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 Repository 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 Repository 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 Repository Specification Template Profile in version 3.0.
https://admin-shell.io/aas/API/3/1/ConceptDescriptionRepositoryServiceSpecification/SSP-001	Indicates that the server implemented all details of the Concept Description Repository Service Specification Read Template Profile in version 3.1.

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

```
{
  "profiles": [
    "https://admin-shell.io/aas/API/3/0/DiscoveryServiceSpecification/SSP-001",
    "https://admin-shell.io/aas/API/3/0/RegistryServiceSpecification/SSP-002"
  ]
}
```

Query Results

Result elements of AAS queries depend on the declared output selector and the target of the query. Possible targets are Asset Administration Shells, Submodels, Concept Descriptions, Asset Administration Shell Descriptors, and Submodel Descriptors. In accordance, also these objects are returned in the Query Result. In addition, their representation can be controlled via the select statements of the input query, e.g., by only selecting the identifiers of the respective objects rather than the whole content.

Table 3. Simple Data Types used for API-specific Classes

Query Result	Definition	Value Examples

QueryResult<AssetAdministrationShell>	List of Asset Administration Shells *or* a list of Asset Administration Shell identifiers	`[ { "modelType": "AssetAdministrationShell", "id": "https://example.com/aas-1", ... }, { "modelType": "AssetAdministrationShell", "id": "https://example.com/aas-2", ... }, ... ]`
		
		`[ "https://example.com/aas-1", "https://example.com/aas-2", ... ]`

QueryResult<Submodel>	List of Submodels *or* a list of Submodel identifiers	<pre>[
		  {
		  "modelType": "Submodel",
		  "id": "https://example.com/sm-1",
		  ...
		  },
		  {
		  "modelType": "Submodel",
		  "id": "https://example.com/sm-2",
		  ...
		  },
		  ...
		]</pre>
		<pre>[
		  "https://example.com/sm-1",
		  "https://example.com/sm-2",
		  ...
		]</pre>

QueryResult<ConceptDescription>	List of Concept Descriptions *or* a list of Concept Description identifiers	<pre>[
		  {
		  "modelType": "ConceptDescription",
		  "id": "https://example.com/cd-1",
		  ...
		  },
		  {
		  "modelType": "ConceptDescription",
		  "id": "https://example.com/cd-2",
		  ...
		  },
		  ...
		]</pre>
		<pre>[
		  "https://example.com/cd-1",
		  "https://example.com/cd-2",
		  ...
		]</pre>

QueryResult<AssetAdministrationShellDescriptor>	List of Asset Administration Shell Descriptors *or* a list of Asset Administration Shell identifiers	<pre>[
		  {
		  "modelType": "AssetAdministrationShellDescriptor",
		  "id": "https://example.com/aas-1",
		  ...
		  },
		  {
		  "modelType": "AssetAdministrationShellDescriptor",
		  "id": "https://example.com/aas-2",
		  ...
		  },
		  ...
		]</pre>
		<pre>[
		  "https://example.com/aas-1",
		  "https://example.com/aas-2",
		  ...
		]</pre>

QueryResult<SubmodelDescriptor>	List of Submodel Descriptors *or* a list of Submodel identifiers	<pre>[
		  {
		  "modelType": "SubmodelDescriptor",
		  "id": "https://example.com/sm-1",
		  ...
		  },
		  {
		  "modelType": "SubmodelDescriptor",
		  "id": "https://example.com/sm-2",
		  },
		  ...
		]</pre>
		<pre>[
		  "https://example.com/sm-1",
		  "https://example.com/sm-2",
		  ...
		]</pre>

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 4.

Table 4. 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 [1]	0 42

Primitive Data Types

All primitive data types from Part 1 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 5. 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* (string with max 128 and min 1 characters) 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”
LocatorType	string with max 2048 and min 1 characters	“https://example.org/” “https://provider-edc.data.plane/v3.0/shells/aas-1/submodels/submodel-1/submodel”
TextType	string with max 2048 and min 1 characters	“asset:prop:id=123;idsEndpoint=http://edc.control.plane/”
SchemeType	same as NameType (string with max 128 and min 1 characters)	“HTTP”

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 General Result Object.

Table 6 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 6. 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 suggests the server to have finished execution of the invoked operation. The server may take this value into account to decide on its effective timeout, however, the server may or may not use by its own discretion.	duration	0..1

OperationRequestValueOnly

Class Name	*OperationRequestValueOnly*		
Explanation	The operation request object		
Inherits from	—		
semanticId*	https://admin-shell.io/aas/API/DataTypes/OperationRequestValueOnly/3/1		
Attribute	**Explanation**	**Type**	**Card.**
inputArguments	Input argument	SubmodelElementValue	0..*
inoutputArguments	InOutput argument	SubmodelElementValue	0..*
clientTimeoutDuration	Duration indicating when the client suggests the server to have finished execution of the invoked operation. The server may take this value into account to decide on its effective timeout, however, the server may or may not use by its own discretion.	duration	0..1

BaseOperationResult

Class Name	*BaseOperationResult*		
Explanation	The object containing the intermediate state of an operation		
Inherits from	Result		
semanticId	https://admin-shell.io/aas/API/DataTypes/BaseOperationResult/3/1		
Attribute	**Explanation**	**Type**	**Card.**
executionState	Execution state	ExecutionState	1
success	Flag indicating whether the business operation behind the operation was successful (true) or not (false)	boolean	0..1

OperationResult

Class Name	*OperationResult*

Explanation	The operation’s invocation result object		
Inherits from	BaseOperationResult		
semanticId	https://admin-shell.io/aas/API/DataTypes/OperationResult/3/0		
Attribute	**Explanation**	**Type**	**Card.**
outputArguments	Output argument	OperationVariable	0..*
inoutputArguments	InOutput argument	OperationVariable	0..*

OperationResultValueOnly

Class Name	*OperationResultValueOnly*		
Explanation	The operation’s invocation result object in the ValueOnly notation.		
Inherits from	BaseOperationResult		
semanticId	https://admin-shell.io/aas/API/DataTypes/OperationResultValueOnly/3/1		
Attribute	**Explanation**	**Type**	**Card.**
outputArguments	Output argument	SubmodelElementValue	0..*
inoutputArguments	InOutput argument	SubmodelElementValue	0..*

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.

Enumeration: Level

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

Table 7. 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 Modifier Constraints.

Enumeration: Content

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

For Content equal to Normal see Clause https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.1/mappings/mappings.html#_format_normal_in_xml and https://https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.1/mappings/mappings.html#_format_normal_in_json in Part 1 for details. It is defined for XML and JSON only.

For Content equal to Metadata see Clause https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.1/mappings/mappings.html#_format_metadata_metadata_serialization in Part 1 for details.

For Content equal to Value see Clause https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.1/mappings/mappings.html#value-only-serialization-in-json in Part 1 for details. It is defined for JSON only.

For Content equal to Reference see Clause https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.1/mappings/mappings.html#_format_reference in Part 1 for details.

For Content equal to Path see Clause https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.1/mappings/mappings.html#_format_path_idshortpath_serialization_in_json in Part 1 for details.

Table 8. 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 Modifier Constraints.

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 9. 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 Modifier Constraints.

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 10. 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

Resource Name	Level Modifier	Content Modifier	Extent Modifier
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.

[1] https://www.w3.org/TR/xmlschema-2/

HTTP/REST API

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/V3.1.0#
This schema includes general objects, which are used in the further defined APIs.

Additional objects are needed for Part 2, e.g., 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/V3.1.0#
This schema includes general objects, which are used in the further defined APIs.

The AAS Service Specification including the AAS API, the Submodel API, the Serialization API, and the Self-Description API is available at:
https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellServiceSpecification/V3.1.0_SSP-001#

The Submodel Service Specification including the Submodel API, the Serialization API, and the Self-Description API is available at:
https://app.swaggerhub.com/apis/Plattform_i40/SubmodelServiceSpecification/V3.1.0_SSP-001#

The AAS Repository Service Specification including the AAS Repository API, the AAS API, the Submodel API, the Submodel Repository API, the Serialization API, and the Self-Description API is available at: https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRepositoryServiceSpecification/V3.1.0_SSP-001#

The Submodel Repository Service Specification including the Submodel Repository API, Submodel API, the Serialization API, and the Self-Description API is available at: https://app.swaggerhub.com/apis/Plattform_i40/SubmodelRepositoryServiceSpecification/V3.1.0_SSP-001#

The AAS Registry Service Specification Registry including the AAS Registry API and the Self-Description API is available at:
https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRegistryServiceSpecification/V3.1.0_SSP-001#

The Submodel Registry Service Specification including the Submodel Registry API and the Self-Description API is available at https://app.swaggerhub.com/apis/Plattform_i40/SubmodelRegistryServiceSpecification/V3.1.0_SSP-001#

The Service Specification including the AAS Discovery API and the Self-Description API is available at: https://app.swaggerhub.com/apis/Plattform_i40/DiscoveryServiceSpecification/V3.1.0_SSP-001#

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 [1] 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 idShortPath to access deeper parts in the hierarchy. To easily support this in the REST API, "." or "[index]" is used as a delimiter in the idShortPaths. Please see API Versioning. Since an idShortPath could include square brackets like "[index]", the idShortPath 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. In particular, base64url-encoded values shall not contain or end with the = character (so-called 'padding') to keep them URL-safe. An appropriate base64url implementation needs to be used for encoding/decoding. See RFC 4648 for further details.

Note: Encoding JSON objects may lead to different encoded strings, depending e.g. whether control characters (white spaces, line-breaks etc.) are included or not. However, servers must be able to deal with included control characters and operate on the original content.

- 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=".

Note: In case more than one globalAssetId or specificAssetId is provided, their order shall not affect the result.

- 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 "[\{"name": "globalAssetId","value": "http://example.company/myAsset"},\{"name": "myOwnInternalAssetId","value": "12345ABC"}]", the resulting base64url-encoded value of the query parameter is "?assetIds=W3sibmFtZSI6ICJnbG9iYWxBc3NldElkIiwidmFsdWUiOiAiaHR0cDovL2V4YW1wbGUuY29tcGFueS9teUFzc2V0In0seyJuYW1lIjogIm15T3duSW50ZXJuYWxBc3NldElkIiwidmFsdWUiOiAiMTIzNDVBQkMifV0".

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 Pagination 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 AAS Query Language is serialised as a JSON Object.
- 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".
- This document does not make any statement about the expected behavior when query parameters with cardinality 1 (e.g. level, extent, etc.) are present more than once in a URL, e.g., "/…?level=deep&level=core". It is up to the implementation how to handle such cases. It is strongly discouraged to make such calls as the result might not be deterministic.

API Versioning

The versioning scheme for AAS API related services follows semantic versioning[1]. 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 Metamodel Versions. 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, "3.1.0RC2" should be interpreted as the second (alternative) release candidate for version 3.1.0.

To provide multiple versions of the APIs to clients, 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).

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.

Figure 5. Example of an AAS endpoint with URL-based versioning

As different solutions also provide different advantages and disadvantages, URL-based versioning is recommended 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.

Upcoming implementations of AAS related servers may implement the version prefix "api/v<X>/" to provide information of the specific major version regarding AAS Part 2 version, where <X> denotes the implemented major version, e.g. "api/v3/" (see Figure 5). As minor releases of one major version must not contain any breaking changes, the declaration of the minor version can be omitted.

Nevertheless, AAS servers may decide to use different paths depending on their context, see Figure 6. The fragment before the functional endpoint is not necessarily the version information itself, it may also be a tenant ID or any other information the server decides to use. A client shall not assume that the pattern from Figure 5 is supported by all servers.

Figure 6. Example of an AAS endpoint with an arbitrary path

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.

Note: The profile identifiers provided at the "/description" endpoint (see Clause ServiceDescription) contain both the major and minor version declaration.

Addressing Resources

The API allows to address each referable element, either by its global identifier or by its idShortPath 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 idShortPath. The idShortPath is a chain of idShorts or SubmodelElementList-indexes, which points to an element within a hierarchy of elements. The root of the idShortPath is always a submodel and the first element in an idShortPath is always an idShort of a first level SubmodelElement within a Submodel. Technically, the idShortPath is a string and the idShorts are separated by a dot while the SubmodelElementList-indexes are written in brackets.

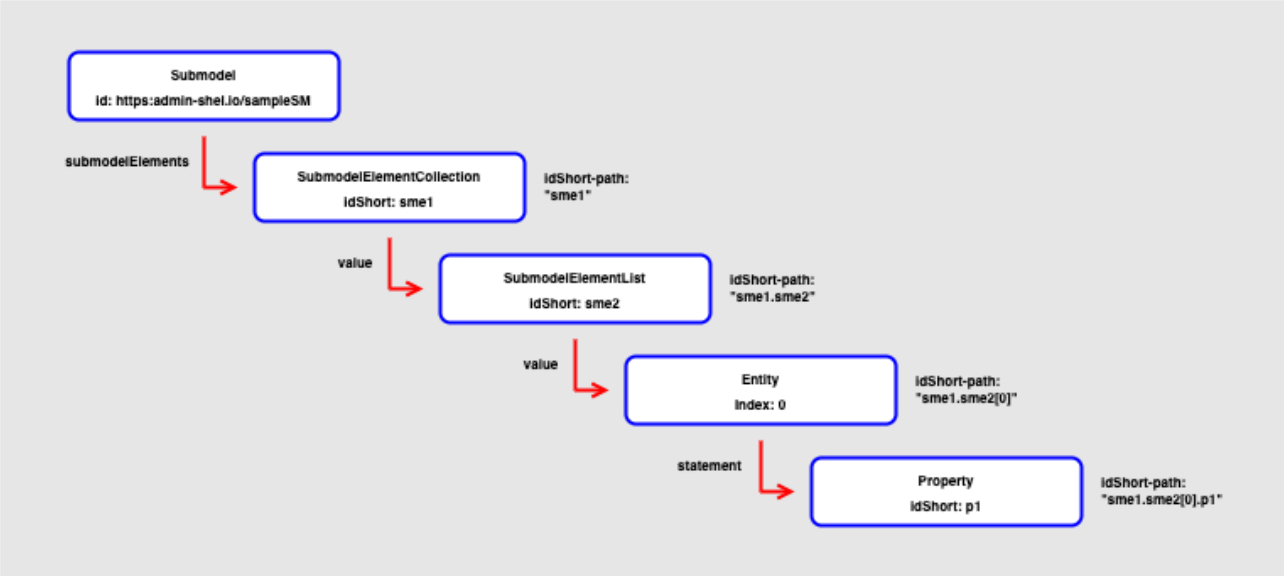


Figure 7. Example Hierarchy of Submodel Elements

The example hierarchy in Figure 7 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 idShortPath. The idShortPath 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 hierarchical 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 11. 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 URL-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]".

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 [2] 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 define the maximum page size of the result list through the limit parameter. The server is allowed to return less elements than requested. If the limit parameter is not specified, the server may decide the number of elements to be returned.

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 12. 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 assumes a deterministic sorting, i.e. the server implementation shall provide a deterministic ordering of the result set. For instance, a server shall 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 be aware that e.g. B and then C are returned after A.

Nevertheless, the inherent order of the result shall 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 client also needs to be aware, that changes may happen in pages which the client has already received, that page results may change over time and that different clients may receive different page results.

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

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 [1].

Additional classes needed for payload of the HTTP/REST API specification can be found in Metamodel Specification Details.

Modifier Constraints

To use metadata objects as described in [1], 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:

- For Content="Value", the requested object shall always be serialized to an unnamed JSON Object or Array. This means that the response object must not have a property with the object's idShort at the root level.
- 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.
- If parameter Content is set to "Metadata" then Level shall not be used. A server shall respond with a ClientErrorBadRequest in this case.
- 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.

> Note: Although metadata and value-only representations of Asset Administration Shells are possible, they are not supported up to now.

Mapping of Operations

The following Table 13 shows the mapping of the generic operations to the HTTP/REST API.

All entries correspond to the generic operations, except the **bold** entries, which exist only in the HTTP/REST API.

Table 13. 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	/aas	
GetAllSubmodelReferences	GET	/aas/submodel-refs	Pagination

Operation Name	HTTP Verb	REST-Path	Comment (e.g. optional query parameters)
PostSubmodelReference	POST	/aas/submodel-refs	Location header of the response contains the value '<baseUrl>/aas/submodel-refs/{submodelIdentifier}' Note 1: submodelIdentifier is the base64url-encoded Submodel.id value. Note 2: There is no API defined for a client to directly send a GET towards this URL of the Location header, however, the information is intended as an input for the DeleteSubmodelReference API Operation.
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

Operation Name	HTTP Verb	REST-Path	Comment (e.g. optional query parameters)
GetAllSubmodelElements	GET	/submodel/submodel-elements	?level=deep/core path-suffix= $metadata/$value/$reference/$path or no suffix for serialization modifier "Normal" ?extent=WithoutBLOBValue/WithBLOBValue Pagination
GetSubmodelElementByPath	GET	/submodel/submodel-elements/{idShortPath}	use separated idShortPath of this element ?level=deep/core path-suffix= $metadata/$value/$reference/$path or no suffix or serialization modifier "Normal" ?extent=WithoutBLOBValue/WithBLOBValue Note: If a client uses a path-suffix for a SubmodelElement that has the associated Serialisation Modifier not defined (e.g. …/capabilityIdShort/$value), a server shall respond with a ClientErrorBadRequest (HTTP status code 400) and an appropriate error description in the response message. URL-encoded IdShortPath
GetFileByPath	GET	/submodel/submodel-elements/{idShortPath}/attachment	use separated idShortPath of this element URL-encoded IdShortPath
PutFileByPath	PUT	/submodel/submodel-elements/{idShortPath}/attachment	use separated idShortPath of this element URL-encoded IdShortPath
DeleteFileByPath	DELETE	/submodel/submodel-elements/{idShortPath}/attachment	use separated idShortPath 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 idShortPath of the parent element SerializationModifiers are not used with POST
PutSubmodelElementByPath	PUT	/submodel/submodel-elements/{idShortPath}	use separated idShortPath of this element URL-encoded IdShortPath

Operation Name	HTTP Verb	REST-Path	Comment (e.g. optional query parameters)
PatchSubmodelElementByPath	PATCH	/submodel/submodel-elements/{idShortPath}	use separated idShortPath 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}/$value	use separated idShortPath of this element; see [1] for values path-suffix=$value URL-encoded IdShortPath
DeleteSubmodelElementByPath	DELETE	/submodel/submodel-elements/{idShortPath}	use separated idShortPath 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>

Operation Name	HTTP Verb	REST-Path	Comment (e.g. optional query parameters)
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**
QueryAssetAdministrationShells	POST	/query/shells	Input query in the request body
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 link: 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

Operation Name	HTTP Verb	REST-Path	Comment (e.g. optional query parameters)
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**
QuerySubmodels	POST	/query/submodels	Input query in the request body
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
QueryConceptDescriptions	POST	/query/concept-descriptions	Input query in the request body
AASX File Server Interface			
GetAllAASXPackageIds	GET	/packages	base64url-encoded identifier Pagination
PostAASXPackage	POST	/packages	

Operation Name	HTTP Verb	REST-Path	Comment (e.g. optional query parameters)
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|Role|NotApplicable 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

Operation Name	HTTP Verb	REST-Path	Comment (e.g. optional query parameters)
Submodel Registry Interface	*****	**/shell-descriptors/{aasIdentifier}/submodelDescriptors/***	**superpath as defined in Service Specification or Profile**
QueryAssetAdministrationShellDescriptors	POST	/query/shell-descriptors	Input query in the request body
CreateBulkAssetAdministrationShellDescriptors	POST	/bulk/shell-descriptors	List of new Asset Administration Shell Descriptors in the request body
PutBulkAssetAdministrationShellDescriptorsById	PUT	/bulk/shell-descriptors	List of new versions of Asset Administration Shell Descriptors in the request body. Mapping to existing Asset Administration Shell Descriptors happens via the id attribute.
DeleteBulkAssetAdministrationShellDescriptorsById	DELETE	/bulk/shell-descriptors	List of Asset Administration Shell identifiers in the request body
GetBulkAsyncStatus	GET	/bulk/status/{handleId}	Server-side defined handle to retrieve the status of a previous bulk request. Note: Same endpoint as for the Submodel Registry Interface bulk status.
GetBulkAsyncResult	GET	/bulk/status/{handleId}	Server-side defined handle to retrieve the result of a previous bulk request. Note: Same endpoint as for the Submodel Registry Interface bulk result.
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

Operation Name	HTTP Verb	REST-Path	Comment (e.g. optional query parameters)
DeleteSubmodelDescriptorById	DELETE	/submodel-descriptors/{submodelIdentifier}	base64url-encoded identifier
QuerySubmodelDescriptors	POST	/query/submodel-decriptors	Input query in the request body
CreateBulkSubmodelDescriptors	POST	/bulk/submodel-descriptors	List of new Submodel Descriptors in the request body
PutBulkSubmodelDescriptorsById	PUT	/bulk/submodel-descriptors	List of new versions of Submodel Descriptors in the request body. Mapping to existing Submodel Descriptors happens via the id attribute.
DeleteBulkSubmodelDescriptorsById	DELETE	/bulk/submodel-descriptors	List of Submodel identifiers in the request body
GetBulkAsyncStatus	GET	/bulk/status/{handleId}	Server-side defined handle to retrieve the status of a previous bulk request. Note: Same endpoint as for the Asset Administration Shell Registry Interface bulk status.
GetBulkAsyncResult	GET	/bulk/status/{handleId}	Server-side defined handle to retrieve the result of a previous bulk request. Note: Same endpoint as for the Asset Administration Shell Registry Interface bulk result.
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.

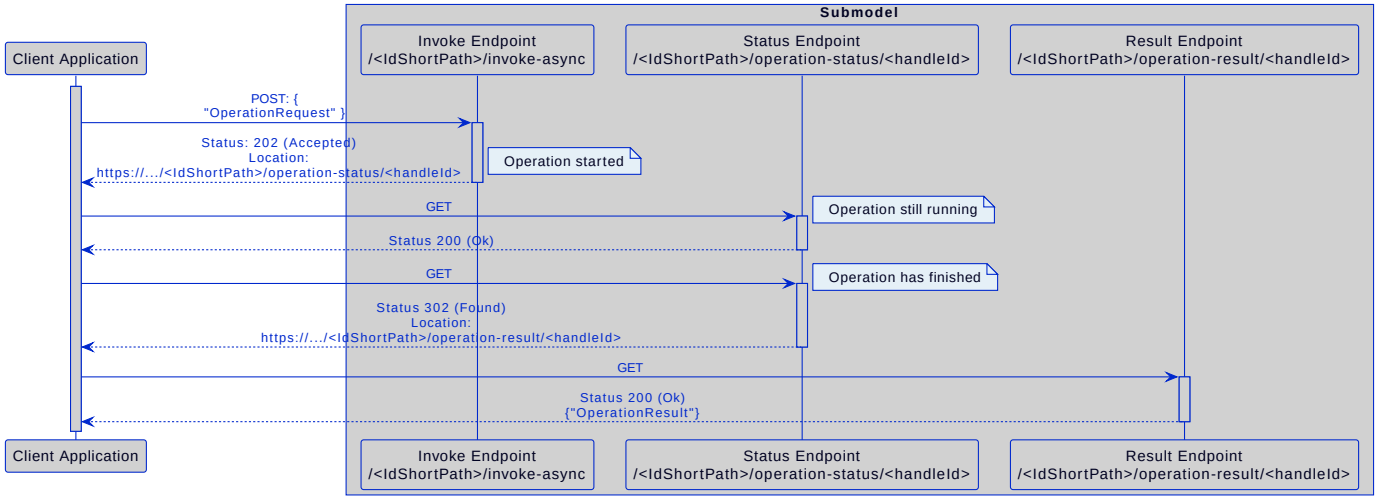


Figure 8. Sequence for asynchronous invocations of the SubmodelElement 'Operation'

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 8). 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.

Bulk Operations

This chapter provides a description of the Bulk APIs. The Bulk APIs are designed to facilitate efficient and scalable operations on a large number of assets within the AAS. This chapter outlines the key concepts, functionalities, and guidelines for implementing and utilizing the Bulk APIs. Bulk operations are intended for the simultaneous manipulation of many objects. Due to the size of bulk requests, it can be expected that the usual execution times takes significantly longer than for non-bulk requests. To avoid frequent timeout errors, the AAS API only defines asynchronous bulk operations for HTTP. The pattern for these operations follows the one introduced in Asynchronous Invocation of the SubmodelElement "Operation".

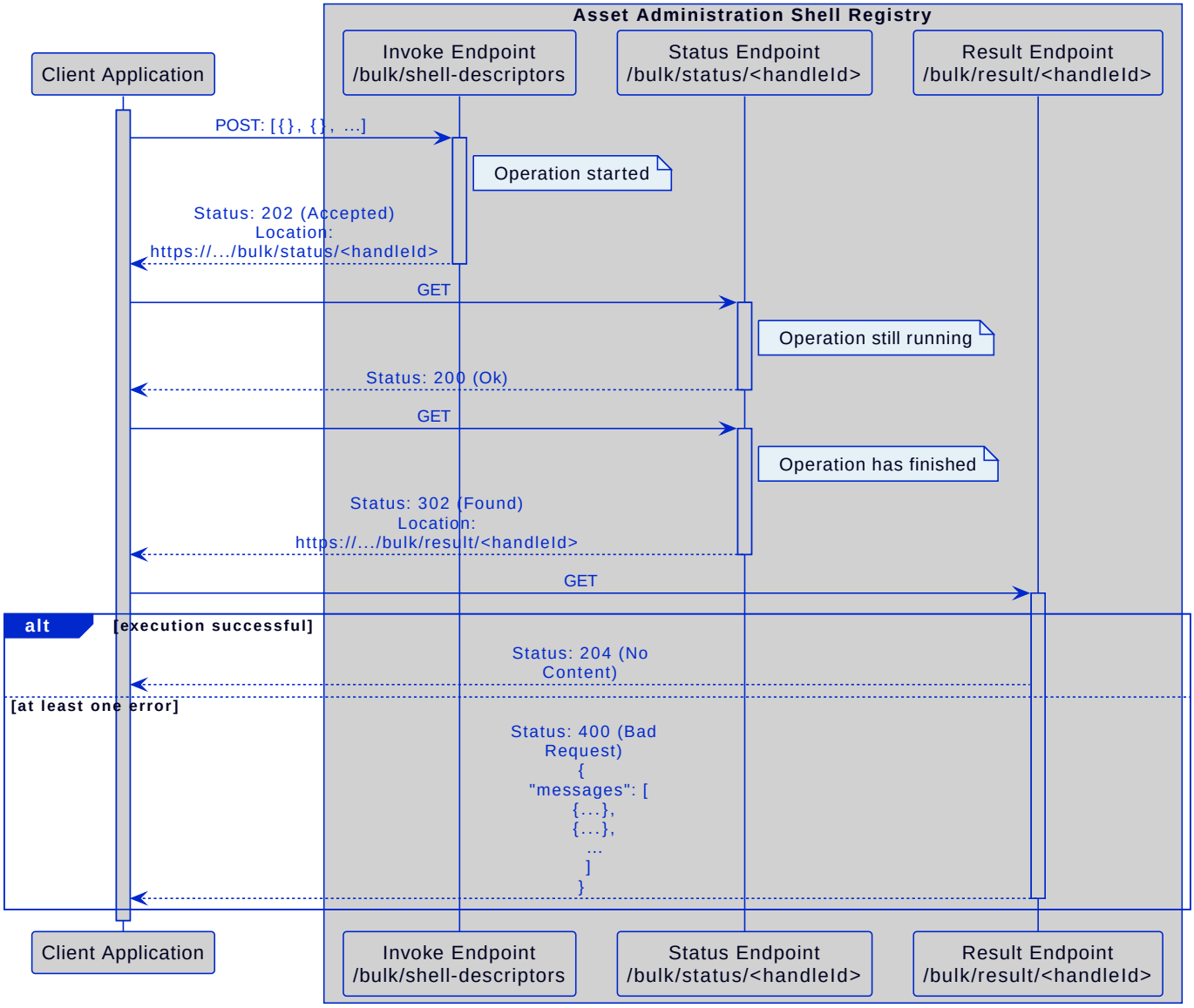


Figure 9. Sequence of a Bulk API Operation

Bulk requests are solely sent to bulk endpoints, which must contain the "/bulk/" path segment. A server may serve bulk endpoints together with non-bulk endpoints. However, in case of available bulk operations, the server must also provide the so-called bulk status (/bulk/status/{handle-id}) and bulk result (/bulk/result/{handle-id}) endpoints. A client executing a bulk request will retrieve the location of the status endpoint through the Location header of the response (see Figure 9). As long as the request is processed, the status endpoint responds with "OK" with an optional information for the client when it shall ask again ("Retry-After"). As soon as the server was able to process the request, the status endpoint provides a redirect to the location of the result. The result endpoint may either signal the client a success of the operation without any additional content, or an error together with a detailed error message in the body. A server may remove information about the result of a bulk request after a certain amount of time. A client requesting a bulk result may retrieve a ClientErrorResourceNotFound even though the bulk request has been processed if a certain amount of time has passed between the sending of the bulk request and the retrieval of the result.

> Note: A server may remove a result object after a client has retrieved it at least once.

To ensure interoperability and consistency, the following guidelines should be adhered to when implementing and utilizing the Bulk APIs:

1. Request Validation

Bulk requests should be validated against the AAS data model to ensure compliance with the defined asset structure

and constraints. Invalid or malformed requests should be rejected with appropriate error codes (see Mapping of Status Codes) and error messages. In case the validation of the request as a whole would take too long, a server may accept the request at first hand but provide the – potential – validation result as the result object. In particular, a client must not expect a correct request solely because the server has accepted the request at the first time.

2. Atomicity

Bulk operations should be performed atomically, ensuring that either all operations within a bulk request are successfully executed, or none of them are. If any individual operation fails, the entire bulk operation should be rolled back, and an appropriate error response should be generated. A client must not expect that any part of the request has been persisted.

3. Error Handling

Bulk responses should provide detailed information about the outcome of each individual operation within the bulk request. In case of failures, error codes and error messages should be included to aid in troubleshooting and error resolution. For each failed operation, at least one item in the message array of the result object shall be provided, linking unambiguously to the problematic incoming request item. A client must not expect that the list of incorrect items is complete, as the server may terminate the execution already when the first error appears.

Querying

Similar to bulk requests, queries are sent to the defined path segments "/query". Clients must use the POST to send queries to the server, with the serialized query in the request body. The result of a query, either the in the form of the result set or an error message, is returned in the http response. Asynchronous behavior by the AAS HTTP endpoints is not defined.

The serialised query has to match the grammar as specified in Clause Serialisation. If a server retrieves a mal-formed query, it must respond with a ClientErrorBadRequest message.

Servers may provide additional information about the perceived error, however, they are not obliged to do so.

AAS Servers expose their capability to support the AAS Querying by providing the according profile identifiers through their "/description" endpoint. If a server exposes a profile identifier in this manner, it must support all querying features that are contained in this profile. An AAS server might support more than one querying profile.

Even though the expressiveness of AAS queries have been limited to a basic amount, clients might still create a critical load at AAS servers, e.g., due to the amount of queries or certain time-expensive constructs. In such cases, servers can abort queries after a certain amount of time but have to inform the client accordingly.

Serialisation

The AAS Query Language for HTTP interactions is serialised in a JSON object. It has to be compliant to the following JSON Schema:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "JSON Schema for AAS Queries",
  "description": "This schema validates AAS Queries.",
  "$ref": "#/definitions/Query",
  "definitions": {
    "standardString": {
      "type": "string",
      "pattern": "^(?!\\$).*"
    },
    "modelStringPattern": {
```

```
      "type": "string",
      "pattern":
"^((?:\\$aas#(?:idShort|id|assetInformation\\.assetKind|assetInformation\\.assetType|assetI
nformation\\.globalAssetId|assetInformation\\.(?:specificAssetIds(\\[[0-
9]*\\])(?:\\.(?:name|value|externalSubjectId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(?:type|value
))?)?)?)|submodels\\.(?:type|keys\\[\\d*\\](?:\\.(?:type|value))?))|submodels\\.(type|keys\
\[\\d*\\](?:\\.(type|value))?))))|(?:\\$sm#(?:semanticId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(t
ype|value))?)?|idShort|id))|(?:\\$sme(?:\\.[a-zA-Z][a-zA-Z0-9_]*(\\[[0-9]*\\])?(?:\\.[a-zA-
Z][a-zA-Z0-9_]*(\\[[0-
9]*\\])?)*)?#(?:semanticId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(type|value))?)?|idShort|value|
valueType|language))|(?:\\$cd#(?:idShort|id))|(?:\\$aasdesc#(?:idShort|id|assetKind|assetTy
pe|globalAssetId|specificAssetIds(\\[[0-
9]*\\])?(?:\\.(name|value|externalSubjectId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(type|value))?
)?)?)|endpoints(\\[[0-
9]*\\])\\.(interface|protocolinformation\\.href)|submodelDescriptors(\\[[0-
9]*\\])\\.(semanticId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(type|value))?)?|idShort|id|endpoint
s(\\[[0-
9]*\\])\\.(interface|protocolinformation\\.href))))|(?:\\$smdesc#(?:semanticId(?:\\.type|\\
.keys\\[\\d*\\](?:\\.(type|value))?)?|idShort|id|endpoints(\\[[0-
9]*\\])\\.(interface|protocolinformation\\.href))))$"
    },
    "hexLiteralPattern": {
      "type": "string",
      "pattern": "^16#[0-9A-F]+$"
    },
    "dateTimeLiteralPattern": {
      "type": "string",
      "format": "date-time"
    },
    "timeLiteralPattern": {
      "type": "string",
      "pattern": "^[0-9][0-9]:[0-9][0-9](:[0-9][0-9])?$"
    },
    "Value": {
      "type": "object",
      "properties": {
        "$field": {
          "$ref": "#/definitions/modelStringPattern"
        },
        "$strVal": {
          "$ref": "#/definitions/standardString"
        },
        "$numVal": {
          "type": "number"
        },
        "$hexVal": {
          "$ref": "#/definitions/hexLiteralPattern"
        },
        "$dateTimeVal": {
          "$ref": "#/definitions/dateTimeLiteralPattern"
        },
        "$timeVal": {
```

```
        "$ref": "#/definitions/timeLiteralPattern"
      },
      "$boolean": {
        "type": "boolean"
      },
      "$strCast": {
        "$ref": "#/definitions/Value"
      },
      "$numCast": {
        "$ref": "#/definitions/Value"
      },
      "$hexCast": {
        "$ref": "#/definitions/Value"
      },
      "$boolCast": {
        "$ref": "#/definitions/Value"
      },
      "$dateTimeCast": {
        "$ref": "#/definitions/Value"
      },
      "$timeCast": {
        "$ref": "#/definitions/Value"
      },
      "$dayOfWeek": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      },
      "$dayOfMonth": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      },
      "$month": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      },
      "$year": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      }
    },
    "oneOf": [
      {
        "required": [
          "$field"
        ]
      },
      {
        "required": [
          "$strVal"
        ]
      },
      {
        "required": [
          "$numVal"
        ]
      },
```

```
        {
          "required": [
            "$hexVal"
          ]
        },
        {
          "required": [
            "$dateTimeVal"
          ]
        },
        {
          "required": [
            "$timeVal"
          ]
        },
        {
          "required": [
            "$boolean"
          ]
        },
        {
          "required": [
            "$strCast"
          ]
        },
        {
          "required": [
            "$numCast"
          ]
        },
        {
          "required": [
            "$hexCast"
          ]
        },
        {
          "required": [
            "$boolCast"
          ]
        },
        {
          "required": [
            "$dateTimeCast"
          ]
        },
        {
          "required": [
            "$timeCast"
          ]
        },
        {
          "required": [
```

```
            "$dayOfWeek"
          ]
        },
        {
          "required": [
            "$dayOfMonth"
          ]
        },
        {
          "required": [
            "$month"
          ]
        },
        {
          "required": [
            "$year"
          ]
        }
      ],
      "additionalProperties": false
    },
    "stringValue": {
      "type": "object",
      "properties": {
        "$field": {
          "$ref": "#/definitions/modelStringPattern"
        },
        "$strVal": {
          "$ref": "#/definitions/standardString"
        },
        "$strCast": {
          "$ref": "#/definitions/Value"
        }
      },
      "oneOf": [
        {
          "required": [
            "$field"
          ]
        },
        {
          "required": [
            "$strVal"
          ]
        },
        {
          "required": [
            "$strCast"
          ]
        }
      ],
      "additionalProperties": false
```

```
    },
    "comparisonItems": {
      "type": "array",
      "minItems": 2,
      "maxItems": 2,
      "items": {
        "$ref": "#/definitions/Value"
      }
    },
    "stringItems": {
      "type": "array",
      "minItems": 2,
      "maxItems": 2,
      "items": {
        "$ref": "#/definitions/stringValue"
      }
    },
    "matchExpression": {
      "type": "object",
      "properties": {
        "$match": {
          "type": "array",
          "minItems": 1,
          "items": {
            "$ref": "#/definitions/matchExpression"
          }
        },
        "$eq": {
          "$ref": "#/definitions/comparisonItems"
        },
        "$ne": {
          "$ref": "#/definitions/comparisonItems"
        },
        "$gt": {
          "$ref": "#/definitions/comparisonItems"
        },
        "$ge": {
          "$ref": "#/definitions/comparisonItems"
        },
        "$lt": {
          "$ref": "#/definitions/comparisonItems"
        },
        "$le": {
          "$ref": "#/definitions/comparisonItems"
        },
        "$contains": {
          "$ref": "#/definitions/stringItems"
        },
        "$starts-with": {
          "$ref": "#/definitions/stringItems"
        },
        "$ends-with": {
```

```
        "$ref": "#/definitions/stringItems"
      },
      "$regex": {
        "$ref": "#/definitions/stringItems"
      },
      "$boolean": {
        "type": "boolean"
      }
    },
    "oneOf": [
      {
        "required": [
          "$eq"
        ]
      },
      {
        "required": [
          "$ne"
        ]
      },
      {
        "required": [
          "$gt"
        ]
      },
      {
        "required": [
          "$ge"
        ]
      },
      {
        "required": [
          "$lt"
        ]
      },
      {
        "required": [
          "$le"
        ]
      },
      {
        "required": [
          "$contains"
        ]
      },
      {
        "required": [
          "$starts-with"
        ]
      },
      {
        "required": [
```

```
            "$ends-with"
          ]
        },
        {
          "required": [
            "$regex"
          ]
        },
        {
          "required": [
            "$boolean"
          ]
        },
        {
          "required": [
            "$match"
          ]
        }
      ],
      "additionalProperties": false
    },
    "logicalExpression": {
      "type": "object",
      "properties": {
        "$and": {
          "type": "array",
          "minItems": 2,
          "items": {
            "$ref": "#/definitions/logicalExpression"
          }
        },
        "$match": {
          "type": "array",
          "minItems": 1,
          "items": {
            "$ref": "#/definitions/matchExpression"
          }
        },
        "$or": {
          "type": "array",
          "minItems": 2,
          "items": {
            "$ref": "#/definitions/logicalExpression"
          }
        },
        "$not": {
          "$ref": "#/definitions/logicalExpression"
        },
        "$eq": {
          "$ref": "#/definitions/comparisonItems"
        },
        "$ne": {
```

```
        "$ref": "#/definitions/comparisonItems"
      },
      "$gt": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$ge": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$lt": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$le": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$contains": {
        "$ref": "#/definitions/stringItems"
      },
      "$starts-with": {
        "$ref": "#/definitions/stringItems"
      },
      "$ends-with": {
        "$ref": "#/definitions/stringItems"
      },
      "$regex": {
        "$ref": "#/definitions/stringItems"
      },
      "$boolean": {
        "type": "boolean"
      }
    },
    "oneOf": [
      {
        "required": [
          "$and"
        ]
      },
      {
        "required": [
          "$or"
        ]
      },
      {
        "required": [
          "$not"
        ]
      },
      {
        "required": [
          "$eq"
        ]
      },
      {
```

```
        "required": [
          "$ne"
        ]
      },
      {
        "required": [
          "$gt"
        ]
      },
      {
        "required": [
          "$ge"
        ]
      },
      {
        "required": [
          "$lt"
        ]
      },
      {
        "required": [
          "$le"
        ]
      },
      {
        "required": [
          "$contains"
        ]
      },
      {
        "required": [
          "$starts-with"
        ]
      },
      {
        "required": [
          "$ends-with"
        ]
      },
      {
        "required": [
          "$regex"
        ]
      },
      {
        "required": [
          "$boolean"
        ]
      },
      {
        "required": [
          "$match"
```

```
        ]
      }
    ],
    "additionalProperties": false
  },
  "attributeItem": {
    "oneOf": [
      {
        "required": [
          "CLAIM"
        ]
      },
      {
        "required": [
          "GLOBAL"
        ]
      },
      {
        "required": [
          "REFERENCE"
        ]
      }
    ],
    "properties": {
      "CLAIM": {
        "type": "string"
      },
      "GLOBAL": {
        "type": "string",
        "enum": [
          "LOCALNOW",
          "UTCNOW",
          "CLIENTNOW",
          "ANONYMOUS"
        ]
      },
      "REFERENCE": {
        "type": "string"
      }
    },
    "additionalProperties": false
  },
  "objectItem": {
    "oneOf": [
      {
        "required": [
          "ROUTE"
        ]
      },
      {
        "required": [
          "IDENTIFIABLE"
```

```
          ]
        },
        {
          "required": [
            "REFERABLE"
          ]
        },
        {
          "required": [
            "FRAGMENT"
          ]
        },
        {
          "required": [
            "DESCRIPTOR"
          ]
        }
      ],
      "properties": {
        "ROUTE": {
          "type": "string"
        },
        "IDENTIFIABLE": {
          "type": "string"
        },
        "REFERABLE": {
          "type": "string"
        },
        "FRAGMENT": {
          "type": "string"
        },
        "DESCRIPTOR": {
          "type": "string"
        }
      },
      "additionalProperties": false
    },
    "rightsEnum": {
      "type": "string",
      "enum": [
        "CREATE",
        "READ",
        "UPDATE",
        "DELETE",
        "EXECUTE",
        "VIEW",
        "ALL",
        "TREE"
      ],
      "additionalProperties": false
    },
    "ACL": {
```

```
      "type": "object",
      "properties": {
        "ATTRIBUTES": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/attributeItem"
          }
        },
        "USEATTRIBUTES": {
          "type": "string"
        },
        "RIGHTS": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/rightsEnum"
          }
        },
        "ACCESS": {
          "type": "string",
          "enum": [
            "ALLOW",
            "DISABLED"
          ]
        }
      },
      "required": [
        "RIGHTS",
        "ACCESS"
      ],
      "oneOf": [
        {
          "required": [
            "ATTRIBUTES"
          ]
        },
        {
          "required": [
            "USEATTRIBUTES"
          ]
        }
      ],
      "additionalProperties": false
    },
    "AccessPermissionRule": {
      "type": "object",
      "properties": {
        "ACL": {
          "$ref": "#/definitions/ACL"
        },
        "USEACL": {
          "type": "string"
        },
```

```
      "OBJECTS": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/objectItem"
        },
        "additionalProperties": false
      },
      "USEOBJECTS": {
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "FORMULA": {
        "$ref": "#/definitions/logicalExpression",
        "additionalProperties": false
      },
      "USEFORMULA": {
        "type": "string"
      },
      "FRAGMENT": {
        "type": "string"
      },
      "FILTER": {
        "$ref": "#/definitions/logicalExpression",
        "additionalProperties": false
      },
      "USEFILTER": {
        "type": "string"
      }
    },
    "oneOf": [
      {
        "required": [
          "ACL"
        ]
      },
      {
        "required": [
          "USEACL"
        ]
      }
    ],
    "oneOf": [
      {
        "required": [
          "OBJECTS"
        ]
      },
      {
        "required": [
          "USEOBJECTS"
```

```
          ]
        }
      ],
      "oneOf": [
        {
          "required": [
            "FORMULA"
          ]
        },
        {
          "required": [
            "USEFORMULA"
          ]
        }
      ],
      "additionalProperties": false
    },
    "Query": {
      "type": "object",
      "properties": {
        "$select": {
          "type": "string",
          "pattern": "^id$"
        },
        "$condition": {
          "$ref": "#/definitions/logicalExpression"
        }
      },
      "required": [
        "$condition"
      ],
      "additionalProperties": false
    },
    "AllAccessPermissionRules": {
      "type": "object",
      "properties": {
        "DEFATTRIBUTES": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "name": {
                "type": "string"
              },
              "attributes": {
                "type": "array",
                "items": {
                  "$ref": "#/definitions/attributeItem"
                }
              }
            },
            "required": [
```

```
        "name",
        "attributes"
      ],
      "additionalProperties": false
    }
  },
  "DEFACLS": {
    "type": "array",
    "items": {
      "type": "object",
      "properties": {
        "name": {
          "type": "string"
        },
        "acl": {
          "$ref": "#/definitions/ACL"
        }
      },
      "required": [
        "name",
        "acl"
      ],
      "additionalProperties": false
    }
  },
  "DEFOBJECTS": {
    "type": "array",
    "items": {
      "type": "object",
      "properties": {
        "name": {
          "type": "string"
        },
        "objects": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/objectItem"
          }
        },
        "USEOBJECTS": {
          "type": "array",
          "items": {
            "type": "string"
          }
        }
      },
      "required": [
        "name"
      ],
      "oneOf": [
        {
          "required": [
```

```
                "objects"
              ]
            },
            {
              "required": [
                "USEOBJECTS"
              ]
            }
          ],
          "additionalProperties": false
        }
      },
      "DEFFORMULAS": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "name": {
              "type": "string"
            },
            "formula": {
              "$ref": "#/definitions/logicalExpression"
            }
          },
          "required": [
            "name",
            "formula"
          ],
          "additionalProperties": false
        }
      },
      "rules": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/AccessPermissionRule"
        }
      }
    },
    "required": [
      "rules"
    ],
    "additionalProperties": false
  }
 }
}
```

In case a syntax error is discovered by a receiver of an AAS query object, the query as a whole must be rejected with a ClientErrorBadRequest. A receiver may accept queries that extend the AAS Query Language, however, a Client must not expect any behavior or feature that are not included in the specific Query Language version that a server supports.

Endpoints

AAS queries are sent via POST requests, delivering the serialised query JSON in the request body, to the reserved query endpoint of an AAS service. This endpoint begins with the /query path segment, followed by a service-specific segment. For the Asset Administration Shell Repository, this is e.g. /query/shells. It is recommended that a server exposes its supported query profiles through the /description endpoint.

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 section 6.1: https://datatracker.ietf.org/doc/html/rfc7231#section-6)

Table 14. Status Code Mapping for HTTP

	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

Message Type	Meaning	HTTP status code	Explanation
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 Profiles

Figure 1 in Clause *General* 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/V3.1_SSP-002” contains the API Operation “GetAllSubmodels” but not “PostSubmodelElementByPath +”, while the more comprehensive “RepositoryServiceSpecification/V3.1_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 15. 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

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
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

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/<major version>/<minor version>/<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. It is sufficient to only expose the latest service specification identifier in case a server supports different minor versions of a service specification at the same time. For instance, if both "https://admin-shell.io/aas/API/3/0/SubmodelRegistryServiceSpecification/SSP-002" and "https://admin-shell.io/aas/API/3/1/SubmodelRegistryServiceSpecification/SSP-002" are supported, the server may only include "https://admin-shell.io/aas/API/3/1/SubmodelRegistryServiceSpecification/SSP-002" in its profiles list.

Clients shall understand that lower minor versions of service specifications are supported even if only one service specification identifier is provided.

Different to minor versions, it is not sufficient to only expose the latest major version of a service specification. A client must not expect support for lower major versions.

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**

Name:	AAS Full Profile
APIs and API Operations	*Asset Administration Shell API:* GetAssetAdministrationShell PutAssetAdministrationShell GetAllSubmodelReferences PostSubmodelReference DeleteSubmodelReference GetAssetInformation PutAssetInformation GetThumbnail PutThumbnail DeleteThumbnail *Submodel API as superpath:* GetSubmodelById GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath PutFileByPath DeleteFileByPath PutSubmodelById PatchSubmodelById PostSubmodelElement PostSubmodelElementByPath +[PostSubmodelElementByPath +] PutSubmodelElementByPath +[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/V3.1.0_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/3/1/AssetAdministrationShellServiceSpecification/SSP-002
Feature	**Appearance**
API Operations	*Asset Administration Shell API:* GetAssetAdministrationShell GetAllSubmodelReferences GetAssetInformation GetThumbnail *Submodel API as superpath:* GetSubmodelById 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/V3.1.0_SSP-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 PatchSubmodelElementByPath 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/V3.1.0_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**

Name:	Submodel Read Profile
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/V3.1.0_SSP-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/V3.1.0_SSP-003

AASX File Server Service Specification

Service Specification / Profiles	Description

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/V3.1.0_SSP-001

Asset Administration Shell Registry Service Specification

Service Specification / Profiles	Description
AssetAdministrationShellRegistryServiceSpecification/SSP-001	Full profile
AssetAdministrationShellRegistryServiceSpecification/SSP-002	Only read operations; is included in the profile AssetAdministrationShellRegistryServiceSpecification/SSP-001.
AssetAdministrationShellRegistryServiceSpecification/SSP-003	Bulk write and delete operations.
AssetAdministrationShellRegistryServiceSpecification/SSP-004	Query operations.
AssetAdministrationShellRegistryServiceSpecification/SSP-005	Only reads operations on AAS Descriptors; is included in the profile AssetAdministrationShellRegistryServiceSpecification/SSP-001 and AssetAdministrationShellRegistryServiceSpecification/SSP-002.

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/V3.1.0_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 GetDescription *Description API:* GetDescription

Name:	AAS Registry Read Profile
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See:
https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRegistryServiceSpecification/V3.1.0_SSP-002

Asset Administration Shell Registry Service Specification – Bulk Profile

Name:	AAS Registry Bulk Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/0/AssetAdministrationShellRegistryServiceSpecification/SSP-003
Feature	**Appearance**
APIs and API Operations	*AAS Registry API:* CreateBulkAssetAdministrationShellDescriptors PutBulkAssetAdministrationShellDescriptorsById DeleteBulkAssetAdministrationShellDescriptorsById *Description API:* GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	Not supported

See:
https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRegistryServiceSpecification/V3.1.0_SSP-003

Asset Administration Shell Registry Service Specification – Query Profile

Name:	AAS Registry Bulk Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/0/AssetAdministrationShellRegistryServiceSpecification/SSP-004
Feature	**Appearance**
APIs and API Operations	*AAS Registry API:* QueryAssetAdministrationShellDescriptors *Description API:* GetDescription

Name:	AAS Registry Bulk Profile
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See: https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRegistryServiceSpecification/V3.1.0_SSP-004

Asset Administration Shell Registry Service Specification – Minimal Read Profile

Name:	AAS Registry Minimal Read Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/0/AssetAdministrationShellRegistryServiceSpecification/SSP-005
Feature	**Appearance**
APIs and API Operations	*AAS Registry API:* GetAllAssetAdministrationShellDescriptors GetAssetAdministrationShellDescriptorById *Description API:* GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See: https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRegistryServiceSpecification/V3.1.0_SSP-005

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.
SubmodelRegistryServiceSpecification/SSP-003	Bulk write and delete operations.
SubmodelRegistryServiceSpecification/SSP-004	Query operations.

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 GetDescription PostSubmodelDescriptor PutSubmodelDescriptorById DeleteSubmodelDescriptorById *Description API:* GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See: https://app.swaggerhub.com/apis/Plattform_i40/SubmodelRegistryServiceSpecification/V3.1.0_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/V3.1.0_SSP-002

Submodel Registry Profile – Bulk Profile

Name:	Submodel Registry Bulk Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/0/SubmodelRegistryServiceSpecification/SSP-003

Name:	Submodel Registry Bulk Profile
Feature	**Appearance**
APIs and API Operations	*Submodel Registry API:* PostBulkSubmodelDescriptors PutBulkSubmodelDescriptorsById DeleteBulkSubmodelDescriptorsById *Description API:* GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	Not Supported

See: https://app.swaggerhub.com/apis/Plattform_i40/SubmodelRegistryServiceSpecification/V3.1.0_SSP-003

Submodel Registry Profile – Query Profile

Name:	Submodel Registry Bulk Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/1/SubmodelRegistryServiceSpecification/SSP-004
Feature	**Appearance**
APIs and API Operations	*Submodel Registry API:* QuerySubmodelDescriptors *Description API:* GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	Supported

See: https://app.swaggerhub.com/apis/Plattform_i40/SubmodelRegistryServiceSpecification/V3.1.0_SSP-004

Discovery Service Specification

Service Specification / Profiles	Description
DiscoveryServiceSpecification/SSP-001	Full feature set
DiscoveryServiceSpecification/SSP-002	only read operations; does not contain the deprecated read operation from DiscoveryServiceSpecification/SSP-001

Discovery Service Specification – Full Profile

Name:	Discovery Service Full Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/1/DiscoveryServiceSpecification/SSP-001
Feature	**Appearance**
API and API Operations	*AAS Basic Discovery API:* GetAllAssetAdministrationShellIdsByAssetLink (<<deprecated>>) SearchAllAssetAdministrationShellIdsByAssetLink GetAllAssetLinksById PostAllAssetLinksById DeleteAllAssetLinksById *Description API:* GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	Not supported

See: https://app.swaggerhub.com/apis/Plattform_i40/DiscoveryServiceSpecification/V3.1.0_SSP-001

Discovery Service Specification – Read Profile

Name:	Discovery Service Full Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/1/DiscoveryServiceSpecification/SSP-002
Feature	**Appearance**
API and API Operations	*AAS Basic Discovery API:* SearchAllAssetAdministrationShellIdsByAssetLink GetAllAssetLinksById *Description API:* GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	Not supported

See: https://app.swaggerhub.com/apis/Plattform_i40/DiscoveryServiceSpecification/V3.1.0_SSP-002

Asset Administration Shell Repository Service Specification

Service Specification / Profiles	Description
AssetAdministrationShellRepositoryServiceSpecification/SSP-001	Full feature set
AssetAdministrationShellRepositoryServiceSpecification/SSP-002	Only read operations; is included in the profile AssetAdministrationShellRepositoryServiceSpecification/SSP-001
AssetAdministrationShellRepositoryServiceSpecification/SSP-003	Query operations

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

Name:	AAS Repository Full Profile
API and API Operations	*AAS Repository API:* GetAllAssetAdministrationShells GetAssetAdministrationShellById GetAllAssetAdministrationShellsByAssetId GetAllAssetAdministrationShellsByIdShort PostAssetAdministrationShell PutAssetAdministrationShellById DeleteAssetAdministrationShellById *AAS API by superpath:* GetAllSubmodelReferences PostSubmodelReference DeleteSubmodelReference GetAssetInformation PutAssetInformation GetThumbnail PutThumbnail DeleteThumbnail *Submodel Repository API by superpath:* GetAllSubmodels GetSubmodelById GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort PostSubmodel PutSubmodelById PatchSubmodelById DeleteSubmodelById *Submodel API by superpath:* GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath PutFileByPath DeleteFileByPath PostSubmodelElement PostSubmodelElementByPath + PutSubmodelElementByPath + PatchSubmodelElementByPath PatchSubmodelElementByPath InvokeOperationSync InvokeOperationAsync GetOperationAsyncStatus GetOperationAsyncResult *AAS Serialization API:* GenerateSerializationByIds *Description API:* GetDescription
SerializationModifier	Level: Core, Deep Content: Normal, Metadata, Value, Reference, Path Extent: WithBLOBValue, WithoutBLOBValue

Name:	AAS Repository Full Profile
SerializationFormat	JSON
Pagination	supported

See:
https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRepositoryServiceSpecification/V3.1.0_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 [specification::interfaces:::GetAllAssetAdministrationShellsByIdShort] *AAS API by superpath:* GetAssetAdministrationShell GetAllSubmodelReferences GetAllSubmodelReferences GetAllSubmodelReferences *Submodel Repository API by superpath:* GetAllSubmodelReferences GetAllSubmodelReferences GetAllSubmodelReferences GetAllSubmodelReferences *Submodel API by superpath:* GetAllSubmodelElements GetSubmodelElementByPath [specification::interfaces:::GetFileByPath] *Serialization API:* GenerateSerializationByIds *Description API:* GetDescription
SerializationModifier	Level: Core, Deep Content: Normal, Metadata, Value, Reference, Path Extent: WithBLOBValue, WithoutBLOBValue
SerializationFormat	JSON

Name:	AAS Repository Read Profile
Pagination	supported

See:
https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRepositoryServiceSpecification/V3.1.0_SSP-002

Asset Administration Shell Repository Service Specification – Query Profile

Name:	AAS Repository Query Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/0/AssetAdministrationShellRepositoryServiceSpecification/SSP-003
Feature	**Appearance**
API and API Operations	*AAS Repository API:* QueryAssetAdministrationShells *Description API:* GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See:
https://app.swaggerhub.com/apis/Plattform_i40/AssetAdministrationShellRepositoryServiceSpecification/V3.1.0_SSP-003

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

SubmodelRepositoryServiceSpecification/SSP-005	Query operations

Submodel Repository - Full Profile

Name:	Submodel Repository Full Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/0/SubmodelRepositoryServiceSpecification/SSP-001
Feature	**Appearance**
API and API Operations	*Submodel Repository API:* GetAllSubmodels GetSubmodelById GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort PostSubmodel PutSubmodelById PatchSubmodelById DeleteSubmodelById *Submodel API by superpath:* GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath PutFileByPath DeleteFileByPath PostSubmodelElement PostSubmodelElementByPath + PutSubmodelElementByPath + PatchSubmodelElementByPath PatchSubmodelElementByPath 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/V3.1.0_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/V3.1.0_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**.

pass:q[[underline]#Constraint AASa-003#: A service implementing the SubmodelRepositoryServiceSpecification/SSP-003 must not accept or provide any _Submodel_ with the attribute https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.1/spec-metamodel/common.html#ModellingKind.

> 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/SubmodelRepositoryServiceSpecification/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 PostSubmodelElement PostSubmodelElementByPath + PutSubmodelElementByPath + PatchSubmodelElementByPath PatchSubmodelElementByPath *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/V3.1.0_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 SubmodelRepositoryServiceSpecification/SSP-004 must not accept or provide any _Submodel with the attribute <a href="https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.1/spec-metamodel/common.html#ModellingKind"

class="bare">https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.1/spec-metamodel/common.html#ModellingKind.

Note: due to Constraint AASa-004, SubmodelServiceSpecification/SSP-004 can not be combined with SubmodelServiceSpecification/SSP-001 or SubmodelServiceSpecification/SSP-002 as SubmodelServiceSpecification/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/SubmodelRepositoryServiceSpecification/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/V3.1.0_SSP-004

Submodel Repository Service Specification – Query Profile

Name:	**Submodel Repository Query Profile**
Profile Identifier:	https://admin-shell.io/aas/API/3/0/SubmodelRepositoryServiceSpecification/SSP-005
Feature	**Appearance**

Name:	Submodel Repository Query Profile
API and API Operations	*AAS Repository API:* QuerySubmodels *Description API:* GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See: https://app.swaggerhub.com/apis/Plattform_i40/SubmodelRepositoryServiceSpecification/V3.1.0_SSP-005

Concept Description Repository Service Specification

Service Specification / Profiles	Description
ConceptDescriptionRepositoryServiceSpecification/SSP-001	Full feature set
ConceptDescriptionRepositoryServiceSpecification/SSP-002	Query operations

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

Name:	Concept Description Repository Full Profile
Pagination	Supported

See: https://app.swaggerhub.com/apis/Plattform_i40/ConceptDescriptionRepositoryServiceSpecification/V3.1.0_SSP-001

Concept Description Repository Service Specification – Query Profile

Name:	Concept Description Repository Query Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/0/ConceptDescriptionRepositoryServiceSpecification/SSP-002
Feature	**Appearance**
API and API Operations	*ConceptDescription Repository API:* QueryConceptDescriptions *Description API:* GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See: https://app.swaggerhub.com/apis/Plattform_i40/ConceptDescriptionRepositoryServiceSpecification/V3.1.0_SSP-002

Interactions

Interactions describe the sequence of calls of operations by a client application to achieve a defined goal in a use case. In the following, exemplary interaction sequences are depicted.

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

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 has 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.
- Several ways are possible to get to know the endpoints for the infrastructure interfaces of AAS Discovery and/or Registries before calling the APIs. Some examples:
  - A discovery service for dataspace connectors as well as endpoints for AAS services can be found as part of the registered applications on a connector, as e.g. specified in the Catena-X dataspace.
  - The endpoints can be configured in the application during start or dynamically at runtime by a user.
- The asset IDs (either the global asset ID or some specific asset IDs) are known to the calling application.
- Access to any API is allowed only if the caller is authenticated.

- Response to any API call takes access rules as defined for the services into account, e.g. access rules for the AASs, the Submodels, or SubmodelElements within a Submodel.

An interaction typically starts with a client application using the discovery service to get the IDs of the AAS representing the asset under consideration. In the second step the AAS Registry is called to get the relevant endpoints for the AASs found. The AAS Registry provides the AAS Descriptor object belonging to this AAS ID (and thus the asset ID used for discovery) and containing the Submodel Descriptors of the Submodels, which are part of the related AAS.

As explained in Clause Descriptor, there are two ways of using the AssetAdministrationShellDescriptor when registering an Asset Administration Shell:

1. Either add the endpoint of the Asset Administration Shell (usage by client see Figure 11)
2. Or add the endpoints of its Submodels (usage by client see Figure 10)

There is also a third way to combine both. In this case the endpoint of the Asset Administration Shell and the information that can be retrieved at this endpoint is the master and a separate Submodel Registry would provide the Submodel endpoints as single source of truth.

> Note: It is not recommended to combine the two approaches in one and the same AAS Registry. An AAS Registry either should always contain the AAS Endpoint or the AAS Endpoint should always be omitted and Submodel endpoints shall be added.

In the second case, the lookup of the Submodel IDs and the Submodel endpoints can be skipped (compare Figure 10 and Figure 11). No separate AAS Service or Submodel Registry Service needs to be offered to the Client. From the point of view of a data provider the Submodel Registry interfaces are still needed to add new SubmodelDescriptors to an existing AssetAdministrationShellDescriptor.

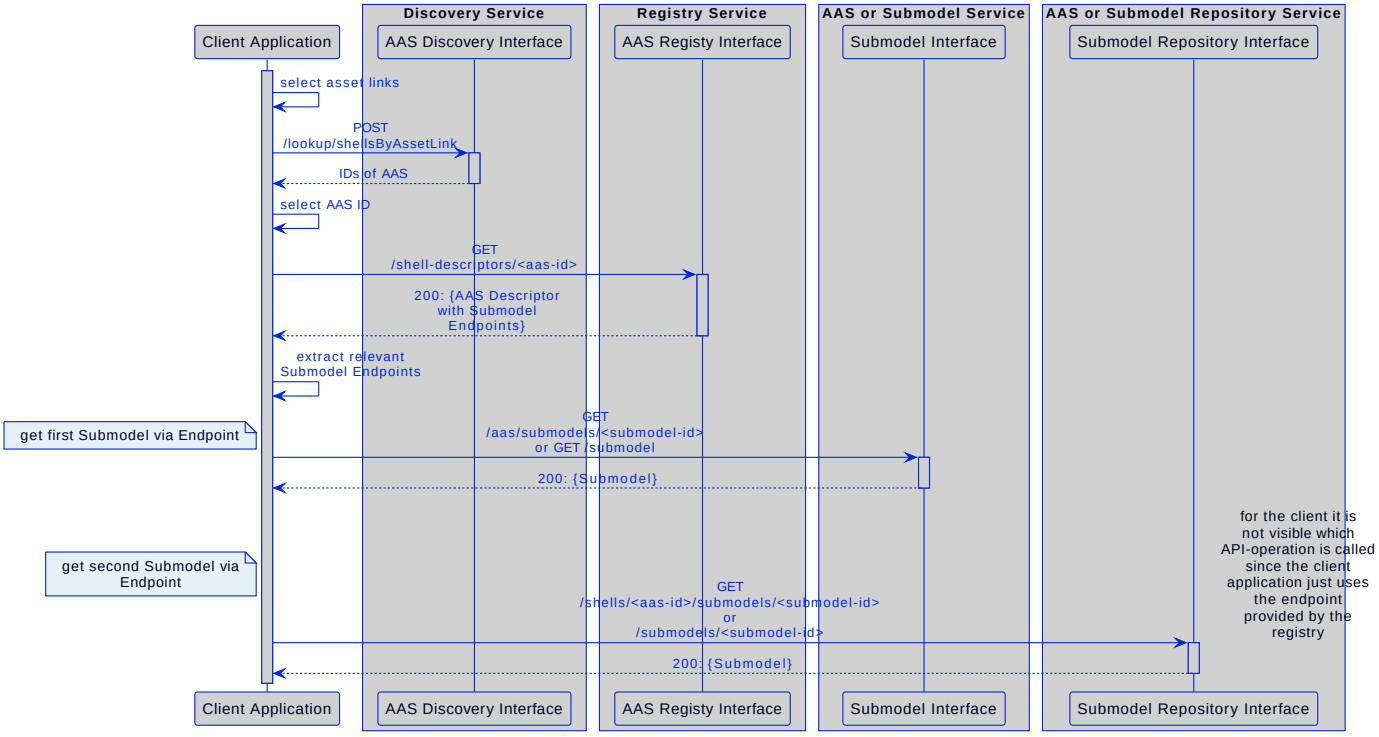


Figure 10. Interaction for Client Application using Submodel Endpoints

For accessing Submodels there are two different ways to do so:

1. Access a Submodel via an AAS or Submodel Server
2. Access a Submodel via a standardized Repository, either AAS or Submodel Repository

In the sequence shown in Figure 10, the first submodel is get via an AAS or Submodel interface whereas the second submodel is get via an AAS or Submodel Repository Service.

Note: The client application just uses the endpoint as provided in the Registry. Thus, for the client there is no difference in the interaction with a pure submodel server or a submodel repository server.

Figure 11 shows a sequence with a Registry providing AAS endpoints. In this case the client needs to look up the corresponding submodel IDs in an AAS or AAS Repository Service first. After looking up the relevant submodel IDs the client calls the Submodel Registry to get the Submodel Descriptors containing the endpoints of the submodels.

Note: If the AAS or AAS Repository Service is used to get the submodel IDs it is recommended that the data provider adds a referredSemanticId to the references of the submodels. Otherwise, it is not possible for the client to decide which of the submodels is relevant. This is typically done via the semanticId of the submodel.

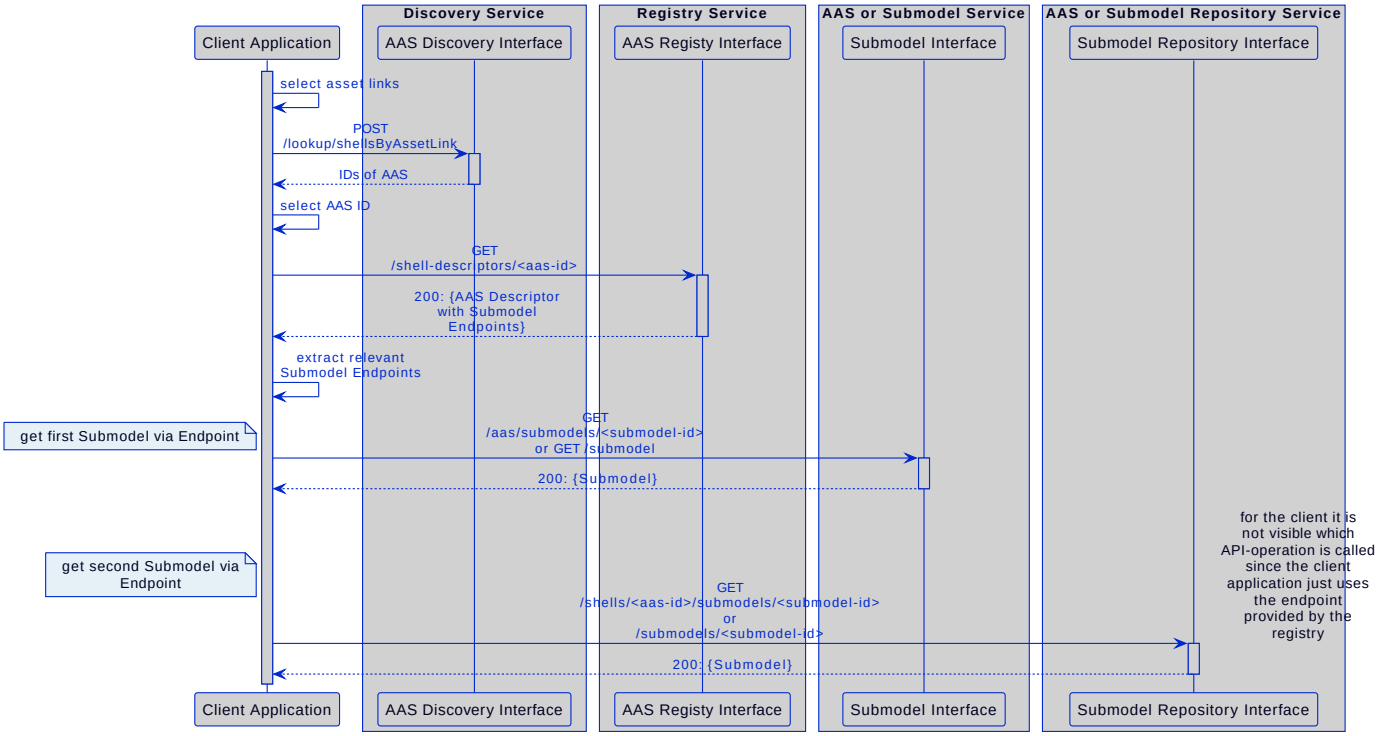


Figure 11. Interaction for Client Application using AAS Endpoints and Repositories

The difference between Interface and API Operations is outlined in Figure 12. This sequence translates the interaction on the interface level of Figure 10, 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. "GetSubmodelById" can be realized by either "GET /shells/<aas-id>/submodels/<submodel-id>" within an AAS Repository or "GET /submodels/<submodel-id>" within a Submodel Repository. The returned objects are shortened for better readability.

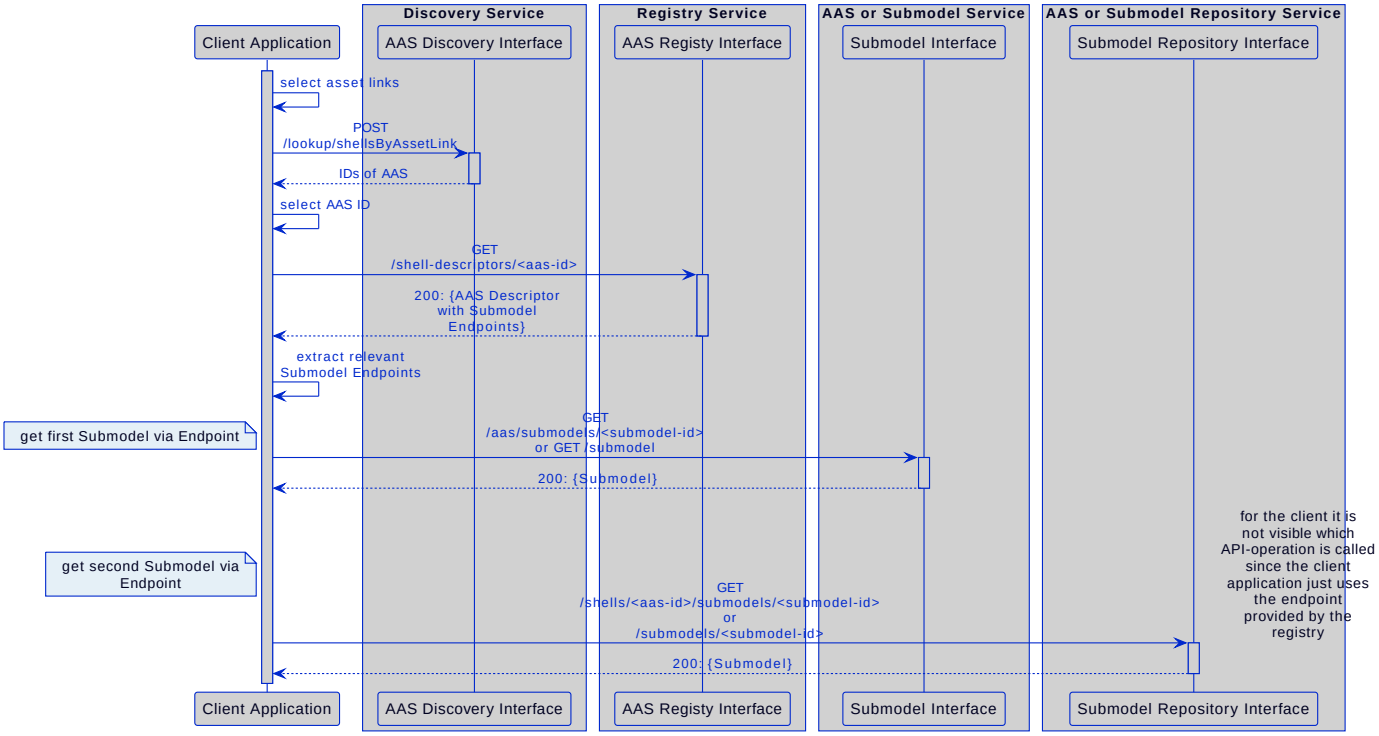
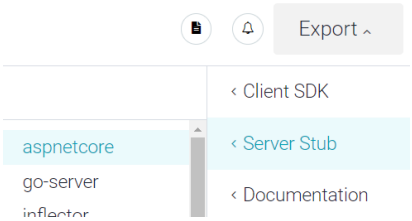


Figure 12. Interaction for Client Application using Submodel Endpoints (for HTTP API Operations)

API Code Generation

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



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 PutAASXByPackageId 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.

[1] http://semver.org

[2] see Chapter 2.4 of RFC 8977

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.

Annex

SerializationModifier Examples

SerializationModifier Examples

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.

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",
	"semanticId": {
	"keys": [{
	"type":
	"GlobalReference",
	"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"
		}
		}]
		}

<table>
<tr><th></th><th>Deep (default)</th><th>Core</th></tr>
<tr><td>Metadata</td><td>If applied to the Submodel (no “Level” modifier for “Metadata”):
<pre>{
  "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"
  }
}</pre>If applied to the SubmodelElementCollection, i.e., idShortPath “RotationSpeed”:
<pre>{
    "modelType": "SubmodelElementCollection",
    "idShort": "RotationSpeed",
    "semanticId": {
      "keys": [{
          "type": "GlobalReference",
          "value": "http://purl.org/iot/vocab/iot-taxonomy-
lite#RotationalSpeed"
      }],
      "type": "ExternalReference"
    }
}</pre>If applied to the Property, i.e. idShortPath “RotationSpeed.MaxRotationSpeed”:
<pre>{
   "modelType": "Property",
   "idShort": "DocumentId",
   "category": "PARAMETER",
   "semanticId": {
       "keys": [{
           "type": "GlobalReference",
           "value": "0173-1#02-BAA120#008"
       }] ,
       "type": "ExternalReference"
   },
   "valueType": "xs:int"
}</pre></td><td></td></tr>
</table>

<table>
<tr><th></th><th>Deep (default)</th><th>Core</th></tr>
<tr><td>Value</td><td>If applied to the Submodel:
<pre>{
  "RotationSpeed": {
    "MaxRotationSpeed": 5000
  }
}</pre>If applied to the SubmodelElementCollection, i.e., idShortPath “RotationSpeed”:
<pre>{
  "MaxRotationSpeed": 5000
}</pre>If applied to the Property, i.e. idShortPath “RotationSpeed.MaxRotationSpeed”:
<pre>5000</pre></td><td>If applied to the Submodel:
<pre>{
  "RotationSpeed": {}
}</pre>If applied to the SubmodelElementCollection, i.e., idShortPath “RotationSpeed”:
<pre>{
  "MaxRotationSpeed": 5000
}</pre>If applied to the Property, i.e. idShortPath “RotationSpeed.MaxRotationSpeed”:
<pre>5000</pre></td></tr>
</table>

	Deep (default)	Core

Reference	Not allowed, see Modifier Constraints:
	
	“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 SubmodelElementCollection, i.e. idShortPath "RotationSpeed":
		
		`{ "keys": [ { "type": "Submodel", "value": "http://i40.customer.com/type/1/1/7A7104BDAB57E184" }, { "type": "SubmodelElementCollection", "value": "RotationSpeed" } ], "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" }],`

	Deep (default)	Core

Path	If applied to the Submodel:
	
	Note: IdShortPaths always start at the first SubmodelElement.
	
	`[ "RotationSpeed", "RotationSpeed.MaxRotationSpeed" ]`
	
	If applied to the SubmodelElementCollection:
	
	`[ "RotationSpeed", "RotationSpeed.MaxRotationSpeed" ]`
	
	If applied to the Property inside the SubmodelElementCollection:
	
	`[ "RotationSpeed.MaxRotationSpeed" ]`
		If applied to the Submodel:
		
		Note: The SubmodelElementCollection “RotationSpeed” is the only direct child of the Submodel, therefore, it’s the only entry.
		
		`[ "RotationSpeed" ]`
		
		If applied to the SubmodelElementCollection:
		
		`[ "RotationSpeed", "RotationSpeed.MaxRotationSpeed" ]`
		
		If applied to the Property inside the SubmodelElementCollection:
		
		`[ "RotationSpeed.MaxRotationSpeed" ]`

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 RotationSpeed:


```
{
    "modelType":  "SubmodelElementCollection",
    "idShort": "RotationSpeed",
    "semanticId": {
      "keys": [ {
          "type": "GlobalReference",
          "value": "http://purl.org/iot/vocab/iot-taxonomy-
```

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:
	
	`{ "RotationSpeed": { "MaxRotationSpeed": 5000 } }`
	
	If applied to the SubmodelElementCollection, i.e. idShortPath “RotationSpeed”:
	
	`{ "MaxRotationSpeed": 5000 }`
	
	If applied to the Property, i.e. idShortPath “RotationSpeed.MaxRotationSpeed”:
	
	`5000`

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.

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

```
<symbol> ::= __expression__
```

where:

- <symbol> is a nonterminal (variable) and the *expression* 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 "|", indicating a choice, the whole being a possible substitution for the symbol on the left,
- symbols that never appear on a left side are terminals, while symbols that appear on a left side are non-terminals 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>], or suffixed with an additional (questionmark) symbol, ?, such as <optional-symbol>?,
- 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 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
```

Class Table Templates

General

The templates used for element specification are explained in this annex. For details for the semantics see Legend for UML Modelling.

For capitalization of titles, rules according to https://capitalizemytitle.com/ are used.

Template for Classes

Template 16. Class

Class:	*<Class Name> ["<<abstract>>"] ["<<Experimental>>"] ["<<Deprecated>>"] ["<<Template>>"]*
Explanation:	<Explanatory text>
Inherits from:	{<Class Name> ";" }+ | "-"
ID:	<metamodel element ID>

Attribute	ID		
	Explanation	Type	Card.
<attribute or association name> ["<<ordered>>"] ["<<Experimental>>"] ["<<Deprecated>>"]	<metamodel element ID>		
	<Explanatory text>	<Type>	<Card>

ID is the metamodel ID of the class or attribute, conformant to the grammar defined in [1]. A metamodel ID for a class attribute is concatenated by *<ID of metamodel element ID of class>/<relative metamodel element ID>*.

The following stereotypes can be used:

- <<abstract>>: Class cannot be instantiated but serves as superclass for inheriting classes
- <<Experimental>>: Class is experimental, i.e. usage is only recommended for experimental purposes because non-backward compatible changes may occur in future versions
- <<Deprecated>>: Class is deprecated, i.e. it is recommended to not use the element any longer; it will be removed in a next major version of the model
- <<Template>>: Class is a template only, i.e. class is not instantiated but used for additional specification purposes (for details see parts 3 of document series)

The following kinds of *Types* are distinguished:

- *<Class>:* Type is an object type (class); it is realized as composite aggregation (composition), and does not exist independent of its parent
- *ModelReference<{Referable}>:* Type is a Reference with *Reference/type=ModelReference* and is called model reference; the {Referable} is to be substituted by any referable element (including *Referable* itself for the most generic case) – the element that is referred to is denoted in the *Key/type*=<{Referable}> for the last *Key* in the model reference; for the graphical representation see Figure in Legend for UML Modelling; for more information on referencing see [1].
- *<Primitive>: Type* Type is no object type (class) but a data type; it is just a value, see [1].
- *<Enumeration>:* Type is an enumeration

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.

Note 1: 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.

Note 2: attributes or attribute elements with data type “string” or “langString” are considered to consist of at least one character.

Note 3: optional lists, i.e. attributes with cardinality > 1 and minimum 0, are considered to consist of at least one element.

Examples for valid and invalid model references

If class type equal to "ModelReference<Submodel>", the following reference would be a valid reference (using the text

serialization as defined in [1]):

```
(Submodel)\https://example.com/aas/1/1/1234859590
```

This would be an invalid reference for "ModelReference<Submodel>" because it references a submodel element "Property":

```
(Submodel)https://example.com/aas/1/1/1234859590, (Property)temperature
```

If class type equal to "ModelReference<Referable>", the following references would be valid references (using the text serialization as defined in [1]) because "Property" and "File" are Referables and "Submodel" itself is also Referable since all Identifiables are referable:

```
(Submodel)\https://example.com/aas/1/1/1234859590

(Submodel)\https://example.com/aas/1/1/1234859590, (Property)temperature

(Submodel)\https://example.com/aas/1/1/1234859590, (File)myDocument
```

This would be an invalid reference for "ModelReference<Referable>" because FragmentReference is no Referable:

```
(Submodel)\https://example.com/aas/1/1/1234859590, (File)myDocument
(FragmentReference)Hints
```

Template for Enumerations

Template 17. Enumeration

Enumeration:	<Enumeration Name> ["<<Experimental>>"] ["<<Deprecated>>"]
Explanation:	<Explanatory text>
Set of:	{<Enumeration> ";" }+ | "-"
ID:	<metamodel element ID>
Literal	**Explanation**
enumValue1>["<<Experimental>>"] ["<<Deprecated>>"]	<metamodel value ID>
	<Explanatory text>
<enumValue2> ["<<Experimental>>"] ["<<Deprecated>>"]	<metamodel value ID>
	<Explanatory text>

"**Set of:**" lists enumerations that are contained in the enumeration. It is only relevant for validation, making sure that all elements relevant for the enumeration are considered.

"**Literal**" lists values of enumeration. All values that are element of one of the enumeration listed in "**Set of:**" are listed

explicitly as well.

Enumeration values use Camel Case notation and start with a small letter. However, there might be exceptions in case of very well-known enumeration values.

Template for Primitives

Template 18. Primitives

Primitive	**ID**	
	Definition	**Value Examples**
<Name of Primitive>	<metamodel ID of Primitive>	
	<Explanatory text>	<Value examples>

API Table Templates

Templates for Specification of APIs and API Operations

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.

Template 19. 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.

Template 20. 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 Generic Status Codes. If no ***payload*** is mentioned as output parameter, the status code shall be SuccessNoContent in case of success, otherwise see [1]. 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 eparameter N	The cardinality of type of the outputParameter N, e.g. zero-to-one (“0..1”) or at-least-one (“1..*”).

Handling Constraints

Constraints are prefixed with **AASa-** followed by a three-digit number. The "a" in "AASa-" was motivated by "API". The numbering of constraints is unique within namespace AASa; a number of a constraint that was removed will not be used again.

Overview Constraints

This annex gives an overview of the constraints contained in this document. No additional comments are added, for details please refer to the normative parts of the specification.

For handling of constraints see Handling Constraints.

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.

Constraint AASa-002: The base64url-encoded identifier of the link: semanticId shall have a length of maximum 3072 characters.

pass:q[[underline]#Constraint AASa-003#: A service implementing the SubmodelRepositoryServiceSpecification/SSP-003 must not accept or provide any _Submodel_ with the attribute https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.1/spec-metamodel/common.html#ModellingKind.

Constraint AASa-004: A service implementing the SubmodelRepositoryServiceSpecification/SSP-004 must not accept or provide any _Submodel with the attribute https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.1/spec-metamodel/common.html#ModellingKind.

Constraint AASa-005: Only the _SubmodelElements_ root delaration can be followed with IdShortPaths.

UML

OMG UML General

This annex explains the UML elements used in this specification. For more information, please refer to the comprehensive literature available for UML. The formal specification can be found in [10].

Figure 13 shows a class with name "Class1" and an attribute with name "attr" of type *Class2*. Attributes are owned by the class. Some of these attributes may represent the end of binary associations, see also Figure 21. In this case, the instance of *Class2* is navigable via the instance of the owning class *Class1*.[3] This convention is now deprecated. Aggregation type, navigability, and end ownership are separate concepts, each with their own explicit notation. Association ends owned by classes are always navigable, while those owned by associations may be navigable or not. [10]

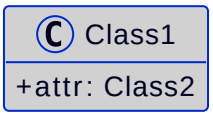


Figure 13. Class

Figure 14 shows that *Class4* inherits all member elements from *Class3*. Or in other word, *Class3* is a generalization of *Class4, Class4* is a specialization of *Class3*. This means that each instance of *Class4* is also an instance of *Class3*. An instance of *Class4* has the attributes *attr1* and *attr2,* whereas instances of *Class3* only have the attribute *attr1*.

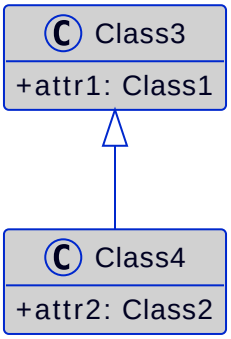


Figure 14. Inheritance/Generalization

Figure 15 defines the required and allowed multiplicity/cardinality within an association between instances of *Class1* and *Class2*. In this example, an instance of *Class2* is always related to exactly one instance of *Class1*. An instance of *Class1* is either related to none, one, or more (unlimited, i.e. no constraint on the upper bound) instances of *Class2*. The relationship can change over time.

Multiplicity constraints can also be added to attributes and aggregations.

The notation of multiplicity is as follows:

<lower-bound>.. <upper-bound>

where <lower-bound> is a value specification of type Integer - i.e. 0, 1, 2, … - and <upper-bound> is a value specification of type UnlimitedNatural. The star character (*) is used to denote an unlimited upper bound.

The default is 1 for lower-bound and upper-bound.

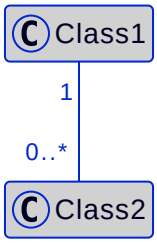


Figure 15. Multiplicity

A multiplicity element represents a collection of values. The default is a set, i.e. it is not ordered and the elements within the collection are unique and contain no duplicates. Figure 16 shows an ordered collection: the instances of *Class2* related to an instance of *Class1*. The stereotype <<ordered>> is used to denote that the relationship is ordered.

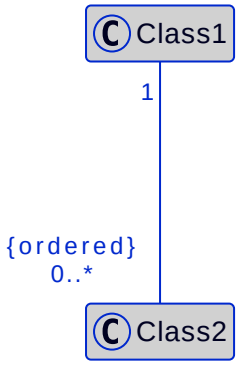


Figure 16. Ordered Multiplicity

Figure 17 shows that the member ends of an association can be named as well, i.e. an instance of *Class1* can be in relationship "relation" to an instance of *Class2*. Vice versa, the instance of *Class2* is in relationship "reverseRelation" to the instance of *Class1*.

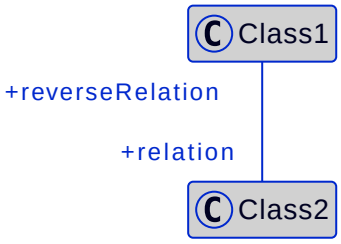


Figure 17. Association

Figure 18 depicts two classes connected by a unidirectional association from *Class1* to *Class2*. In this association,

only the endpoint is navigable, meaning it is possible to navigate from an instance of *Class1* to an instance of *Class2*, but not the other way around. An instance of *Class1* can be in a 'relation' with an instance of *Class2*, and the association is labeled 'Reference'.

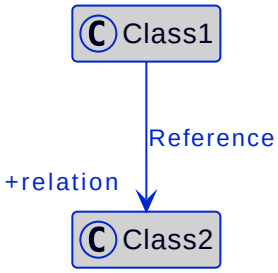


Figure 18. Association

A specialty in Figure 18 is that the label 'Reference' indicates the relationship between *Class1* and *Class2* is of a *Reference* type. This means that an instance of Class1 holds a reference to an instance of *Class2*. Furthermore, the instance of *Class2* is considered 'referable' according to the Asset Administration Shell metamodel, implying that it inherits from the predefined abstract class 'Referable' in the AAS framework. The structure of a reference to a model element of the Asset Administration Shell is explicitly defined.

Figure 19 shows a composition, also called a composite aggregation. A composition is a binary association, grouping a set of instances. The individuals in the set are typed as specified by *Class2*. The multiplicity of instances of *Class2* to *Class1* is always 1 (i.e. upper-bound and lower-bound have value "1"). One instance of *Class2* belongs to exactly one instance of *Class1*. There is no instance of *Class2* without a relationship to an instance of *Class1*. Figure 19 shows the composition using an association relationship with a filled diamond as composition adornment.

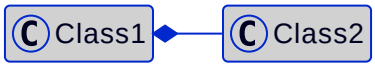


Figure 19. Composition (Composite Aggregation)

Figure 20 shows an aggregation. An aggregation is a binary association. In contrast to a composition, an instance of *Class2* can be shared by several instances of *Class1*. Figure 20 shows the shared aggregation using an association relationship with a hallow diamond as aggregation adornment.

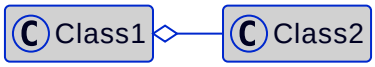


Figure 20. Aggregation

Figure 21 illustrates that the attribute notation can be used for an association end owned by a class. In this example, the attribute name is "attr" and the elements of this attribute are typed with *Class2.* The multiplicity, here "0..*", is added in square brackets. If the aggregation is ordered, it is added in curly brackets like in this example.

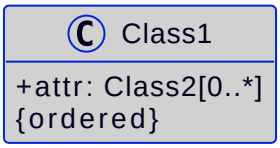


Figure 21. Navigable Attribute Notation for Associations

Figure 22 shows a class with three attributes with primitive types and default values. When a property with a default value is instantiated in the absence of a specific setting for the property, the default value is evaluated to provide the initial values of the property.

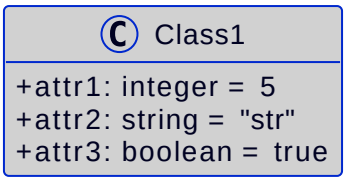


Figure 22. Default Value

Figure 23 shows that there is a dependency relationship between *Class1* and *Class2*. In this case, the dependency means that *Class1* depends on *Class2* because the type of attribute *attr* depends on the specification of class *Class2.* A dependency is depicted as dashed arrow between two model elements.

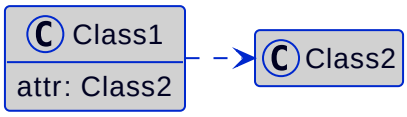


Figure 23. Dependency

Figure 24 shows an abstract class. It uses the stereotype <<abstract>>. There are no instances of abstract classes. They are typically used for specific member elements that are inherited by non-abstract classes.

Figure 24. Abstract Class

Figure 25 shows a package named "Package2". A package is a namespace for its members. In this example, the member belonging to *Package2* is class *Class2*.

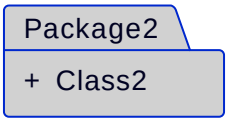


Figure 25. Package

Figure 26 shows that all elements in *Package2* are imported into the namespace defined by *Package1*. This is a special dependency relationship between the two packages with stereotype <<import>>.

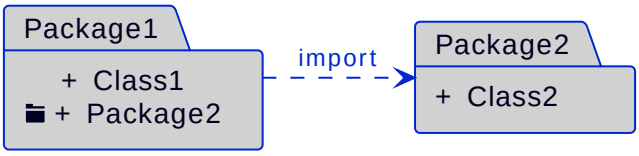


Figure 26. Imported Package

Figure 27 shows an enumeration with the name "Enumeration1". An enumeration is a data type with its values enumerated as literals. It contains two literal values, "a" and "b". It is a class with stereotype <<enumeration>>. The literals owned by the enumeration are ordered.

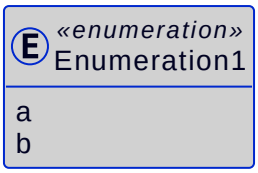


Figure 27. Enumeration

Figure 28 shows how a note can be attached to an element, in this example to class "Class1".

Figure 28. Note

Figure 29 shows how a constraint is attached to an element, in this example to class "Class1".

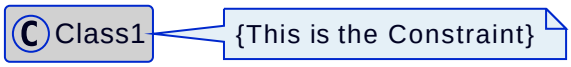


Figure 29. Constraint

UML Naming Rules

The following rules are used for naming of classes, attributes etc.:

- all names use CamelCase; for exceptions see rules for Enumeration values,
- class names always start with a capital letter,
- attribute names always start with a small letter,
- primitive types start with a capital letter; exception: predefined types of XSD like string,
- enumerations start with a capital letter,
- names of member ends of an association start with a capital letter,
- all stereotypes specific to the Asset Administration Shell specification start with a capital letter, e.g. "<<Deprecated>>"; predefined stereotypes in UML start with a small letter, e.g. "<<abstract>>" or "<<enumeration>>".

In UML, the convention is to name associations and aggregations in singular form. The multiplicity is to be taken into account to decide on whether there are none, a single, or several elements in the corresponding association or aggregation.

> Note: a plural form of the name of attributes with cardinality >=1 may be needed in some serializations (e.g. in JSON). In this case, it is recommended to add an "s". In case of resulting incorrect English (e.g. isCaseOf isCaseOfs), it must be decided whether to support such exceptions.

Templates, Inheritance, Qualifiers, and Categories

At first glance, there seems to be some overlapping within the concepts of data specification templates, extensions, inheritance, qualifiers, and categories introduced in the metamodel. This clause explains the commonalities and differences and gives hints for good practices.

In general, an extension of the metamodel by inheritance is foreseen. Templates might also be used as alternatives.

- Extensions can be used to add proprietary and/or temporary information to an element. Extensions do not support

interoperability. They can be used as work-around for missing properties in the standard. In this case, the same extensions are attached to all elements of a specific class (e.g. to properties). However, in general, extensions can be attached in a quite arbitrary way. Properties are defined in a predefined way as key values pairs (in this case keys named "name").

- In contrast to extensions, templates aim at enabling interoperability between the partners that agree on the template. A template defines a set of attributes, each of them with clear semantics. This set of attributes corresponds to a (sub-)schema. Templates should only be used if different instances of the class follow different schemas and the templates for the schemas are not known at design time of the metamodel. Templates might also be used if the overall metamodel is not yet stable enough or a tool supports templates but not (yet) the complete metamodel. Typically, all instances of a specific class with the same category provide the same attribute values conformant to the template. In contrast to extensions, the attributes in the template have speaking names.

Note: categories are deprecated and should no longer be used.

- However, when using non-standardized proprietary data specification templates, interoperability cannot be ensured and thus should be avoided whenever possible.
- In case all instances of a class follow the same schema, inheritance and/or categories should be used.
- Categories can be used if all instances of a class follow the same schema but have different constraints depending on their category. Such a constraint might specify that an optional attribute is mandatory for this category (like the unit that is mandatory for properties representing physical values). Realizing the same via inheritance would lead to multiple inheritance - a state that is to be avoided[4].

Note: categories are deprecated and should no longer be used.

- Qualifiers are used if the structure and the semantics of the element is the same independent of its qualifiers. Only the quality or the meaning of the value for the element differs.
- Value qualifiers are used if only the quantity but not the semantics of the value changes. Depending on the application, either both value and qualifier define the "real" semantics together, or the qualifier is not really relevant and is ignored by the application. Example: the actual temperature might be good enough for non-critical visualization of trends, independent of whether the temperature is measured or just estimated (qualifier would denote: measured or estimated).
- Concept qualifiers are used to avoid multiplying existing semantically clearly defined concepts with the corresponding qualifier information, e.g. life cycle.
- Template qualifiers are used to guide the creation and validation of element instances.

Notes to Graphical UML Representation

Specific graphical modelling rules, which are used in this specification but not included in this form, are explained below [10].

Figure 30 shows different graphical representations of a composition (composite aggregation). In Variant A, a relationship with a filled aggregation diamond is used. In Variant B, an attribute with the same semantics is defined. And in Variant C, the implicitly assumed default name of the attribute in Variant A is explicitly stated. This document uses notation B.

It is assumed that only the end member of the association is navigable per default, i.e. it is possible to navigate from an instance of *Class1* to the owned instance of *Class2* but not vice versa. If there is no name for the end member of the association given, it is assumed that the name is identical to the class name but starting with a small letter - compared to Variant C.

Class2 instance only exists if the parent object of type *Class1* exists.

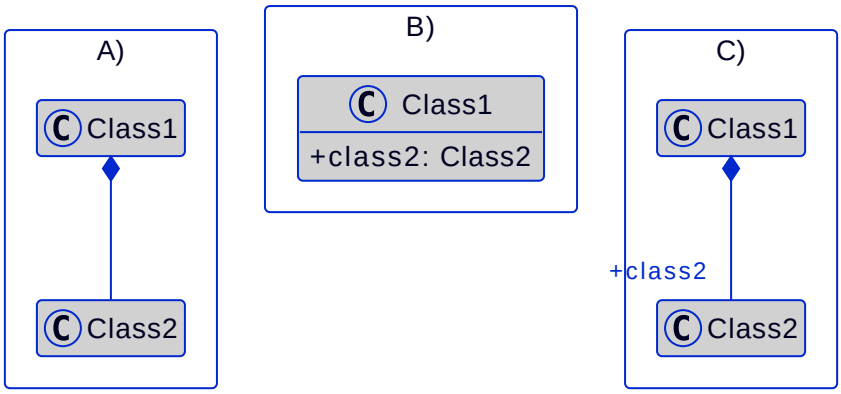


Figure 30. Graphical Representations of Composite Aggregation/Composition

Figure 31 shows a representation of a shared aggregation: a *Class2* instance can exist independently of a *Class1* instance. It is assumed that only the end member of the aggregation association is navigable per default, i.e. it is possible to navigate from an instance of *Class1* to the owned instance of *Class2* but not vice versa.

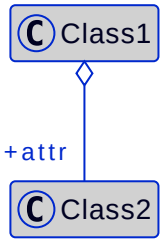


Figure 31. Graphical Representation of Shared Aggregation

Figure 32 show different graphical representations of generalization. Variant A is the classical graphical representation as defined in [10]. Variant B is a short form. The name of the class that *Class3* is inheriting from is depicted in the upper right corner.

Variant C not only shows which class *Class3* instances are inheriting from, but also what they are inheriting. This is depicted by the class name it is inheriting from, followed by "::" and then the list of all inherited elements - here attribute *class2*. Typically, the inherited elements are not shown.

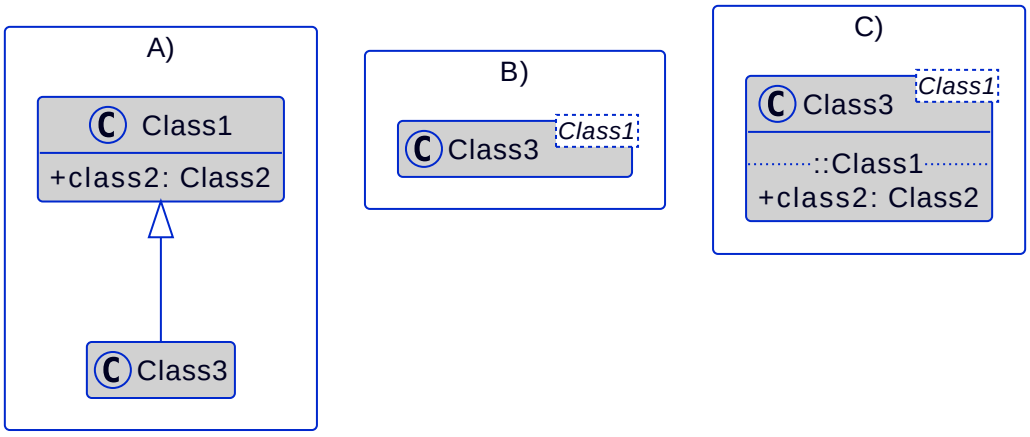


Figure 32. Graphical Representation of Generalization/Inheritance

Figure 33 depicts different graphical notations for enumerations in combination with inheritance. On the left side "Enumeration1" additionally contains the literals as defined by "Enumeration2".

Note 1: the direction of inheritance is opposite to the one for class inheritance. This can be seen on the right side of Figure 33 that defines the same enumeration but without inheritance.

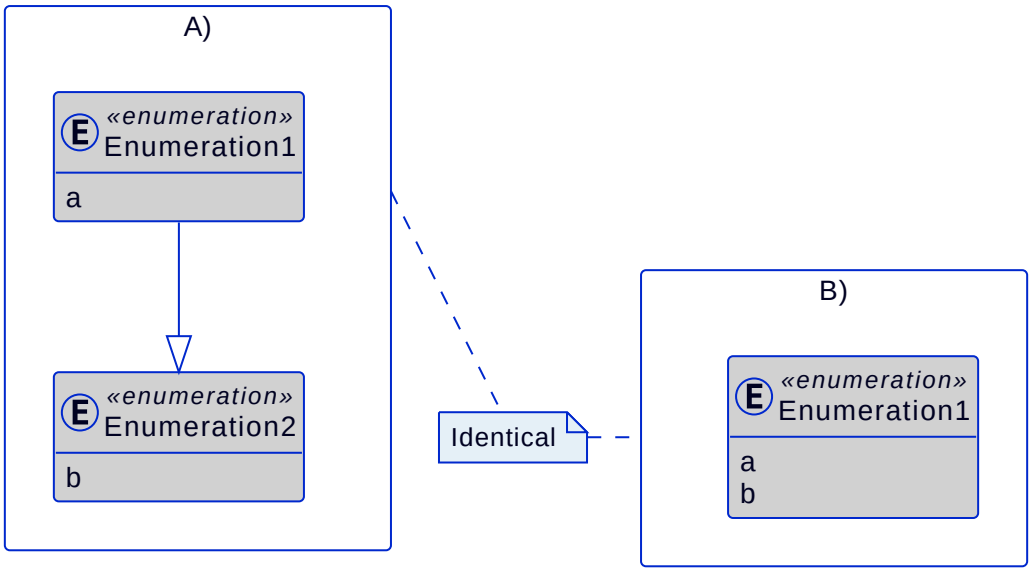


Figure 33. Graphical Representation for Enumeration with Inheritance

Note 2: in this specification all elements of an enumeration are ordered alphabetically.

Figure 34 shows an experimental class, marked by the stereotype "Experimental".

Figure 34. Graphical Representation for Experimental Classes

Figure 35 depicts a deprecated class, which is marked by the stereotype "Deprecated".

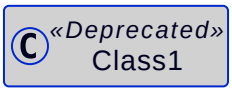


Figure 35. Graphical Representation for Deprecated Elements

Figure 36 shows a class representing a template. It is marked by the stereotype "Template".

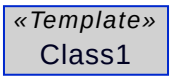


Figure 36. Graphical Representation of a Template Class

[1] Navigability notation was often used in the past according to an informal convention, whereby non-navigable ends were assumed to be owned by the Association whereas navigable ends were assumed to be owned by the Classifier at the opposite end.

[2] Exception: multiple inheritance is used in this specification, but only in case of inheriting from abstract classes.

[3] Navigability notation was often used in the past according to an informal convention, whereby non-navigable ends were assumed to be owned by the Association whereas navigable ends were assumed to be owned by the Classifier at the opposite end.

[4] Exception: multiple inheritance is used in this specification, but only in case of inheriting from abstract classes.

Change Log

General

- * Means not backward compatible
- (*) means not backward compatible but just renaming

> Note: Changes in Metamodel (IDTA-01001) will not be listed here, although they have an impact on the payload of many operations.

Changes w.r.t. V3.0.4 to V3.1

Major Changes:

- new: Bulk Operations and Profiles for Bulk Operations (#3)
- new: Query Operations and Profiles for executing complex queries
- deprecated: operation GetAllAssetAdministrationShellIdsByAssetLink
- new: operation SearchAllAssetAdministrationShellIdsByAssetLink (as substitute for GetAllAssetAdministrationShellIdsByAssetLink) (#19)
- new: Profile for Discovery Service: Read Only SSP-002 (#201)
- new: Profile for the Asset Administration Shell Registry: Minimal Read SSP-005 (#201)
- update: Terms and Definitions adopted to IEC 63278-1:2023 (before IEC 63278-1 Draft July 2022 was the basis), also abbreviations partly adopted; changed definitions (#210):
  - changed: interface
  - changed: service
- update: Individual versioning of datatype IDs, operations, interfaces etc. (#247)
  - including changes due to implicit changes of IDTA-01001, e.g. of primitive data types NameType and Identifier
- change: Data type Identifier: change length from 2000 to 2048 characters (#306)
- update: Publication of a new V3.1 version for all existing profiles due to changes to fundamental data types, e.g., Identifier
- update: interaction diagram in Clause "Interactions"
- Transfer of chapters on formats Metadata, Paths and Value-Only from Part 2 API to Part 1 Metamodel (#214)
- Transfer from .docx to asciidoc (.adoc) and maintenance in GitHub
- Transfer of all UML figures to PlantUML (.puml) and maintenance in GitHub
- OpenAPI: marked parameter 'level' as deprecated for all URLs ending with '/$reference'
- change: ProtocolInformation/securityAttributes changed from mandatory to optional.
- change: Endpoint/interface extended with "VALUE" and "METADATA" aspects.
- change: data type for ProtocolInformation/endpointProtocolVersion and subprotocolBodyEncoding changed from LabelType to NameType
- new: Added 'Role' and 'NotApplicable' wherever AssetAdministrationShellDescriptor/AssetKind is used.
- change: Added minLength=1 requirement and regex pattern "^[a-zA-Z][a-zA-Z0-9_-]*[a-zA-Z0-9_]+$" to all classes with idShort parameters to keep definitions in sync with the declaration of Referable/idShort
- change: Fix missing 'Repository' in ConceptDescriptionServiceSpecification to ConceptDescriptionRepositoryServiceSpecification (#171)
- new: Add classes OperationRequestValueOnly and OperationResultValueOnly explicitly (#269)
- change: Additional clarifications on the implicit support of major and minor versions of service specifications (264)

- change: PUT can also create the considered resource, not only replace it, if a creation via POST was possible already.

Minor Changes:

- clarification on how to handle duplicate query parameters
- removal of Clause "Security" because Part 4 Security now covers security aspects
- replace "servers" clause in OpenAPI files
- add notes for base64url-encoded values and order of "assetIds" query parameters
- update Bibliography
- add overview of constraints to Annex
- Design Decisions for the HTTP API: Clarify that padding is not allowed for base64url-encoded values (#423)
- HTTP Modifier Constraints: adding note for metadata and value-only representations of Asset Administration Shells (#268)
- Interactions: extending the explanation how to discover endpoints of AAS services
- Mapping of Operations: adding missing path for the row defining the mapping of PutAssetAdministrationShell to HTTP
- Fixing several "operationId" and "x-semanticIds" values in OpenAPI files.

Interface Changes w.r.t. V3.0.4 to V3.1

BWC	Interface	Kind of Change	Comment
	AAS Registry, Submodel Registry, AAS Repository, Submodel Repository, Concept Description Repository	Extended	Newly introduced API operations for querying
	AAS Basic Discovery	Changed	deprecate GetAllAssetAdministrationShellIdsByAssetLink add SearchAllAssetAdministrationShellIdsByAssetLink changed GetAllAssetAdministrationShellDescriptors: Changed the parameters assetKind and assetType from mandatory to optional. OpenAPI remains unchanged. (#298) Changed the parameter assetIds from mandatory to optional. OpenAPI remains unchanged. (#301)

BWC	Interface	Kind of Change	Comment
	AAS Registry	Extended	add CreateBulkAssetAdministrationShellDescriptors add PutBulkAssetAdministrationShellDescriptorsById add DeleteBulkAssetAdministrationShellDescriptorsById
	Submodel Registry	Extended	add PostBulkSubmodelDescriptors add PutBulkSubmodelDescriptorsById add DeleteBulkSubmodelDescriptorsById
	GenerateSerializationByIds	Extended	Extended description for the "includeConceptDescriptions" parameter
	AAS Repository, AAS Registry, AAS Service, Submodel Repository, Submodel Registry, Submodel Service, AASX File Server Service, Concept Description Repository	Extended	Allowed PUT operations to also create objects if a creation via POST was available.

Operation Changes w.r.t. V3.0.4 to V3.1

Operation	Kind of Change	Comment
QueryAssetAdministrationShells	new	new query API-Operation for AAS Repository interface
QuerySubmodels	new	new query API-Operation for Submodel Repository interface
QueryAssetAdministrationShellDescriptors	new	new query API-Operation for AAS Registry interface
QuerySubmodelDescriptors	new	new query API-Operation for Submodel Registry interface

Operation	Kind of Change	Comment
QueryConceptDescriptions	new	new query API-Operation for Concept Description Repository interface
GetAllAssetAdministrationShellIdsByAssetLink	deprecated	substituted by SearchAllAssetAdministrationShellIdsByAssetLink
SearchAllAssetAdministrationShellIdsByAssetLink	new	substitute for GetAllAssetAdministrationShellIdsByAssetLink
CreateBulkAssetAdministrationShellDescriptors	new	new API-Operation for AAS Registry Interface
PutBulkAssetAdministrationShellDescriptorsById	new	new API-Operation for AAS Registry Interface
DeleteBulkAssetAdministrationShellDescriptorsById	new	new API-Operation for AAS Registry Interface
PostBulkSubmodelDescriptors	new	new API-Operation for Submodel Registry Interface
PutBulkSubmodelDescriptorsById	new	new API-Operation for Submodel Registry Interface
DeleteBulkSubmodelDescriptorsById	new	new API-Operation for Submodel Registry Interface

Profile Changes w.r.t. V3.0.4 to V3.1

Profile	Kind of Change	Comment
Query Profiles	new	
Asset Administration Shell Registry Profile - Bulk Profile	new	
Submodel Registry Profile - Bulk Profile	new	
Discovery Profile - Full Profile	update	GetAllAssetAdministrationShellIdsByAssetLink set to deprecated added new API-operation SearchAllAssetAdministrationShellIdsByAssetLink
Discovery Profile - Read Profile	new	
all	new	Added version 3.1 for all existing profiles

Class Changes w.r.t. V3.0.4 to V3.1

Template 21. Changes in Data Types for Payload

Nc	V3.1 Change w.r.t. V3.0	Comment
	AssetAdministrationShellDescriptor/assetType	data type: change length from 2000 to 2048 characters
	AssetAdministrationShellDescriptor/globalAssetId	data type: change length from 2000 to 2048 characters
	AssetAdministrationShellDescriptor/id	data type: change length from 2000 to 2048 characters
	SubmodelDescriptor/id	data type: change length from 2000 to 2048 characters
	PackageDescription/aasIds	data type: change length from 2000 to 2048 characters
	ProtocolInformation/href	data type: change length from 2000 to 2048 characters
	ProtocolInformation/endpointProtocolVersion	data type: change from LabelType to NameType (i.e.g change of length from 64 to 128 characters)
	ProtocolInformation/subprotocolBodyEncoding	data type: change from LabelType to NameType (i.e.g change of length from 64 to 128 characters)
	ProtocolInformation/securityAttributes	Changed securityAttributes from mandatory to optional. OpenAPI remains unchanged. (#384)

Template 22. New Data Types for Payload

	New Elements V3.1 vs V3.0	Comment
	AssetLink	new class for discovery operation(s)
	AssetLink/name	
	AssetLink/value	

Changes w.r.t. V3.0.3 to V3.0.4

Major: * Change: paging_metadata is required in OpenAPI to match the definition in this document. * Change: ValueOnly classes changed from array to object for AnnotatedRelationshipElementValue/value, EntityValue/statements * Change: Return type of the OpenAPI classes GetSubmodelsValueResult and GetSubmodelElementsValueResult from array to object (#251)

Interface Changes w.r.t. V3.0.3 to V3.0.4

None.

Operation Changes w.r.t. V3.0.3 to V3.0.4

Operation Change Old	Operation Change New	Kind of Change	Comment
PostAllAssetLinksById had cardinality of the payload of “1”	payload cardinality is “1..*”	Change	

Changes w.r.t. V3.0.2 to V3.0.3

Major:

- Clause 12.2: Reintroducing the design decision for the ReferenceParent class due to resolvement problems reappearing in SwaggerHub, also adding the class again to the OpenAPI file.
- Adding the missing GetSubmodelElementsMetadataResult class to the OpenAPI definitions for the AAS API classes.

Interface Changes w.r.t. V3.0.2 to V3.0.3

BWC	Interface Change	Kind of Change	Comment
	GetAllAssetAdministrationShellDescriptors	Changed	Changed the parameters assetKind and assetType from 'mandatory' to 'optional'. OpenAPI remains unchanged. (#298)
	AAS Basic Discovery Interface	Changed	Changed the parameter assetIds from mandatory to optional. OpenAPI remains unchanged. (#301)

Operation Changes w.r.t. V3.0.2 to V3.0.3

None.

Changes w.r.t. V3.0.1 to V3.0.2

Major:

- Remove "format: byte" from OpenAPI files, as this annotation enforces base64 encodings while base64url is actually required. Remove the QueryParameter "level" from all requests ending with /$metadata in the OpenAPI files.
- Change the values for the ServiceDescription class from enum to a list of strings in the OpenAPI definition for the Part 2 classes.
- Clause 11.4.2 and 11.4.3: Change the ValueOnly attribute "annotation" to "annotations" and its value from an array to ValueOnly for the AnnotatedRelationshipElementValue class in the examples and schema to match the "AnnotatedRelationshipElement/annotations" attribute. Furthermore, "AnnotatedRelationshipElementValue/annotations" is optional now.
- Clause 11.4.2: Add serialisation rule for empty "FileValue/value" and "BlobValue/value".
- Clause 11.4.3: "FileValue/value" and "BlobValue/value" are optional but non-empty attributes.
- Clause 11.4.3: "EntityValue/statements" made optional.
- Clause 11.4.3: Fix maxLength for "FileValue/value" from 200 to 2000 characters.
- Clause 12.2: Remove the design decision for the ReferenceParent class, also removing it from the OpenAPI file.
- Clause 12.3: Relax the requirements for API paths and version declaration.
- (Editorial) Clause 12.8: Added constraint on the return object for ValueOnly requests and that the Level modifier is undefined for Metadata requests.
- Adopt the V3.0.1 bugfix changes of the AAS Metamodel in the OpenAPI files and references.

Minor:

- (Editorial) Fix links to SwaggerHub in Clause 4.6 and Clause 12
- (Editorial) Clause 8.2.2: Correct the Note that explains the usage of "globalAssetId" for the "assetIds" parameter.
- (Editorial) ServiceSpecificationProfileEnum: Corrected explanations for Submodel Repository and Registry profile entries
- (Editorial) Clause 11.4.2: Add "Submodel" to the list of possible ValueOnly objects and add more details how

SubmodelElementLists have to be serialized.

- (Editorial) Clause 11.4.4: Added sentence explaining where the idShortPath has to start.
- (Editorial) Clause 12.2: Correct example for GetAllAssetAdministrationShellIdsByAssetLink
- (Editorial) Correct Note 1 in Clause 12.4: "[…] IdshortPaths are base64url-encoded …" to "[…] IdshortPaths are url-encoded …"
- (Editorial) Clause 12.13: Corrected the list of constraints.
- (Editorial) Annex C.2: Adjusted and extended the examples for GETs on Metadata, Path, and Value
- (Editorial) Annex C.3: Adjusted the examples for PATCH on Value

Interface Changes w.r.t. V3.0.2 to V3.1

BWC	Interface Change	Kind of Change	Comment
	GetOperationAsnycStatus	Changed	Replace payload type ‘OperationResult’ with ‘BaseOperationResult’ that OpenAPI descriptions already contained ‘BaseOperationResult’ from V3.0 on.
	ServiceSpecificationProfileEnum	Remove	Removed profiles https://admin-shell.io/aas/API/3/0/RepositoryServiceSpecification/SSP-001 and https://admin-shell.io/aas/API/3/0/RepositoryServiceSpecification/SSP-002. Both profiles were not included in the V3.0.1 ServiceDescription class in the OpenAPI definition and only left-overs from previous drafts.
	PutAssetAdministrationShell	Change	Output parameter "payload" changed from "mandatory=yes" to "mandatory=no"
	PutSubmodel	Change	Output parameter "payload" changed from "mandatory=yes" to "mandatory=no"
	PatchSubmodel	Change	Output parameter "payload" changed from "mandatory=yes" to "mandatory=no"
	PutSubmodelElementByPath	Change	Output parameter "payload" changed from "mandatory=yes" to "mandatory=no"
	PatchSubmodelElementByPath	Change	Output parameter "payload" changed from "mandatory=yes" to "mandatory=no"
	PutAssetAdministrationShellDescriptorById	Change	Output parameter "payload" changed from "mandatory=yes" to "mandatory=no"
	PutSubmodelDescriptorById	Change	Output parameter "payload" changed from "mandatory=yes" to "mandatory=no"
	PutAssetAdministrationShellById	Change	Output parameter "payload" changed from "mandatory=yes" to "mandatory=no"
	PutSubmodelById	Change	Output parameter "payload" changed from "mandatory=yes" to "mandatory=no"

BWC	Interface Change	Kind of Change	Comment
	PatchSubmodelById	Change	Output parameter "payload" changed from "mandatory=yes" to "mandatory=no"
	PutConceptDescriptionById	Change	Output parameter "payload" changed from "mandatory=yes" to "mandatory=no"

Operation Changes w.r.t. V3.0.1 to V3.0.2

Operation Change Old	Operation Change New	Kind of Change	Comment
GetSubmodelElementByPath-Metadata had ‘level’ and/or ‘cursor’ parameters in OpenAPI	‘level’ and/or ‘cursor’ parameters have been removed GetSubmodelElementByPath-Metadata	Remove	Neither ‘level’ nor ‘cursor’ could influence the behavior of the operation, therefore, removing them has no effect on any implementation.
CreateSubmodelReference	New comment in Table 15 clarifies the content of the Location header of the response.	Change	
GetSelfDescription output type "ServiceDescription" contains enum for "profiles"	GetSelfDescription output type "ServiceDescription" contains list of strings for "profiles"	Change	Custom profiles can be added with the new structure.
String attributes of Part 2 classes are restricted to the regex pattern "^[\\x09\\x0A\\x0D\\x20-\\uD7FF\\uE000-\\uFFFD\\U00010000-\\U0010FFFF]*$"	String attributes of Part 2 classes are restricted to the regex pattern "^([\\t\\n\\r -\ud7ff\ue000-\ufffd]|\ud800[\\udc00-\\udfff]|[\\ud801-\\udbfe][\\udc00-\\udfff]|\\udbff[\\udc00-\\udfff])*$"	Change	The new pattern has been introduced for the JSON schema and the Part 1 OpenAPI domain already. This change synchronizes the pattern for both Part 1 and Part 2 classes.
GetSubmodelElementByPath had single PathItem as the response object in some OpenAPI files	GetSubmodelElementByPath returns an array of PathItems	Change	
PathItem regex did not allow SubmodelElementLists	PathItem regex does allow SubmodelElementLists	Change	

Changes w.r.t. V3.0 to V3.0.1

Major:

- Added Location header for POSTs that create a new resource, according to RFC 9110 Section 15.3.2
- Correcting definitions of SerialisationModifiers and Pagination parameters in the OpenAPI files.

- Cleaning of incorrectly located API Operations from the OpenAPI files.
- Removing several outdated/inconsistently named OpenAPI files from the GitHub Release.
- Fixing the values of the ServiceDescription/profiles enum in the OpenAPI Domain.

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

Interface Changes w.r.t. V1.0RC03 to V3.0

BWC	Interface Change	Kind of Change	Comment
	Submodel	New	PatchSubmodel and PatchSubmodelElementByPath (PUT to completely replace and PATCH to update content)

BWC	Interface Change	Kind of Change	Comment
	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 GetAllConceptDescriptionsByDataSpecification Reference 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

BWC	Interface Change	Kind of Change	Comment
	*	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.

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

Changes w.r.t. V1.0RC02 to V1.0RC03

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

BWC	Interface Change	Kind of Change	Comment
*	Discovery	Changed	IdentifierKeyValuePair 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

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
GetAllAssetAdministrationShells ByAssetLink		Changed	IdentifierKeyValuePair to SpecificAssetId
GetAllAssetLinksById		Changed	IdentifierKeyValuePair to SpecificAssetId
PostAllAssetLinksById		Changed	IdentifierKeyValuePair to SpecificAssetId

Changes w.r.t. V1.0RC01 to V1.0RC02

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

BW C	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

BWC	Interface Change	Kind of Change	Comment
	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 PutAssetAdministrationShellById 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: GetAllConceptDescriptionsWithDataSpecificationReference to GetAllConceptDescriptionsByDataSpecificationReference, PutConceptDescription to PutConceptDescriptionById New: PostConceptDescription

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

Operation Change Old	Operation Change New	Kind of Change	Comment
	PostAssetAdministrationDescriptor	New	
PutSubmodelDescriptor	PutSubmodelDescriptorById	Changed	Naming pattern byId
	PostSubmodelDescriptor	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 pattern By
PutConceptDescription	PutConceptDescriptionById	Changed	Naming pattern byId
	PostConceptDescription	New	

Bibliography

Bibliography

[1] IDTA-01001-3-1. Specification of the Asset Administration Shell. Part 1: Metamodel. Industrial Digital Twin Association (IDTA). Online. Available: https://industrialdigitaltwin.org/en/content-hub/aasspecifications

[2] IDTA-01003-a-3-1. Specification of the Asset Administration Shell. Part 3a: Data Specification – IEC 61360. Industrial Digital Twin Association (IDTA). Online. Available: https://industrialdigitaltwin.org/en/content-hub/aasspecifications

[3] IDTA-01004-3-1. Specification of the Asset Administration Shell. Part 4: Security. Industrial Digital Twin Association (IDTA). Online. Available: https://industrialdigitaltwin.org/en/content-hub/aasspecifications

[4] IDTA-01005-3-1. Specification of the Asset Administration Shell. Part 5: Package File Format. Industrial Digital Twin Association (IDTA). Online. Available: https://industrialdigitaltwin.org/en/content-hub/aasspecifications

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

[6] DIN EN IEC 61406-1: "Identification Link - Part 1: General requirements (IEC 61406-1:2022)". December 2023. Online. Available: https://www.dinmedia.de/en/standard/din-en-iec-61406-1/372053652

[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

[9] Tom Preston-Werner. Semantic Versioning. Version 2.0.0. Online. Available: https://semver.org/spec/v2.0.0.html

[10] "OMG Unified Modelling Language (OMG UML)". Formal/2017-12-05. Version 2.5.1. December 2018. Online. Available: https://www.omg.org/spec/UML/

[11] "RQL: A resource query language for REST". Ariel Mashraki Version 1.3.0. March 2021. Online. Available: https://github.com/a8m/rql

[12] "SPARQL 1.1 Query Language". W3C Recommendation. Edited by Steve Harris and Andy Seaborne March 2013. Online. Available: https://www.w3.org/TR/sparql11-query/

[13] ISO/IEC 39075:2024. Information technology — Database languages — GQL. ISO/IEC. 2024. Online. Available: https://www.iso.org/standard/76120.html

[14] "JSONPath - XPath for JSON". Stefan Gössner 2007. Online. Available: http://goessner.net/articles/JsonPath/

www.industrialdigitaltwin.org