



Specification of the Asset Administration Shell

Part 2: Application Programming Interfaces

SPECIFICATION

IDTA Number: 01002
Version 3.2

Specification of the Asset Administration Shell

This specification is part of the [Asset Administration Shell Specification series](#).

Version

This is version 3.2.0 of the specification IDTA-01002.

Previous version: 3.1.3

Alignment with other AAS Specifications

This version of IDTA-01002 (v3.2) is designed to be used together with the following companion specifications:

Dependency	Specification	Aligned Version	Notes
x	IDTA-01001 Part 1: Metamodel	v3.2	API: - defines payload of many API operations like <code>GetAssetAdministrationShell</code>, <code>GetSubmodel</code> etc. (collected in <code>Part1-MetaModel-Schemas</code>) - is the basis for derived data types for payload like <code>AssetLink</code>, <code>SubmodelDescriptor</code> etc. used in Registry or Discovery API operations - defines mapping rules for serializations like <code>Value</code>, <code>Metadata</code> etc. Query Language: - defines the classes referenced by <code>\$aas</code>, <code>\$sm</code>, <code>\$sme</code>, <code>\$cd</code> - defines the attributes used in conditions like <code>semanticId</code>, <code>specificAssetIds</code> etc.
x	IDTA-01003-a Part 3a: Data Specification Template IEC61360	v3.1	is integrated into the schema of Part 1
x	IDTA-01003-b Part 3b: Data Specification Template Measurement Units	v3.0	is integrated into the schema of Part 1

Dependency	Specification	Aligned Version	Notes
x	IDTA-01004 Part 4: Security	v3.1	API: • security measurements shall be applied to API • access control rules shall be implemented (Note: if an implementation only supports a subset of access control rules this shall be specified) Query Language: • IDTA-01002 and IDTA-01004 share the same BNF grammar and JSON Schema for formula expressions; see Query Language and the Access Rule Model in IDTA-01004.
x	IDTA-01005 Part 5: Package File Format (AASX)	v3.2	defines return object for API operations like GetAASXByPackageld

Bugfix (patch) releases of aligned specifications shall be substituted.

Notice

Copyright: Industrial Digital Twin Association e.V. (IDTA)

IDTA Number: IDTA-01002

Version: 3.2.0

DOI: <https://doi.org/10.62628/IDTA.01002-3-2>

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).

SPDX-License-Identifier: CC-BY-4.0

July 2026

How to Get in Contact

Contact: [IDTA](#)

Sources: [GitHub](#)

Feedback:

- [New issues, bugs](#)
- [Questions and Answers](#)

Editorial Notes

History

This document (version 3.2) was produced by the Work Stream "Specification of the Asset Administration Shell" of the Working Group "Open Technology" of the [Industrial Digital Twin Association \(IDTA\)](#).

Version 3.1 was finalized in May 2025 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 [\[9\]](#) with the following deviation: Bugfix versions are not necessarily backward compatible.

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
JWS	JSON Web Signature
MIME	Multipurpose Internet Mail Extensions
OPC	Open Packaging Conventions (ECMA-376, ISO/IEC 29500-2)

Abbreviation	Description
OPCF	OPC Foundation
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 specified 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](#)).

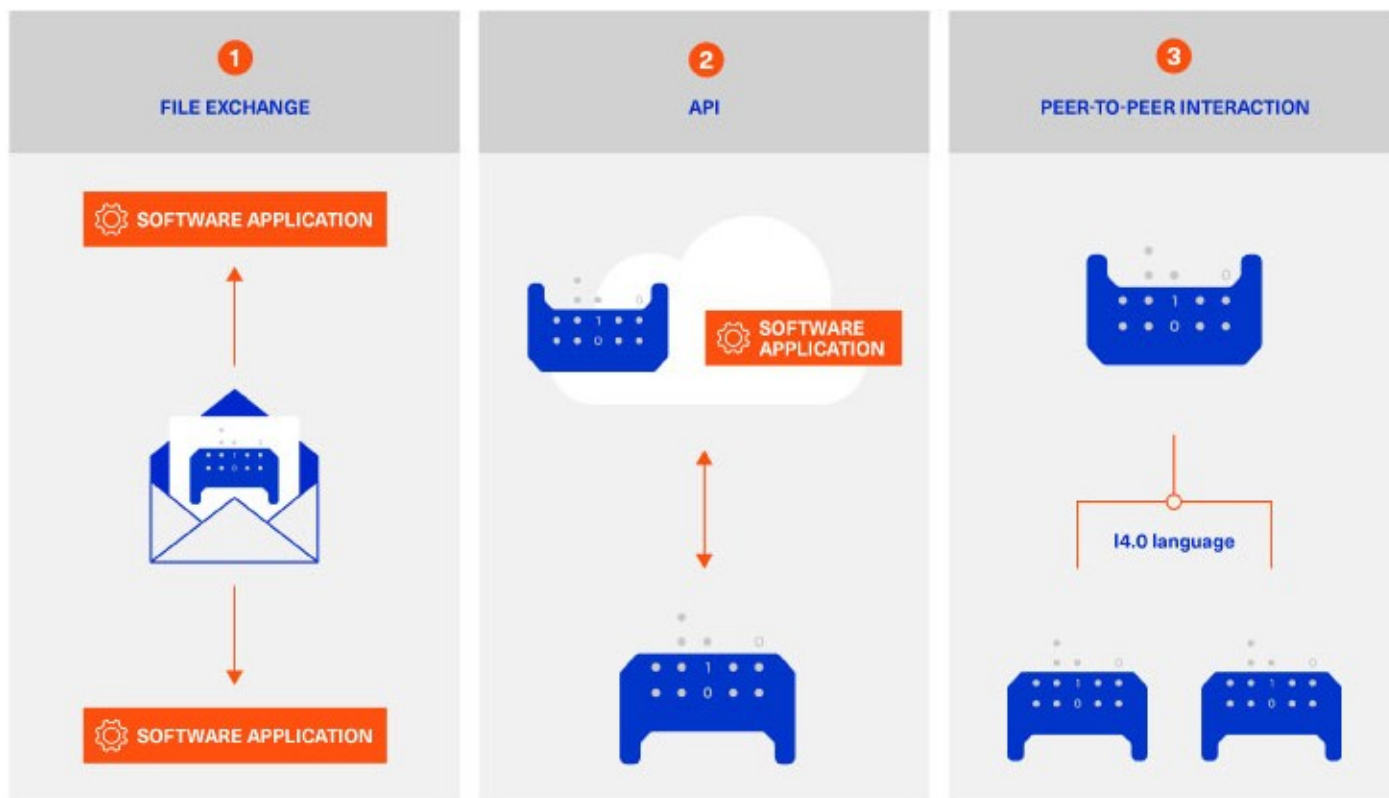


Figure 1. Types of Information Exchange via Asset Administration Shells

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.

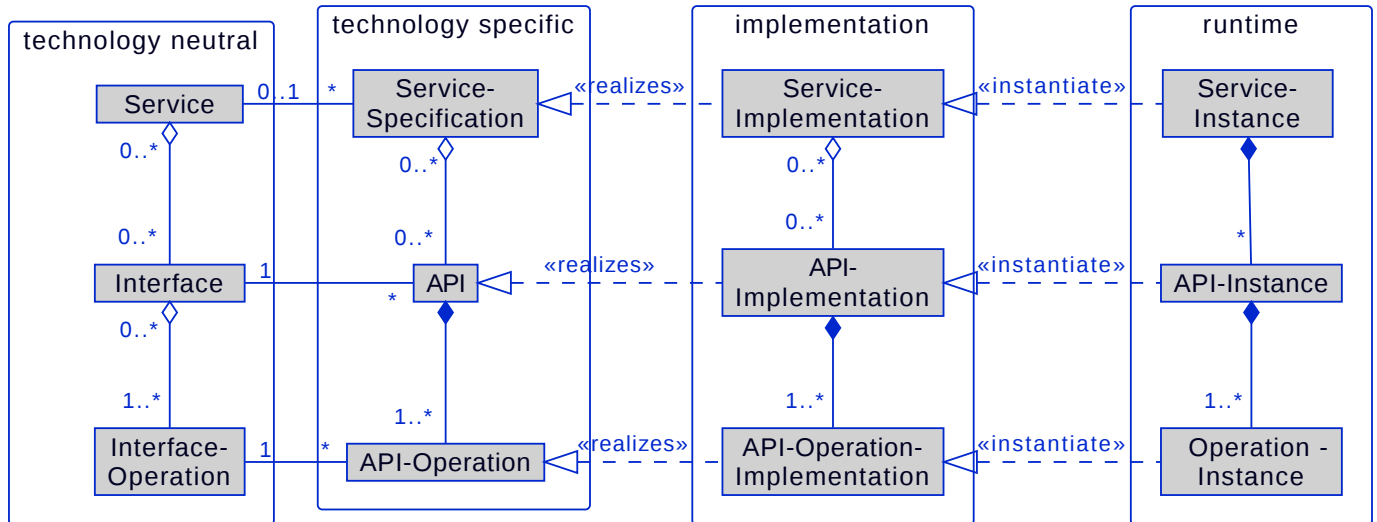


Figure 2. Services, Interfaces & APIs and Operations

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: asynchronous invocation of an operation.
- SEARCHALL: returns a list of 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-
Qualifier>][<RecentChanges>]

<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/3/2>

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.

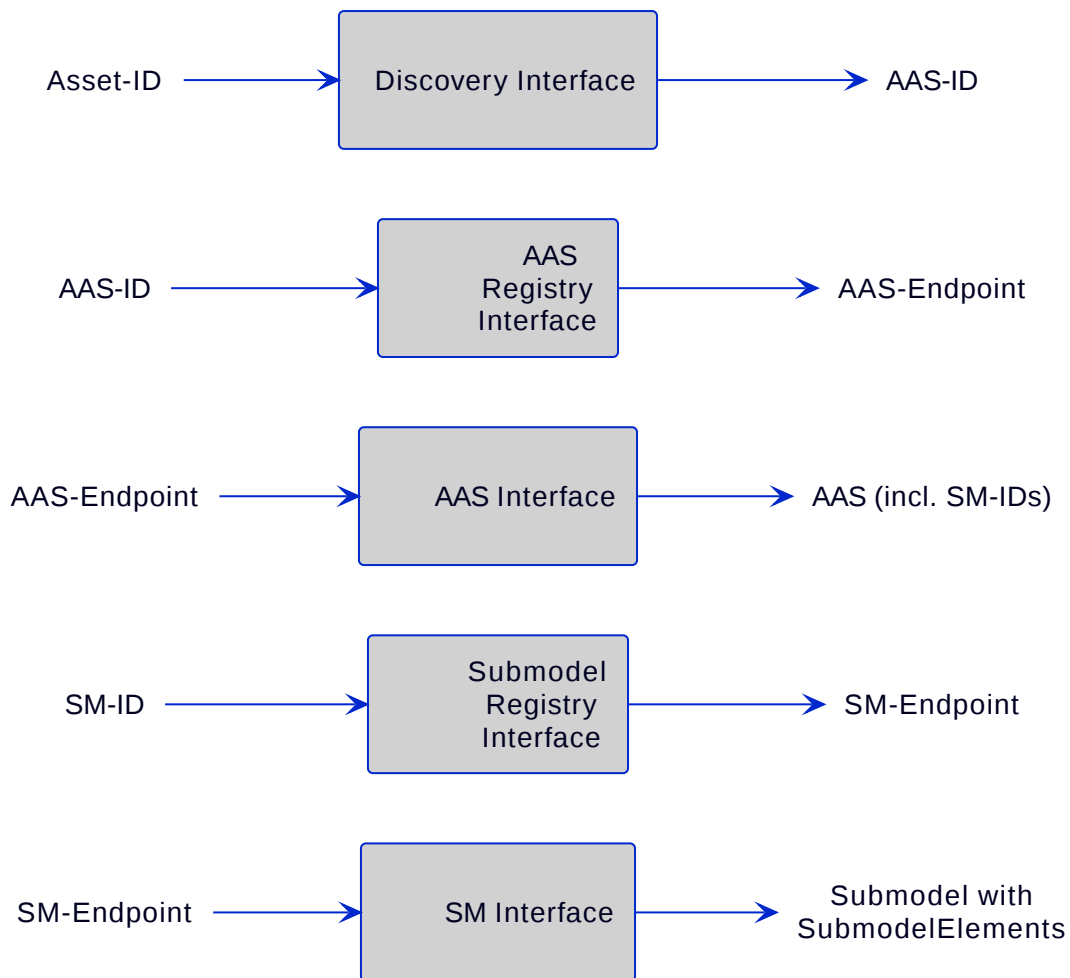


Figure 3. Retrieval of Asset-related Information by AAS and Submodels

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://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AasxFileServerServiceSpecification/V3.2 SSP-001.yaml&version=v3.2.0>. 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.

Single source of truth for the formula grammar and JSON Schema

The BNF grammar and JSON Schema productions for **formula expressions** (logical expressions, comparisons, FieldIdentifiers, value literals, operands, type casts, and \$match / \$and / \$or / \$not operators) are **normatively defined in this specification** (IDTA-01002, this chapter and the JSON Schema below).

IDTA-01004 Part 4: Security [\[3\]](#) uses the same formula grammar and schema for **FORMULA**, **FILTER** and related constructs in Access Rules. IDTA-01004 MUST NOT redefine or diverge from these shared productions; it only adds Access-Rule-specific productions (ACL, OBJECTS, RIGHTS, ACCESS, ROUTE, etc.) on top.

If a future bugfix or minor release changes a shared production in IDTA-01002, IDTA-01004 inherits that change. Implementations SHOULD validate formulas against the schema shipped with their IDTA-01002 version.

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").

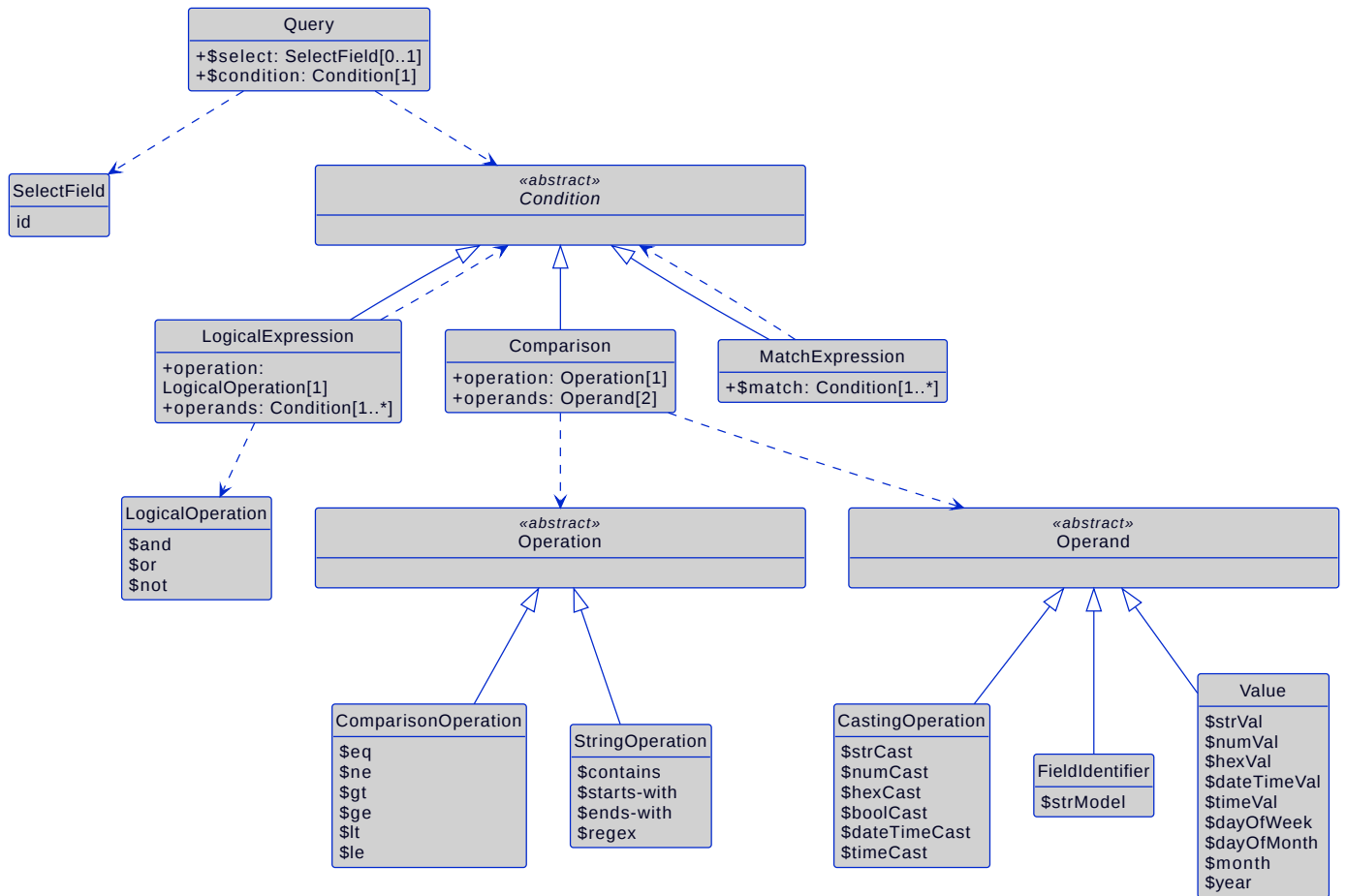


Figure 4. Main elements of the AAS Query Language

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: Queries 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 rejection 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, supplementalSemanticIds, idShort, value, valueType, and language 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> ( <ws> "$filters" <ws> <QueryFilter>* )?
<selectStatement> ::= "$select" <ws> "id" <ws>
<QueryFilter> ::=
    ( "$fragment" <ws> <FieldIdentifierFragment> <ws> )
    ( "$condition" <ws> <logicalExpression> <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> <FieldIdentifier> <ws> ) |
( <FieldIdentifier> <ws> <allComparisons> <ws> <stringOperand> <ws> )

<numericalComparison> ::=
  ( <numericalOperand> <ws> <allComparisons> <ws> <numericalOperand> <ws> ) |
  ( <numericalOperand> <ws> <allComparisons> <ws> <FieldIdentifier> <ws> ) |
  ( <FieldIdentifier> <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> ::=
  <FieldIdentifier> | <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> <logicalExpression> <ws> )*
    ( <AccessPermissionRule> <ws> )*

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

<SecurityQueryFilter> ::=
    ( "FRAGMENT:" <ws> <FieldIdentifierFragment> <ws> )
    ( ( "CONDITION:" <ws> <logicalExpression> <ws> ) | ( <UseFormula> <ws> ) )

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

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

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

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

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

```

```

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

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

<ReferenceAttribute> ::=
    "REFERENCE(" (
        ("aas" <IdentifierInstance> "#" <FieldsAAS> ) |
        ("sm" <IdentifierInstance> "#" <FieldsSM> ) |
        ("cd" <IdentifierInstance> "#" <FieldsCD> ) |
        ("sme" <IdentifierInstance> "." <idShortPath> "#" <FieldsSME> )
    ) ")"

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

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

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

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

<IdentifierInstance> ::= "(" <IdentifierStringLiteral> ")"
<IdentifierInstanceOrAll> ::= <IdentifierInstance> | "("*"")"

<IdentifiableObject> ::=
    "IDENTIFIABLE" <ws> ("aas" | "sm" | "cd") <IdentifierInstanceOrAll>

<ReferableObject> ::=
    "REFERABLE" <ws> "sme" <IdentifierInstanceOrAll> "." <idShortPath>

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

<DescriptorObject> ::=
    "DESCRIPTOR" <ws> ("aasdesc" | "smdesc") <IdentifierInstanceOrAll>

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

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

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

```

```

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

<digit> ::= [0-9]
<ArrayIndex> ::= "0" | ( [1-9] [0-9]* )
<StringLiteral> ::= "\"" <StringCharacter>+ "\""
<StringCharacter> ::= <UnescapedStringCharacter> | <EscapedStringCharacter>
<UnescapedStringCharacter> ::= ? any Unicode character except quotation mark and reverse
solidus ?
<EscapedStringCharacter> ::= "\\" ( "\"" | "\\" )
<IdentifierStringLiteral> ::= "\"" <IdentifierCharacter>+ "\""
<IdentifierCharacter> ::= <UnescapedIdentifierCharacter> | <EscapedStringCharacter>
<UnescapedIdentifierCharacter> ::= ? any character permitted by the AAS String data type
except quotation mark and reverse solidus ?
<ClaimLiteral> ::= <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> ::= <FieldIdentifierAAS> | <FieldIdentifierSM> | <FieldIdentifierSME> |
<FieldIdentifierCD> | <FieldIdentifierAasDescriptor> | <FieldIdentifierSmDescriptor>

<FieldIdentifierAAS> ::= "$aas#" <FieldsAAS>
<FieldIdentifierSM> ::= "$sm#" <FieldsSM>
<FieldIdentifierSME> ::= "$sme" ( "." <idShortPath> )? "#" <FieldsSME>
<FieldIdentifierCD> ::= "$cd#" <FieldsCD>
<FieldIdentifierAasDescriptor> ::= "$aasdesc#" <FieldsAasDescriptor>
<FieldIdentifierSmDescriptor> ::= "$smdesc#" <SmDescriptorClause>

<FieldsAAS> ::= "idShort" | "id" | "assetInformation.assetKind" |
"assetInformation.assetType" | "assetInformation.globalAssetId" | "assetInformation."
<SpecificAssetIdsClause> | "submodels" ( "[" <ArrayIndex>? "]" ) ( "." <ReferenceClause> )?
<FieldsSM> ::= <SemanticIdClause> | <SupplementalSemanticIdClause> | "idShort" | "id"
<FieldsSME> ::= <SemanticIdClause> | <SupplementalSemanticIdClause> | "idShort" | "value" |

```

```

"valueType" | "language"
<FieldsCD> ::= "idShort" | "id"
<FieldsAasDescriptor> ::= "idShort" | "id" | "assetKind" | "assetType" | "globalAssetId" |
<SpecificAssetIdsClause> | "endpoints" ( "[" <ArrayIndex>? "]" ) "." <EndpointClause> |
"submodelDescriptors" ( "[" <ArrayIndex>? "]" ) "." <SmDescriptorClause>
<SmDescriptorClause> ::= ( <SemanticIdClause> | <SupplementalSemanticIdClause> | "idShort"
| "id" | "endpoints" ( "[" <ArrayIndex>? "]" ) ) "." <EndpointClause> )
<EndpointClause> ::= "interface" | "protocolInformation.href"

<ReferenceClause> ::= ( "type" | "keys" ( "[" <ArrayIndex>? "]" ) ( ".type" | ".value" ) )
<SemanticIdClause> ::= ( "semanticId" | "semanticId." <ReferenceClause> )
<SupplementalSemanticIdClause> ::= ( "supplementalSemanticIds" ( "[" <ArrayIndex>? "]" ) )? (
"." <ReferenceClause> )? )
<SpecificAssetIdsClause> ::= ( "specificAssetIds" ( "[" <ArrayIndex>? "]" ) ( ".name" |
"value" | ".externalSubjectId" | ".externalSubjectId." <ReferenceClause> ) )

<FieldIdentifierFragment> ::= <FieldIdentifierAASFragment> | <FieldIdentifierSMFragment> |
<FieldIdentifierSMEFragment> | <FieldIdentifierCDFragment> |
<FieldIdentifierAasDescriptorFragment> | <FieldIdentifierSmDescriptorFragment>

<FieldIdentifierAASFragment> ::= "$aas#" ( "idShort" | "assetInformation.assetType" |
"assetInformation.globalAssetId" | "assetInformation." <SpecificAssetIdsClauseFragment> |
<SubmodelsClauseFragment> )
<FieldIdentifierSMFragment> ::= "$sm#" ( <SemanticIdClauseFragment> |
<SupplementalSemanticIdClauseFragment> | "idShort" )
<FieldIdentifierSMEFragment> ::= "$sme" ( "." <idShortPath> )? ( "#" (
<SemanticIdClauseFragment> | <SupplementalSemanticIdClauseFragment> | "idShort" | "value" |
"valueType" | "language" ))?
<FieldIdentifierCDFragment> ::= "$cd#" ( "idShort" )
<FieldIdentifierAasDescriptorFragment> ::= "$aasdesc#" ( "idShort" | "description" |
"displayName" | "extension" | "administration" | "assetKind" | "assetType" |
"globalAssetId" | <SpecificAssetIdsClauseFragment> | <EndpointClauseFragment> |
<SubmodelDescriptorsClauseFragment> )
<FieldIdentifierSmDescriptorFragment> ::= "$smdesc#" <SmDescriptorClauseFragment>

<SpecificAssetIdsClauseFragment> ::= "specificAssetIds" | "specificAssetIds" ( "["
<ArrayIndex>? "]" ) ( ".externalSubjectId" ( "." <ReferenceClauseFragment> )? )?
<SubmodelsClauseFragment> ::= "submodels" | "submodels" ( "[" <ArrayIndex>? "]" ) ( "."
<ReferenceClauseFragment> )?
<SubmodelDescriptorsClauseFragment> ::= "submodelDescriptors" | "submodelDescriptors" ( "["
<ArrayIndex>? "]" ) ( "." <SmDescriptorClauseFragment> )?
<SmDescriptorClauseFragment> ::= ( <SemanticIdClauseFragment> |
<SupplementalSemanticIdClauseFragment> | "idShort" | <EndpointClauseFragment> )
<EndpointClauseFragment> ::= "endpoints" ( "[" <ArrayIndex>? "]" )?

<SemanticIdClauseFragment> ::= "semanticId" | "semanticId." <ReferenceClauseFragment>
<SupplementalSemanticIdClauseFragment> ::= "supplementalSemanticIds" ( "[" <ArrayIndex>?
"]" ( "." <ReferenceClauseFragment> )? )?
<ReferenceClauseFragment> ::= "keys" ( "[" <ArrayIndex>? "]" )?

```

```

<idShortPath> ::= ( <idShort> ( "[" <ArrayIndex>? "]" ) * ( "." <idShortPath> ) * )
<idShort> ::= ( ( [a-z] | [A-Z] ) ( [a-z] | [A-Z] | [0-9] | "_" | "-" ) * ( [a-z] | [A-Z] | [0-9] | "_" ) )
<ws> ::= ( " " | "\t" | "\r" | "\n" ) *

```

For AAS Query Language requests, only the <query> production SHALL be accepted as the request payload. The <AllAccessPermissionRules>, <AccessPermissionRule>, FILTER and FILTERLIST productions are included only for the shared grammar with IDTA-01004 and SHALL NOT be accepted as AAS Query request payloads unless explicitly specified by a security profile.

Within a <StringLiteral>, a quotation mark and a reverse solidus SHALL be escaped as \" and \\, respectively. All other Unicode characters may be used without escaping. The <IdentifierStringLiteral> used in identifier instance selectors is additionally limited to 1–2048 characters from the AAS String character range. This allows selectors to represent the complete AAS Identifier value space without making their closing quotation mark ambiguous.

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 declaration 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



The prefixes \$aas, \$sm, \$sme, \$cd, \$aasdesc and \$smdesc correspond directly to the AAS Metamodel classes AssetAdministrationShell, Submodel, SubmodelElement, ConceptDescription, AssetAdministrationShellDescriptor and SubmodelDescriptor, and to the API resources /shells, /submodels, .../submodel-elements/{idShortPath}, /concept-descriptions, /shell-descriptors and /submodel-descriptors respectively. The normative cross-specification mapping is given in the "Terminology Across IDTA-01001, IDTA-01002 and IDTA-01004" section of IDTA-01001 (AAS Metamodel). All prefixes are lower-case; variants such as \$aasDesc or \$SMDesc MUST be rejected by conforming parsers.

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 Concept Description	\$aas#id
idShort	Value of the idShort attribute	\$aas#idShort
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.specificAssetIds[<index>].name	Name of a SpecificAssetId of an AAS	\$aas#assetInformation.specificAssetIds[0].name
assetInformation.specificAssetIds[<index>].value	Value of a SpecificAssetId of an AAS	\$aas#assetInformation.specificAssetIds[1].value
assetInformation.specificAssetIds[<index>].externalSubjectId.type	Type of a Reference used as an externalSubjectId in a SpecificAssetId of an AAS	\$aas#assetInformation.specificAssetIds[0].externalSubjectId.type
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.specificAssetIds[0].externalSubjectId.keys[0].value
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.specificAssetIds[0].externalSubjectId.keys[0].type

submodels[<index>]	Shortcut for Submodels referenced by an AAS, see Clause References	\$aas#submodels[]
submodels[<index>].type	Type of a Reference that associates an AAS with a Submodel	\$aas#submodels[].type
submodels[<index>].keys[<index>].value	Value of a key used in a Reference that associates an AAS with a Submodel	\$aas#submodels[].keys[].value
submodels[<index>].keys[<index>].type	Type of a key used in a Reference that associates an AAS with a Submodel	\$aas#submodels[0].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
supplementalSemanticIds	Shortcut for supplementalSemanticIds, see Clause References	\$sm#supplementalSemanticIds
supplementalSemanticIds[<index>].type	ReferenceType of a supplementalSemanticId Reference	\$sm#supplementalSemanticIds[].type
supplementalSemanticIds[<index>].keys[<index>].type	KeyType of a supplementalSemanticId Reference	\$sm#supplementalSemanticIds[].keys[].type
supplementalSemanticIds[<index>].keys[<index>].value	Value of a key of a supplementalSemanticId Reference	\$sm#supplementalSemanticIds[].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[<index>].semanticId	Shortcut for semanticIds, see Clause References	\$aasdesc#submodelDescriptors[.semanticId
submodelDescriptors[<index>].semanticId.type	ReferenceType of a semanticId Reference used in a referenced Submodel of an Asset Administration Shell Descriptor	\$aasdesc#submodelDescriptors[.semanticId.type

submodelDescriptors[<index>].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[<index>].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[<index>].supplementalSemanticIds	Shortcut for supplementalSemanticIds, see Clause References	\$aasdesc#submodelDescriptors[].supplementalSemanticIds
submodelDescriptors[<index>].supplementalSemanticIds[<index>].type	ReferenceType of a supplementalSemanticId Reference used in a referenced Submodel of an Asset Administration Shell Descriptor	\$aasdesc#submodelDescriptors[].supplementalSemanticIds[].type
submodelDescriptors[<index>].supplementalSemanticIds[<index>].keys[<index>].type	KeyType of a supplementalSemanticId Reference used in a referenced Submodel of an Asset Administration Shell Descriptor	\$aasdesc#submodelDescriptors[].supplementalSemanticIds[].keys[].type
submodelDescriptors[<index>].supplementalSemanticIds[<index>].keys[<index>].value	Value of a key of a supplementalSemanticId Reference used in a referenced Submodel of an Asset Administration Shell Descriptor	\$aasdesc#submodelDescriptors[].supplementalSemanticIds[].keys[].value
submodelDescriptors[<index>].id	Identifier of a referenced Submodel as available in an Asset Administration Shell Descriptor	\$aasdesc#submodelDescriptors[].id
submodelDescriptors[<index>].idShort	idShort of a referenced Submodel as available in an Asset Administration Shell Descriptor	\$aasdesc#submodelDescriptors[].idShort
submodelDescriptors[<index>].endpoints[<index>].interface	Endpoint interface of a referenced Submodel as available in an Asset Administration Shell Descriptor	\$aasdesc#submodelDescriptors[].endpoints[0].interface
submodelDescriptors[<index>].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 implicitly casted to the string "17", however, this is different to the value of the first parameter "Instance".
\$aas#assetInformation.assetKind \$ne 17	true	17 is implicitly casted to the string "17", which is not equal to the value of the first parameter "Instance".
\$aas#assetInformation.assetKind \$le \$aas#assetInformation.assetKind	true	\$eq implies \$le
\$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
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-datetime
time(<value>)	Casts the value to xs:time.	Defined by https://www.w3.org/TR/xpath-functions/#casting-to-datetime

<TimeLiteral> follows the <time> production in the BNF grammar: HH:MM, HH:MM:SS, and these forms with fractional seconds are valid. Timezone-qualified xs:time lexical forms, such as 12:00:00Z or 12:00:00+01:00, are also valid <TimeLiteral> values.

<DateTimeLiteral> follows the <datetime> production in the BNF grammar: a valid four-digit calendar date is followed by a <time>, with an optional T separator and optional timezone.

Implicit 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 implicitly 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 implicit 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 implicitly 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 "smIdShort", this notation may lead to a following declaration:

```
$sme.<someIdShortPath>.smIdShort[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 "smIdShort", this notation may lead to a following declaration:

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

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 equivalent:

```
<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"
```

This shall also work for supplementalSemanticIds, e.g.:

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

Which is equivalent to:

```
$sme#supplementalSemanticIds[].keys[0].value $eq  
"https://example.com/a/supplemental/semantic/id"
```

Note: supplementalSemanticIds is a list, therefore, the comparison with the shortcut will be true if at least one of the supplementalSemanticIds has the given value, not necessarily the first one.

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 comparisons. 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
------------	--------	---------

<pre>\$match(\$aas#assetInformation.specificAssetIds[]. name \$eq "supplierId", \$aas#assetInformation.specificAssetIds[]. value \$eq "aas-1")</pre>	<pre>[{ "modelType": "AssetAdministrationShell", "id": "https://example. com/asset- administration- shell-1", ... }]</pre>	The values for 'supplierId' and 'aas-1' exist in the same SpecificAssetId object.
<pre>\$match(\$aas#assetInformation.specificAssetIds[]. name \$eq "supplierId", \$aas#assetInformation.specificAssetIds[]. value \$eq "aas-2")</pre>	<pre>[]</pre>	The values for 'supplierId' and 'aas-2' do not exist in the same SpecificAssetId object.
<pre>\$and (\$aas#assetInformation.specificAssetIds[]. name \$eq "supplierId", \$aas#assetInformation.specificAssetIds[]. value \$eq "aas-2")</pre>	<pre>[{ "modelType": "AssetAdministrationShell", "id": "https://example. com/asset- administration- shell-1", ... }]</pre>	The values for 'supplierId' and 'aas-2' exist in the example AAS, even though in <i>different</i> SpecificAssetId objects. As no \$match enforces the evaluation on the same SpecificAssetId, the AAS from the example fulfils the condition.

<pre> \$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")) </pre>	<pre> [{ "modelType": "AssetAdministrationShell", "id": "https://example. com/asset- administration- shell-1", ... }] </pre>	Both \$match conditions are fulfilled, even though for different SpecificAssetIds. Therefore the OR clause also evaluates to true for both items.
--	--	---

Query Filter

The Query Language supports a **\$filters** option that allows clients to reduce the amount of returned data. This can significantly improve response times, as the backend may scan and process fewer data elements. Furthermore, filtering is an important mechanism in the security domain, as it helps restrict access to sensitive information by preventing unnecessary exposure of internal data structures.

\$filters expressions can be applied to specific metamodel elements of the AAS (e.g., **Submodel**, **Submodel Elements** contained within a **Submodel**, etc.). However, not all elements can be filtered arbitrarily. If a particular metamodel element is mandatory according to the AAS specification, filtering it out would result in invalid or incomplete data structures. Therefore, such filtering operations are prohibited to ensure consistency and correctness of the returned data. Since a valid Reference requires at least one key, filters that eliminate all keys remain valid but result in the removal of the entire Reference rather than just its keys.

If no **\$filters** is specified for a particular metamodel element, that element remains unfiltered. Thus, applying additional filters can only reduce the amount of data returned; it can never increase the amount of data or introduce information that is not already present in the underlying AAS. If multiple filters are specified and apply to overlapping data, they are combined using logical AND semantics.

For fragments that address an array-valued object, a fragment ending with **[]** applies row-wise filtering to each array entry. The corresponding fragment without the trailing **[]** applies the condition to the entire array object. A row-wise condition evaluates the **\$fragment** and any **\$field** references in its **\$condition** in the context of the same array entry. If a fragment does not define a form ending with **[]**, row-wise filtering for that fragment SHALL NOT be assumed.

Table 3. Array-valued fragment filter examples

Grammar	JSON Schema	Comment
---------	-------------	---------

<pre> true \$filters \$fragment \$sme.someIdShort[] \$condition \$sme.someIdShort[]#value \$eq "val" </pre>	<pre> { "\$condition": { "\$boolean": true }, "\$filters": [{ "\$fragment": "\$sme.someIdShort[]", "\$condition": { "\$eq": [{ "\$field": "\$sme.someIdShort[]#value" }, { "\$strVal": "val" }] } }] } </pre>	Applies the condition row-wise to each array entry.
<pre> true \$filters \$fragment \$sme.someIdShort \$condition \$sme.someIdShort[]#value \$eq "val" </pre>	<pre> { "\$condition": { "\$boolean": true }, "\$filters": [{ "\$fragment": "\$sme.someIdShort", "\$condition": { "\$eq": [{ "\$field": "\$sme.someIdShort[]#value" }, { "\$strVal": "val" }] } }] } </pre>	Applies the condition to the entire array-valued object.

Assume the returned object contains this fragment before filtering:

```

{
  "submodelElements": [
    {
      "idShort": "someIdShort",
      "value": [
        {
          "idShort": "entry1",
          "value": "val"
        },
        {

```

```

        "idShort": "entry2",
        "value": "other"
      }
    ]
  }
]
}

```

The row-wise fragment `$sme.someIdShort[]` keeps only the array entries for which the condition is true:

```

{
  "submodelElements": [
    {
      "idShort": "someIdShort",
      "value": [
        {
          "idShort": "entry1",
          "value": "val"
        }
      ]
    }
  ]
}

```

The fragment `$sme.someIdShort` keeps or removes the entire array-valued object. Since at least one entry satisfies the condition, the whole fragment remains:

```

{
  "submodelElements": [
    {
      "idShort": "someIdShort",
      "value": [
        {
          "idShort": "entry1",
          "value": "val"
        },
        {
          "idShort": "entry2",
          "value": "other"
        }
      ]
    }
  ]
}

```

Filter example with boolean condition

Grammar	JSON Schema	Comment
---------	-------------	---------

```
$sm#semanticId $eq
  "https://admin-
shell.io/idta/nameplate/3/0/Nam
eplate"
$filters
  $fragment
$sme.AddressInformation.Zipcode
$condition false
```

```
{
  "$condition": {
    "$eq": [
      {
        "$field":
"$sm#semanticId"
      },
      {
        "$strVal":
"https://admin-
shell.io/idta/nameplate/3/0/Nam
eplate"
      }
    ],
    "$filters": [
      {
        "$fragment":
"$sme.AddressInformation.Zipcod
e",
        "$condition": {
          "$boolean": false
        }
      }
    ]
  }
}
```

Selects Nameplate submodels and filters out AddressInformation.Zipcode data.

```

$sm#semanticId $eq
  "https://admin-
shell.io/idta/nameplate/3/0/Nam
eplate"
$filters
  $fragment $sme#value
  $condition $not
($sm#semanticId.keys[].value
$eq
"https://example.com/semantic-
id/hide-value")
  $fragment
$sm#semanticId.keys[]
  $condition $not
($sm#semanticId.keys[].value
$eq
"https://example.com/semantic-
id/hide-value")

```

```

{
  "$condition": {
    "$eq": [
      {
        "$field":
"$sm#semanticId"
      },
      {
        "$strVal":
"https://admin-
shell.io/idta/nameplate/3/0/Nam
eplate"
      }
    ],
    "$filters": [
      {
        "$fragment":
"$sme#value",
        "$condition": {
          "$not": {
            "$eq": [
              {
                "$field":
"$sm#semanticId.keys[].value"
              },
              {
                "$strVal":
"https://example.com/semantic-
id/hide-value"
              }
            ]
          }
        }
      },
      {
        "$fragment":
"$sm#semanticId.keys[]",
        "$condition": {
          "$not": {
            "$eq": [
              {
                "$field":
"$sm#semanticId.keys[].value"
              },
              {
                "$strVal":
"https://example.com/semantic-
id/hide-value"
              }
            ]
          }
        }
      }
    ]
  }
}

```

Selects Nameplate submodels and filters out values from all nested submodel elements when the key <https://example.com/semantic-id/hide-value> is present in the submodel's semanticId. Additionally, any keys with this value in the submodel's semanticId are removed from the result.

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.

Query Evaluation Semantics (normative)

This clause defines how an AAS query **MUST** be evaluated so that client and server implementations behave deterministically.

Parse and Validation

- A query that does not match the BNF `grammar.bnf` or does not validate against the JSON Schema **MUST** be rejected with HTTP status 400 Bad Request.
- Combinations that parse but are semantically unsupported by the server (e.g. operator/type combinations the server does not implement) **MUST** be rejected with HTTP status 422 Unprocessable Entity.

Evaluation of FieldIdentifiers

- A FieldIdentifier that does not resolve to any value on a candidate object (because the field does not exist on that target) produces **no comparison match**: the candidate is excluded from the result set. This **MUST NOT** raise an error.
- A FieldIdentifier that is not applicable for the active profile (see [FieldIdentifier Applicability per Profile](#)) is treated like "field does not exist": the candidate is excluded, no error is raised.
- A list-valued FieldIdentifier (`submodels[]`, `specificAssetIds[]`, `supplementalSemanticIds[]`, ...) matches when **at least one** list element satisfies the comparison, unless an explicit index is provided.

Casting Errors

- An explicit cast that fails on the current candidate (e.g. `num("abc")`) **MUST** exclude the candidate from the result set; it **MUST NOT** fail the whole query.
- Implicit casts between the types listed in the [casting table](#) follow the same rule.

Combination with Access Rules

When a query is evaluated by a service that also enforces Access Rules (see IDTA-01004), the query's candidate set **MUST** be filtered through the applicable access rules before being returned. Access rules that evaluate to **not applicable** for a candidate (e.g. because a referenced FieldIdentifier does not exist in the current profile) **MUST NOT** cause the query to fail; such rules simply do not contribute an **ALLOW** for the candidate.

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, polymorphism 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.

```
{
```

```

"$schema": "http://json-schema.org/draft-07/schema#",
"title": "Common JSON Schema for AAS Queries and Access Rules",
"description": "This schema contains all classes that are shared between the AAS Query
Language and the AAS Access Rule Language.",
"oneOf": [
  {
    "$ref": "#/definitions/Query"
  },
  {
    "$ref": "#/definitions/AllAccessPermissionRules"
  }
],
"definitions": {
  "standardString": {
    "type": "string",
    "minLength": 1
  },
  "ReferenceIdentifier": {
    "type": "string",
    "pattern": "^(?:\\$aas\\(\\(?:[\\x09\\x0A\\x0D\\x20-\\x21\\x23-\\x5B\\x5D-
\\uD7FF\\uE000-\\uFFFD]|\\uD800-\\uDBFF|\\uDC00-\\uDFFF]|^\\u0000-
\\uFFFF|\\\\\\[\\\"\\\\\\\\\\]\\{1,2048\\}\"\\)\\#(?:idShort|id|assetInformation\\.assetKind|assetInforma
tion\\.assetType|assetInformation\\.globalAssetId|assetInformation\\.specificAssetIds\\[(?:
0|[1-9][0-9]*)?\\]\\.(?:name|value|externalSubjectId(?:\\.\\(?:type|keys\\[(?:0|[1-9][0-
9]*)?\\]\\.(?:type|value)))?)|submodels\\[(?:0|[1-9][0-
9]*)?\\]\\.(?:type|keys\\[(?:0|[1-9][0-
9]*)?\\]\\.(?:type|value)))?)|\\$sm\\(\\(?:[\\x09\\x0A\\x0D\\x20-\\x21\\x23-\\x5B\\x5D-
\\uD7FF\\uE000-\\uFFFD]|\\uD800-\\uDBFF|\\uDC00-\\uDFFF]|^\\u0000-
\\uFFFF|\\\\\\[\\\"\\\\\\\\\\]\\{1,2048\\}\"\\)\\#(?:semanticId(?:\\.\\(?:type|keys\\[(?:0|[1-9][0-
9]*)?\\]\\.(?:type|value)))?)|supplementalSemanticIds(?:\\.\\[(?:0|[1-9][0-
9]*)?\\])?(?:\\.\\(?:type|keys\\[(?:0|[1-9][0-
9]*)?\\]\\.(?:type|value)))?)|idShort|id)|\\$cd\\(\\(?:[\\x09\\x0A\\x0D\\x20-\\x21\\x23-
\\x5B\\x5D-\\uD7FF\\uE000-\\uFFFD]|\\uD800-\\uDBFF|\\uDC00-\\uDFFF]|^\\u0000-
\\uFFFF|\\\\\\[\\\"\\\\\\\\\\]\\{1,2048\\}\"\\)\\#(?:idShort|id)|\\$sme\\(\\(?:[\\x09\\x0A\\x0D\\x20-
\\x21\\x23-\\x5B\\x5D-\\uD7FF\\uE000-\\uFFFD]|\\uD800-\\uDBFF|\\uDC00-\\uDFFF]|^\\u0000-
\\uFFFF|\\\\\\[\\\"\\\\\\\\\\]\\{1,2048\\}\"\\)\\.[A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-9_])?(?:\\.\\[(?:0|[1-
9][0-9]*)?\\])?(?:\\.\\.[A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-9_])?(?:\\.\\[(?:0|[1-9][0-
9]*)?\\])?)*\\#(?:semanticId(?:\\.\\(?:type|keys\\[(?:0|[1-9][0-
9]*)?\\]\\.(?:type|value)))?)|supplementalSemanticIds(?:\\.\\[(?:0|[1-9][0-
9]*)?\\])?(?:\\.\\(?:type|keys\\[(?:0|[1-9][0-
9]*)?\\]\\.(?:type|value)))?)|idShort|value|valueType|language))$"
  },
  "IdentifiableIdentifier": {
    "type": "string",
    "pattern": "^\\$\\(?:aas|sm|cd\\)\\(\\(?:[\\x09\\x0A\\x0D\\x20-\\x21\\x23-\\x5B\\x5D-
\\uD7FF\\uE000-\\uFFFD]|\\uD800-\\uDBFF|\\uDC00-\\uDFFF]|^\\u0000-
\\uFFFF|\\\\\\[\\\"\\\\\\\\\\]\\{1,2048\\}\"\\)\\$"
  },
  "ReferableIdentifier": {
    "type": "string",
    "pattern": "^\\$sme\\(\\(?:[\\x09\\x0A\\x0D\\x20-\\x21\\x23-\\x5B\\x5D-
\\uD7FF\\uE000-\\uFFFD]|\\uD800-\\uDBFF|\\uDC00-\\uDFFF]|^\\u0000-

```

```

\\uFFFF|\\\\\\\\\\"\\\\\\\\){,2048}\\\"\\\\\\\\.[A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-9_])?(?:\\\"(?:[1-9][0-9]*)?\\\")*(?:\\. [A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-9_])?(?:\\\"(?:[1-9][0-9]*)?\\\")*)*$\"
    },
    \"FragmentIdentifier\": {
        \"$ref\": \"#/definitions/FragmentFieldIdentifier\"
    },
    \"DescriptorIdentifier\": {
        \"type\": \"string\",
        \"pattern\": \"^\\\\$$(?:aasdesc|smdesc)\\\\\\\\(\\\\\"(?:[\\\\x09\\\\x0A\\\\x0D\\\\x20-\\\\x21\\\\x23-\\\\x5B\\\\x5D-\\\\uD7FF\\\\uE000-\\\\uFFFF]|\\\\uD800-\\\\uDBFF]|\\\\uDC00-\\\\uDFFF]|^(\\\\u0000-\\\\uFFFF)|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\){1,2048}\\\\\\\\\\\\\\\\)$\"
    },
    \"FieldIdentifier\": {
        \"type\": \"string\",
        \"pattern\":
\"^((?:\\\\$aas#(?:idShort|id|assetInformation\\\\.assetKind|assetInformation\\\\.assetType|assetIn
formation\\\\.globalAssetId|assetInformation\\\\.specificAssetIds\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:name|value|externalSubjectId(?:\\\\. (?:type|keys\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:type|value)))?)|submodels\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\(?:\\\\. (?:type|keys\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:type|value))))|\\\\$sm#(?:semanticId(?:\\\\. (?:type|keys\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:type|value)))?|supplementalSemanticIds(?:\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\)?(?:\\\\. (?:type|keys\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:type|value))))|idShort|id)|\\\\$sme(?:\\\\. [A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-
9_])?(?:\\\"(?:[1-9][0-9]*)?\\\")*(?:\\\\. [A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-
9_])?(?:\\\"(?:[1-9][0-9]*)?\\\")*)*)?#(?:semanticId(?:\\\\. (?:type|keys\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:type|value)))?|supplementalSemanticIds(?:\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\)?(?:\\\\. (?:type|keys\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:type|value))))|idShort|value|valueType|language)|\\\\$cd#(?:idShort|id)|\\\\$aasd
esc#(?:idShort|id|assetKind|assetType|globalAssetId|specificAssetIds\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:name|value|externalSubjectId(?:\\\\. (?:type|keys\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:type|value)))?)|endpoints\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:interface|protocolinformation\\\\.href)|submodelDescriptors\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:semanticId(?:\\\\. (?:type|keys\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:type|value)))?|supplementalSemanticIds(?:\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\)?(?:\\\\. (?:type|keys\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:type|value))))|idShort|id|endpoints\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:interface|protocolinformation\\\\.href)))]|\\\\$smdesc#(?:semanticId(?:\\\\. (?:type|
keys\\\\\\\\(?:[1-9][0-9]*)?\\\\\\\\\\\\\\\\.(?:type|value)))?|supplementalSemanticIds(?:\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\)?(?:\\\\. (?:type|keys\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:type|value))))|idShort|id|endpoints\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\\\\\\\\\.(?:interface|protocolinformation\\\\.href)))$\"
    },
    \"FragmentFieldIdentifier\": {
        \"type\": \"string\",
        \"pattern\":
\"^((?:\\\\$aas#(?:idShort|assetInformation\\\\.assetType|assetInformation\\\\.globalAssetId|assetI
nformation\\\\.specificAssetIds(?:\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\(?:\\\\. externalSubjectId(?:\\\\. keys(?:\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\)?))?)?|submodels(?:\\\\\\\\(?:[1-9][0-9]*)?\\\\\\\\(?:\\\\. keys(?:\\\\\\\\(?:[1-9][0-
9]*)?\\\\\\\\)?))?)|\\\\$sm#(?:semanticId(?:\\\\. keys(?:\\\\\\\\(?:[1-9][0-

```

```

9]*)?\\])?)?|supplementalSemanticIds(?:\\[(?:0|[1-9][0-9]*)?\\](?:\\.keys(?:\\[(?:0|[1-
9][0-9]*)?\\])?)?)?|idShort)\\$sme(?:\\. [A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-
9_]?)?(?:\\[(?:0|[1-9][0-9]*)?\\])*(?:\\. [A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-
9_]?)?(?:\\[(?:0|[1-9][0-9]*)?\\])**)?(?:#(?:semanticId(?:\\.keys(?:\\[(?:0|[1-9][0-
9]*)?\\])?)?|supplementalSemanticIds(?:\\[(?:0|[1-9][0-9]*)?\\](?:\\.keys(?:\\[(?:0|[1-
9][0-
9]*)?\\])?)?)?|idShort|value|valueType|language))?)?\\$cd#idShort|\\$aasdesc#(?:idShort|desc
ription|displayName|extension|administration|assetKind|assetType|globalAssetId|specificAsse
tIds(?:\\[(?:0|[1-9][0-9]*)?\\](?:\\.externalSubjectId(?:\\.keys(?:\\[(?:0|[1-9][0-
9]*)?\\])?)?)?)?|endpoints(?:\\[(?:0|[1-9][0-9]*)?\\])?|submodelDescriptors(?:\\[(?:0|[1-
9][0-9]*)?\\](?:\\. (?:semanticId(?:\\.keys(?:\\[(?:0|[1-9][0-
9]*)?\\])?)?|supplementalSemanticIds(?:\\[(?:0|[1-9][0-9]*)?\\](?:\\.keys(?:\\[(?:0|[1-
9][0-9]*)?\\])?)?)?)?|idShort|endpoints(?:\\[(?:0|[1-9][0-
9]*)?\\])?)?)?)?|\\$smdesc#(?:semanticId(?:\\.keys(?:\\[(?:0|[1-9][0-
9]*)?\\])?)?|supplementalSemanticIds(?:\\[(?:0|[1-9][0-9]*)?\\](?:\\.keys(?:\\[(?:0|[1-
9][0-9]*)?\\])?)?)?)?|idShort|endpoints(?:\\[(?:0|[1-9][0-9]*)?\\])?)?)\\$")
},
"hexLiteralPattern": {
  "type": "string",
  "pattern": "^16#[0-9A-F]+$"
},
"dateTimeLiteralPattern": {
  "type": "string",
  "pattern": "^(?:([0-9]{4}-(?:([0-9]{2}|1[02])-(?:[0-9]{1}|12)[0-
9]{3}[01])|([0-9]{4}69|11)-(?:[0-9]{1}|12)[0-9]{30}|02-(?:[0-9]{1}|1[0-9]|2[0-8]))|([0-9]{2}([0-9]{4}8|
[2468][048]|13579)[26])|([0-9]{2}468)[048]|13579)[26])00)-02-
29))T(?:([0-9]{1}[0-9]|2[0-3]):[0-5][0-9](?:[0-5][0-9])?(?:\\. [0-
9]+)?|24:00(?::00)?(?:\\.0+)?(?:Z|[-+](?:([0-9]{1}|1[0-3]):[0-5][0-9]|14:00))?)\\$"
},
"timeLiteralPattern": {
  "type": "string",
  "pattern": "^(?:([0-9]{1}[0-9]|2[0-3]):[0-5][0-9](?:[0-5][0-9])?(?:\\. [0-
9]+)?|24:00(?::00)?(?:\\.0+)?(?:Z|[-+](?:([0-9]{1}|1[0-3]):[0-5][0-9]|14:00))?)\\$"
},
"dateTimeAttributeItem": {
  "type": "object",
  "properties": {
    "GLOBAL": {
      "type": "string",
      "enum": [
        "LOCALNOW",
        "UTCNOW",
        "CLIENTNOW"
      ]
    }
  }
},
"required": [
  "GLOBAL"
],
"additionalProperties": false
},
"dateTimeOperand": {

```

```

"type": "object",
"properties": {
  "$dateTimeVal": {
    "$ref": "#/definitions/dateTimeLiteralPattern"
  },
  "$dateTimeCast": {
    "$ref": "#/definitions/stringValue"
  },
  "$attribute": {
    "$ref": "#/definitions/dateTimeAttributeItem"
  }
},
"oneOf": [
  {
    "required": [
      "$dateTimeVal"
    ]
  },
  {
    "required": [
      "$dateTimeCast"
    ]
  },
  {
    "required": [
      "$attribute"
    ]
  }
],
"additionalProperties": false
},
"fieldOperand": {
  "type": "object",
  "properties": {
    "$field": {
      "$ref": "#/definitions/FieldIdentifier"
    }
  },
  "required": [
    "$field"
  ],
  "additionalProperties": false
},
"numericalOperand": {
  "type": "object",
  "properties": {
    "$numVal": {
      "type": "number"
    },
    "$numCast": {
      "$ref": "#/definitions/Value"
    }
  },

```

```

    "$dayOfWeek": {
      "$ref": "#/definitions/dateTimeOperand"
    },
    "$dayOfMonth": {
      "$ref": "#/definitions/dateTimeOperand"
    },
    "$month": {
      "$ref": "#/definitions/dateTimeOperand"
    },
    "$year": {
      "$ref": "#/definitions/dateTimeOperand"
    }
  },
  "oneOf": [
    {
      "required": [
        "$numVal"
      ]
    },
    {
      "required": [
        "$numCast"
      ]
    },
    {
      "required": [
        "$dayOfWeek"
      ]
    },
    {
      "required": [
        "$dayOfMonth"
      ]
    },
    {
      "required": [
        "$month"
      ]
    },
    {
      "required": [
        "$year"
      ]
    }
  ],
  "additionalProperties": false
},
"hexOperand": {
  "type": "object",
  "properties": {
    "$hexVal": {
      "$ref": "#/definitions/hexLiteralPattern"
    }
  }
}

```

```

    },
    "$hexCast": {
      "$ref": "#/definitions/Value"
    }
  },
  "oneOf": [
    {
      "required": [
        "$hexVal"
      ]
    },
    {
      "required": [
        "$hexCast"
      ]
    }
  ],
  "additionalProperties": false
},
"boolOperand": {
  "type": "object",
  "properties": {
    "$boolean": {
      "type": "boolean"
    },
    "$boolCast": {
      "$ref": "#/definitions/Value"
    }
  },
  "oneOf": [
    {
      "required": [
        "$boolean"
      ]
    },
    {
      "required": [
        "$boolCast"
      ]
    }
  ],
  "additionalProperties": false
},
"timeOperand": {
  "type": "object",
  "properties": {
    "$timeVal": {
      "$ref": "#/definitions/timeLiteralPattern"
    },
    "$timeCast": {
      "anyOf": [
        {

```

```

        "$ref": "#/definitions/stringValue"
      },
      {
        "$ref": "#/definitions/dateTimeOperand"
      }
    ]
  },
  "oneOf": [
    {
      "required": [
        "$timeVal"
      ]
    },
    {
      "required": [
        "$timeCast"
      ]
    }
  ],
  "additionalProperties": false
},
"Value": {
  "type": "object",
  "properties": {
    "$field": {
      "$ref": "#/definitions/FieldIdentifier"
    },
    "$strVal": {
      "$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/stringValue"
},
"$timeCast": {
  "anyOf": [
    {
      "$ref": "#/definitions/stringValue"
    },
    {
      "$ref": "#/definitions/dateTimeOperand"
    }
  ]
},
"$dayOfWeek": {
  "$ref": "#/definitions/dateTimeOperand"
},
"$dayOfMonth": {
  "$ref": "#/definitions/dateTimeOperand"
},
"$month": {
  "$ref": "#/definitions/dateTimeOperand"
},
"$year": {
  "$ref": "#/definitions/dateTimeOperand"
}
},
"oneOf": [
  {
    "required": [
      "$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/FieldIdentifier"
    },
    "$strVal": {
      "$ref": "#/definitions/standardString"
    },
    "$strCast": {
      "$ref": "#/definitions/Value"
    },
    "$attribute": {
      "$ref": "#/definitions/attributeItem"
    }
  },
  "oneOf": [
    {
      "required": [
        "$field"
      ]
    },
    {
      "required": [
        "$strVal"
      ]
    }
  ],

```

```

    {
      "required": [
        "$strCast"
      ]
    },
    {
      "required": [
        "$attribute"
      ]
    }
  ],
  "additionalProperties": false
},
"stringNonFieldValue": {
  "type": "object",
  "properties": {
    "$strVal": {
      "$ref": "#/definitions/standardString"
    },
    "$strCast": {
      "$ref": "#/definitions/Value"
    },
    "$attribute": {
      "$ref": "#/definitions/attributeItem"
    }
  },
  "oneOf": [
    {
      "required": [
        "$strVal"
      ]
    },
    {
      "required": [
        "$strCast"
      ]
    },
    {
      "required": [
        "$attribute"
      ]
    }
  ],
  "additionalProperties": false
},
"comparisonItems": {
  "$ref": "#/definitions/equalityComparisonItems"
},
"stringItems": {
  "anyOf": [
    {
      "type": "array",

```

```

        "minItems": 2,
        "maxItems": 2,
        "items": [
            {
                "$ref": "#/definitions/stringValue"
            },
            {
                "$ref": "#/definitions/stringNonFieldValue"
            }
        ]
    },
    {
        "type": "array",
        "minItems": 2,
        "maxItems": 2,
        "items": [
            {
                "$ref": "#/definitions/stringNonFieldValue"
            },
            {
                "$ref": "#/definitions/stringValue"
            }
        ]
    },
    {
        "type": "array",
        "minItems": 2,
        "maxItems": 2,
        "items": [
            {
                "$ref": "#/definitions/fieldOperand"
            },
            {
                "$ref": "#/definitions/fieldOperand"
            }
        ]
    }
]
},
"numericalComparisonItems": {
    "oneOf": [
        {
            "type": "array",
            "minItems": 2,
            "maxItems": 2,
            "items": {
                "$ref": "#/definitions/numericalOperand"
            }
        },
        {
            "type": "array",
            "minItems": 2,

```

```

    "maxItems": 2,
    "items": [
      {
        "$ref": "#/definitions/numericalOperand"
      },
      {
        "$ref": "#/definitions/fieldOperand"
      }
    ]
  },
  {
    "type": "array",
    "minItems": 2,
    "maxItems": 2,
    "items": [
      {
        "$ref": "#/definitions/fieldOperand"
      },
      {
        "$ref": "#/definitions/numericalOperand"
      }
    ]
  }
],
"hexComparisonItems": {
  "oneOf": [
    {
      "type": "array",
      "minItems": 2,
      "maxItems": 2,
      "items": {
        "$ref": "#/definitions/hexOperand"
      }
    },
    {
      "type": "array",
      "minItems": 2,
      "maxItems": 2,
      "items": [
        {
          "$ref": "#/definitions/hexOperand"
        },
        {
          "$ref": "#/definitions/fieldOperand"
        }
      ]
    }
  ],
  {
    "type": "array",
    "minItems": 2,
    "maxItems": 2,

```

```

        "items": [
            {
                "$ref": "#/definitions/fieldOperand"
            },
            {
                "$ref": "#/definitions/hexOperand"
            }
        ]
    },
    ],
    },
    "boolComparisonItems": {
        "oneOf": [
            {
                "type": "array",
                "minItems": 2,
                "maxItems": 2,
                "items": {
                    "$ref": "#/definitions/boolOperand"
                }
            },
            {
                "type": "array",
                "minItems": 2,
                "maxItems": 2,
                "items": [
                    {
                        "$ref": "#/definitions/boolOperand"
                    },
                    {
                        "$ref": "#/definitions/fieldOperand"
                    }
                ]
            },
            {
                "type": "array",
                "minItems": 2,
                "maxItems": 2,
                "items": [
                    {
                        "$ref": "#/definitions/fieldOperand"
                    },
                    {
                        "$ref": "#/definitions/boolOperand"
                    }
                ]
            }
        ]
    },
    ],
    },
    "dateTimeComparisonItems": {
        "oneOf": [
            {

```

```

        "type": "array",
        "minItems": 2,
        "maxItems": 2,
        "items": {
            "$ref": "#/definitions/dateTimeOperand"
        }
    },
    {
        "type": "array",
        "minItems": 2,
        "maxItems": 2,
        "items": [
            {
                "$ref": "#/definitions/dateTimeOperand"
            },
            {
                "$ref": "#/definitions/fieldOperand"
            }
        ]
    },
    {
        "type": "array",
        "minItems": 2,
        "maxItems": 2,
        "items": [
            {
                "$ref": "#/definitions/fieldOperand"
            },
            {
                "$ref": "#/definitions/dateTimeOperand"
            }
        ]
    }
],
},
"timeComparisonItems": {
    "oneOf": [
        {
            "type": "array",
            "minItems": 2,
            "maxItems": 2,
            "items": {
                "$ref": "#/definitions/timeOperand"
            }
        },
        {
            "type": "array",
            "minItems": 2,
            "maxItems": 2,
            "items": [
                {
                    "$ref": "#/definitions/timeOperand"
                }
            ]
        }
    ]
}

```

```

        },
        {
            "$ref": "#/definitions/fieldOperand"
        }
    ]
},
{
    "type": "array",
    "minItems": 2,
    "maxItems": 2,
    "items": [
        {
            "$ref": "#/definitions/fieldOperand"
        },
        {
            "$ref": "#/definitions/timeOperand"
        }
    ]
}
]
},
"equalityComparisonItems": {
    "anyOf": [
        {
            "$ref": "#/definitions/stringItems"
        },
        {
            "$ref": "#/definitions/numericalComparisonItems"
        },
        {
            "$ref": "#/definitions/hexComparisonItems"
        },
        {
            "$ref": "#/definitions/boolComparisonItems"
        },
        {
            "$ref": "#/definitions/dateTimeComparisonItems"
        },
        {
            "$ref": "#/definitions/timeComparisonItems"
        }
    ]
},
"orderedComparisonItems": {
    "anyOf": [
        {
            "$ref": "#/definitions/stringItems"
        },
        {
            "$ref": "#/definitions/numericalComparisonItems"
        },
        {

```

```

    "$ref": "#/definitions/hexComparisonItems"
  },
  {
    "$ref": "#/definitions/dateTimeComparisonItems"
  },
  {
    "$ref": "#/definitions/timeComparisonItems"
  }
]
},
"matchExpression": {
  "type": "object",
  "properties": {
    "$match": {
      "type": "array",
      "minItems": 1,
      "items": {
        "$ref": "#/definitions/matchExpression"
      }
    },
    "$eq": {
      "$ref": "#/definitions/equalityComparisonItems"
    },
    "$ne": {
      "$ref": "#/definitions/equalityComparisonItems"
    },
    "$gt": {
      "$ref": "#/definitions/orderedComparisonItems"
    },
    "$ge": {
      "$ref": "#/definitions/orderedComparisonItems"
    },
    "$lt": {
      "$ref": "#/definitions/orderedComparisonItems"
    },
    "$le": {
      "$ref": "#/definitions/orderedComparisonItems"
    },
    "$contains": {
      "$ref": "#/definitions/stringItems"
    },
    "$starts-with": {
      "$ref": "#/definitions/stringItems"
    },
    "$ends-with": {
      "$ref": "#/definitions/stringItems"
    },
    "$regex": {
      "$ref": "#/definitions/stringItems"
    }
  },
  "oneOf": [

```

```

{
  "required": [
    "$eq"
  ]
},
{
  "required": [
    "$ne"
  ]
},
{
  "required": [
    "$gt"
  ]
},
{
  "required": [
    "$ge"
  ]
},
{
  "required": [
    "$lt"
  ]
},
{
  "required": [
    "$le"
  ]
},
{
  "required": [
    "$contains"
  ]
},
{
  "required": [
    "$starts-with"
  ]
},
{
  "required": [
    "$ends-with"
  ]
},
{
  "required": [
    "$regex"
  ]
},
{
  "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": {
            "$ref": "#/definitions/equalityComparisonItems"
        },
        "$ne": {
            "$ref": "#/definitions/equalityComparisonItems"
        },
        "$gt": {
            "$ref": "#/definitions/orderedComparisonItems"
        },
        "$ge": {
            "$ref": "#/definitions/orderedComparisonItems"
        },
        "$lt": {
            "$ref": "#/definitions/orderedComparisonItems"
        },
        "$le": {
            "$ref": "#/definitions/orderedComparisonItems"
        },
        "$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": {
    "$ref": "#/definitions/ReferenceIdentifier"
  }
},
"additionalProperties": false
},
"objectItem": {
  "oneOf": [
    {
      "required": [
        "ROUTE"
      ]
    },
    {
      "required": [
        "IDENTIFIABLE"
      ]
    },
    {
      "required": [
        "REFERABLE"
      ]
    },
    {
      "required": [
        "FRAGMENT"
      ]
    },
    {
      "required": [
        "DESCRIPTOR"
      ]
    }
  ]
}

```

```

    ]
  }
],
"properties": {
  "ROUTE": {
    "type": "string"
  },
  "IDENTIFIABLE": {
    "$ref": "#/definitions/IdentifiableIdentifier"
  },
  "REFERABLE": {
    "$ref": "#/definitions/ReferableIdentifier"
  },
  "FRAGMENT": {
    "$ref": "#/definitions/FragmentIdentifier"
  },
  "DESCRIPTOR": {
    "$ref": "#/definitions/DescriptorIdentifier"
  }
},
"additionalProperties": false
},
"rightsEnum": {
  "type": "string",
  "enum": [
    "CREATE",
    "READ",
    "UPDATE",
    "DELETE",
    "EXECUTE",
    "VIEW",
    "ALL"
  ]
},
"additionalProperties": false
},
"ACL": {
  "type": "object",
  "properties": {
    "ATTRIBUTES": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/attributeItem"
      }
    },
    "USEATTRIBUTES": {
      "type": "array",
      "items": {
        "type": "string"
      }
    }
  },
  "RIGHTS": {
    "type": "array",

```

```

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

```

```

"FORMULA": {
  "$ref": "#/definitions/logicalExpression",
  "additionalProperties": false
},
"USEFORMULA": {
  "type": "string"
},
"FILTER": {
  "$ref": "#/definitions/SecurityQueryFilter",
  "additionalProperties": false
},
"FILTERLIST": {
  "type": "array",
  "items": {
    "$ref": "#/definitions/SecurityQueryFilter"
  }
},
},
"allOf": [
  {
    "oneOf": [
      {
        "required": [
          "ACL"
        ]
      },
      {
        "required": [
          "USEACL"
        ]
      }
    ]
  },
  {
    "anyOf": [
      {
        "required": [
          "OBJECTS"
        ]
      },
      {
        "required": [
          "USEOBJECTS"
        ]
      }
    ]
  },
  {
    "oneOf": [
      {
        "required": [
          "FORMULA"

```

```

        ]
      },
      {
        "required": [
          "USEFORMULA"
        ]
      }
    ]
  },
  ],
  "additionalProperties": false
},
"SecurityQueryFilter": {
  "type": "object",
  "properties": {
    "FRAGMENT": {
      "$ref": "#/definitions/FragmentFieldIdentifier"
    },
    "CONDITION": {
      "$ref": "#/definitions/logicalExpression"
    },
    "USEFORMULA": {
      "type": "string"
    }
  },
  "required": [
    "FRAGMENT"
  ],
  "oneOf": [
    {
      "required": [
        "CONDITION"
      ]
    },
    {
      "required": [
        "USEFORMULA"
      ]
    }
  ],
  "additionalProperties": false
},
"QueryFilter": {
  "type": "object",
  "properties": {
    "$fragment": {
      "$ref": "#/definitions/FragmentFieldIdentifier"
    },
    "$condition": {
      "$ref": "#/definitions/logicalExpression"
    }
  },
  },

```

```

    "required": [
        "$fragment",
        "$condition"
    ],
    "additionalProperties": false
},
"Query": {
    "type": "object",
    "properties": {
        "$select": {
            "type": "string",
            "pattern": "^id$"
        },
        "$condition": {
            "$ref": "#/definitions/logicalExpression"
        },
        "$filters": {
            "type": "array",
            "items": {
                "$ref": "#/definitions/QueryFilter"
            }
        }
    },
    "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
            }
        },
        "USEATTRIBUTES": {
            "type": "array",
            "items": {
                "type": "string"
            }
        }
    },
    "required": [
        "DEFATTRIBUTES",
        "USEATTRIBUTES"
    ],
    "additionalProperties": false
}

```

```

        "required": [
            "name"
        ],
        "oneOf": [
            {
                "required": [
                    "attributes"
                ]
            },
            {
                "required": [
                    "USEATTRIBUTES"
                ]
            }
        ],
        "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
  }
}
}

```

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
FieldIdentifier	<FieldIdentifier>	AAS model elements are strings which start with a \$
FragmentFieldIdentifier	<FieldIdentifierFragment>	AAS model fragments for query/security filtering
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: \$field etc.
stringValue	<stringOperand>	String operations contains, starts-with, ends-with, regex; explicit properties for automatic code generation: \$field etc.
\$field	-	string following the FieldIdentifier pattern
\$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>	string operands can be used as input
\$timeCast	<castToTime>	string or DateTime operands 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
QueryFilter	<QueryFilter>	A single filter block with \$fragment and \$condition
\$condition	<logicalExpression>	Root object for the query condition expression
\$filters	<QueryFilter>*	Optional list of filter blocks evaluated together
\$select	<selectStatement>	Optional expression to control the returned fields. Only 'id' is possible.

Examples for Grammar and JSON Schema

Single comparison

Grammar	JSON Representation
---------	---------------------

<pre> \$aas#idShort \$eq \$aas#assetInformation.assetType </pre>	<pre> { "\$condition": { "\$eq": [{ "\$field": "\$aas#idShort" }, { "\$field": "\$aas#assetInformation.assetType" }] } } </pre>
--	--

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

Grammar	JSON Representation
<pre> \$match(\$sme.Documents[].DocumentClassification .Class#value \$eq "03-01", \$sme.Documents[].DocumentVersion. SMLLanguages[]#language \$eq "nl") </pre>	<pre> { "\$condition": { "\$match": [{ "\$eq": [{ "\$field": "\$sme.Documents[]. DocumentClassification.Class #value" }, { "\$strVal": "03-01" }] }, { "\$eq": [{ "\$field": "\$sme.Documents[]. DocumentVersion.SMLLanguages[] #language" }, { "\$strVal": "nl" }] }] } } </pre>

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

Grammar	JSON Representation
---------	---------------------

```

$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: Find Asset Administration Shells where either the "supplierId" is equal to "aas-1", or the "customerId" is equal to "aas-2", or both.

Grammar

```

$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"
    )
)

```

JSON Representation

```

{
  "$condition": {
    "$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" }
            ]
          }
        ]
      }
    ]
  }
}

```

Signatures

Some use cases of the Asset Administration Shell require the proof that data has not been changed and that it is still the original data of the AAS originator. An example is a device manufacturer supplying to an integrator supplying to a plant operator. The plant operator wants to check the remained integrity of the device manufacturer's AAS.

The AASX package format includes the possibility of signing an AASX package, but this feature is not used very often. AASX packages can also not be protected by AAS security and access rules. This is why signatures are needed as part of the API.

Different levels of API signatures have been investigated by the IDTA TF Security, including JWS (JSON Web Signature) or JAdES (JSON advanced digital signature). This version explains and defines new endpoints /\$signed for AAS, Submodel and ConceptDescription, which provide a plain text JWS.

JWS ist definiert in RFC 7515 (<https://datatracker.ietf.org/doc/html/rfc7515>).

AAS signatures include the AAS JSON content as embedded payload in the JWS. The following header parameters are used:

- alg: The "alg" (algorithm) Header Parameter identifies the cryptographic algorithm used to secure the JWS (see RFC 7515 4.1.1). Currently only RS256 is used.
- typ: The "typ" (type) Header Parameter is used by JWS applications to declare the media type [IANA.MediaType] of this complete JWS (see RFC 7515 4.1.9). Currently only JWS is used.
- x5c: The "x5c" (X.509 certificate chain) Header Parameter contains the X.509 public key certificate or certificate chain [RFC5280] corresponding to the key used to digitally sign the JWS (see RFC 7515 4.1.6). This may be a single certificate or certificate chain. The related root certificate must be exchanged by the business partners by an additional secure channel.
- sigT: The "sigT" (signature timestamp) Header Parameter contains the ISO-8601 timestamp when the JWS was created. This may be used to check for newer content and if certificates have been valid at signature time.
- sid: The "sid" (signature identifier) Header Parameter contains a unique GUID to make the storage of JWS easier.

JWS can be decoded by jwt.io. A JWS example and its decoded content can be seen below.

JWS:

[illegible]

jh4a213UUZVbnRGVlK2N0kwazdzRVJjdER0eDg2TEhScUZhUmZSejRHWjZCaWph0VhiSkhLI119.eyJhZG1pbmldHJhdGlvbiI6eyJyZXZpc2lvbiI6IjAiLCJ2ZXJzaW9uIjoimiJ9LCJkaXNwbGF5TmFtZSI6W3sibGFuZ3VhZ2U0iJlbiIsInRleHQ0iJEZW1vIFN1Ym1vZGVsIn1dLCJpZCI6Imp3c0RlbW8iLCJpZFNob3J0IjoIRGVtb1N1Ym1vZGVsIiwia2luZCI6IlRlbXBsYXRRIiwibW9kZWxUeXB1Ijoiu3VibW9kZWwiLCJzZW1hbnRpY0lkIjp7ImtleXMiOlt7InR5cGUiOijtDwJtb2RlbCIsInZhbHVlIjoiTlEZW1vU3VibW9kZWwifV0sInR5cGUiOiJNb2RlbFJlZmVhZ2U0Ijoia2luZCI6IlRlbEVsZW1lbnRzIjpbeypJpZFNob3J0IjoivGVtcGVyYXR1cmVWYXN1ZSI6Im1vZGVsVHlwZSI6IlByb3BlcnR5IiwidmFsdWUiOiIyMi40IiwidmFsdWVUeXB1IjoieHM6ZG91YmxlIn1dfQ.pjZZN8p10U_ywcyLKo0fnTtzVaa-OUV_b_TYuR1fMQCVZez1TeWbB9rZYpywVg6hXSCPK8YWN9rHJoJV6U_XbkmT0ZdvwaIVTqp88vYtg4QMzonCj5VaXBZhp26PorPbZUDj4mck8KCbXRO3oHskIQN4taRbvsVR6MvaoZWb6pPPuthIK_YUjbkPthepVqQqFtZTGoNtaRwxI7GVjeKmbWvcs3gL5vNzf0jt9Qf4g4C02iR3P99PLegcMRV4X829sduRdDA-lWWf4f9TnsIsBQJGZYiGE30p6WLiXc7TqtehzA10SuR67v2HxQG0F2XwSZFfdSYthT0in9T0fGstLg9Zv_q28av26bprHV7K2hek5aZGuEZHCZsEVENtSzDEDGRgAu5LscYK0aQ73nbVfJkYs0gIT0T5StE5KdDo9LncJxySsBn8k_D-ua7VsLHUr6IwRrnWMilvCnbUbfnE1ad-yBc1FS5B7u3_IYtE_IWRNX9B0EUWx0p03-k33H

Decoded JWS:

```
{
  "alg": "RS256",
  "sid": "2bb11688-873a-4be1-abf1-ed34ae7ea28d",
  "sigT": "2026-06-11T12:39:25Z",
  "typ": "JWS",
  "x5c": [

"MIIDnjCCAgYCCQDG28qiDgqadTANBgkqhkiG9w0BAQsFADAQMq4wDAYDVQQDDAViYXN1eDAgFw0yNjAxMjExOTAzMjlaGA8yMTI1MTIyODE5MDMyOVowEDEOMAwGA1UEAwwFYmFzeXgwggGIMA0GCsqGSIB3DQEBAQUAA4IBjwAwggGKAoIBgQDBaEyTYU94ZN9xPMFA3pQCTkIZOd/gOC0BdXqT3Bj6Y1oeYV6zn54qtC45JzvVnfPPOkACuZiTeoLOJDwS0qForBIBMurTWW86AmT39ammvx+ONKPPBIInuJWsXCSww2CvCXUsA3XHRNdWcMbrZT/onzh3tCr+9An3xVnEbjXPjv1HXEY+LFSOxHpRzIT0mmJfDWChTZUnn00XMce7RzaOcrhtR68P3PXJ7gFz4+ER7Da1gbBhWcQpd3N3MXwSur6Aay7dYx2dNamInkPzLoDwkKGf/PK+xOMT6pGpD9p14XM7Pj/7fTm+/IkbkCz+HnyQ6cQVSv03KNxfXwnh0+qnAWoAU+rcgM77FzLvB78CLb4iFP8A3FNpmeNpI0kHTk/vduRZX7kvU7fulp2/UCpK7HfoXTK2qaac/x1w/zoTFmpquQIbu1ahddTPRRhLzTFwsXdE5xOVAu2PIe6iPwop6vXaLu1Lp5b7zMwqQR0x/i/cNfJJH88sYgqotY/gfKS8CAwEAATANBgkqhkiG9w0BAQsFAAOCAYEAeM6g/f0uhLH9BBLaV7GMzvHURAnRy2gjHxKsmVd72FXLYkktUleOeA9R3oztBgHxG+eo08cU441/fAjb6Pj23sNXt4pTjSH2n6VWjQZi/uFeLoEHcLty4qno1LgonmY+LrA1VrUNGmCKPTaBiqq3o07zj7AGye2G0up07XVI+nyD2+bgR0GEdbI1vDk8Ce5YLhaWLZmTf/ggKSEDzQUbjJh0aJISAYP1jmkB06QPq4WscjN5F5670aEp/YMbVubJu9PX4eSP5E/t/1Zxbxm0SeUAouWLBUT/EkbMsloM43R19vykWNCHHgKAYvD2sHzH+fcR6Z5kX9AM8TV0sD/MNyx6gy1lyjd6owQTXOMRf3L1ZMarQguqrjyetmdaXtBJmTIJu/rfYUEAEgKGLSYVwjKscbhZTrG12qdgrgP0rgdFvZNgWJWK+loHe3xLiJ5JhLhiAIX3n8xkmwQFUntFVY67I0k7sERctDNx86LHRqFaRfRz4GZ6BiJa9XbJHe"

  ]
}
```

JSON Payload of example Submodel:

```
{
  "administration": {
    "revision": "0",
    "version": "2"
  },
  "displayName": [
    {
      "language": "en",
```

```

    "text": "Demo Submodel"
  }
],
"id": "jwsDemo",
"idShort": "DemoSubmodel",
"kind": "Template",
"modelType": "Submodel",
"semanticId": {
  "keys": [
    {
      "type": "Submodel",
      "value": "MyDemoSubmodel"
    }
  ],
  "type": "ModelReference"
},
"submodelElements": [
  {
    "idShort": "TemperatureValue",
    "modelType": "Property",
    "value": "22.4",
    "valueType": "xs:double"
  }
]
}

```

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: <i>Asset Administration Shell</i>	
Operation Name	Description
<u><i>GetAssetAdministrationShell</i></u>	Returns the Asset Administration Shell
<u><i>PutAssetAdministrationShell</i></u>	Replaces the current Asset Administration Shell
<u><i>GetAllSubmodelReferences</i></u>	Returns all Submodel References
<u><i>PostSubmodelReference</i></u>	Creates a Submodel Reference at the Asset Administration Shell
<u><i>DeleteSubmodelReference</i></u>	Deletes a specific Submodel Reference from the Asset Administration Shell
<u><i>GetAssetInformation</i></u>	Returns the Asset Information
<u><i>PutAssetInformation</i></u>	Replaces the Asset Information
<u><i>GetThumbnail</i></u>	Returns the thumbnail file
<u><i>PutThumbnail</i></u>	Replaces the thumbnail file
<u><i>DeleteThumbnail</i></u>	Deletes the thumbnail

Operation GetAssetAdministrationShell

Operation Name	<i>GetAssetAdministrationShell</i>			
Explanation	Returns the Asset Administration Shell			
semanticId	https://admin-shell.io/aas/API/GetAssetAdministrationShell/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>serializationModifier</i>	Defines the format of the response	no	SerializationModifier	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Asset Administration Shell	yes	AssetAdministrationShell	1

Operation PutAssetAdministrationShell

Operation Name	<i>PutAssetAdministrationShell</i>			
Explanation	Replaces the Asset Administration Shell			
semanticId	https://admin-shell.io/aas/API/PutAssetAdministrationShell/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>aas</i>	AssetAdministrationShell	yes	AssetAdministrationShell object	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	AssetAdministrationShell	no	Replaced AssetAdministrationShell	1

Operation GetAllSubmodelReferences

Operation Name	<i>GetAllSubmodelReferences</i>			
Explanation	Returns all Submodel References			
semanticId	https://admin-shell.io/aas/API/GetAllSubmodelReferences/3/2			
Name	Description	Mand.	Type	Card.

Input Parameter				
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Submodel References	yes	Reference	0..*

Operation PostSubmodelReference

Operation Name	<i>PostSubmodelReference</i>			
Explanation	Creates a Submodel Reference at the Asset Administration Shell			
semanticId	https://admin-shell.io/aas/API/PostSubmodelReference/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>submodelRef</i>	Reference to the Submodel	yes	Reference	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Created Submodel Reference	yes	Reference	1

Operation DeleteSubmodelReference

Operation Name	<i>DeleteSubmodelReference</i>			
Explanation	Deletes the Submodel Reference from the Asset Administration Shell			
semanticId	https://admin-shell.io/aas/API/DeleteSubmodelReference/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>submodelId</i>	The unique ID of the Submodel for the reference to be deleted	yes	Identifier	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1

Operation GetAssetInformation

Operation Name	<i>GetAssetInformation</i>			
Explanation	Returns the Asset Information			
semanticId	https://admin-shell.io/aas/API/GetAssetInformation/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Asset Information	yes	AssetInformation	1

Operation PutAssetInformation

Operation Name	<i>PutAssetInformation</i>			
Explanation	Replaces the Asset Information			
semanticId	https://admin-shell.io/aas/API/PutAssetInformation/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>assetInfo</i>	Asset Information object	yes	AssetInformation	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1

Operation GetThumbnail

Operation Name	<i>GetThumbnail</i>			
Explanation	Returns the thumbnail file			
semanticId	https://admin-shell.io/aas/API/GetThumbnail/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested thumbnail file	yes	File Content	1

Operation PutThumbnail

Operation Name	<i>PutThumbnail</i>			
Explanation	Replaces the thumbnail file			
semanticId	https://admin-shell.io/aas/API/PutThumbnail/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>file</i>	Thumbnail file	yes	File Content	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1

Operation DeleteThumbnail

Operation Name	<i>DeleteThumbnail</i>			
Explanation	Deletes the thumbnail file			
semanticId	https://admin-shell.io/aas/API/DeleteThumbnail/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1

Submodel Interface and Operations

Submodel Interface

Interface: <i>Submodel</i>	
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

<u>DeleteFileByPath</u>	Deletes the file of an existing submodel element at a specified path within the submodel element hierarchy
<u>PutSubmodel</u>	Replaces the Submodel
<u>PatchSubmodel</u>	Updates the Submodel
<u>PostSubmodelElement</u>	Creates a new submodel element as a child of the submodel. The idShort of the new submodel element must be set in the payload.
<u>PostSubmodelElementByPath</u>	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.
<u>PutSubmodelElementByPath</u>	Creates a new or replaces an existing submodel element at a specified path within the submodel element hierarchy
<u>PatchSubmodelElementByPath</u>	Updates an existing submodel element or creates a new submodel element at a specified path within the submodel element hierarchy
<u>GetSubmodelElementValueByPath</u>	Returns the value of the submodel element at a specified path according to the protocol-specific RAW-value payload
<u>DeleteSubmodelElementByPath</u>	Deletes a submodel element at a specified path within submodel element hierarchy
<u>InvokeOperationSync</u>	Synchronously invokes an Operation at a specified path with a client timeout in ms
<u>InvokeOperationAsync</u>	Asynchronously invokes an Operation at a specified path with a client timeout in ms
<u>GetOperationAsyncStatus</u>	Returns the current status of an asynchronously invoked operation
<u>GetOperationAsyncResult</u>	Returns the OperationResult of an asynchronously invoked operation

Operation GetSubmodel

Operation Name	<i>GetSubmodel</i>			
Explanation	Returns the Submodel			
semanticId	https://admin-shell.io/aas/API/GetSubmodel/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>serializationModifier</i>	Defines the format of the response	no	<u>SerializationModifier</u>	1
Output Parameter				
<i>statusCode</i>	Status code	yes	<u>StatusCode</u>	1

payload	Requested Submodel	yes	Submodel	1
---------	--------------------	-----	--------------------------	---

Operation GetAllSubmodelElements

Operation Name	<i>GetAllSubmodelElements</i>			
Explanation	Returns all submodel elements including their hierarchy			
semanticId	https://admin-shell.io/aas/API/GetAllSubmodelElements/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>serializationModifier</i>	Defines the format of the response	no	SerializationModifier	1
<i>limit</i>	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	<i>GetSubmodelElementByPath</i>			
Explanation	Returns a specific submodel element from the Submodel at a specified path			
semanticId	https://admin-shell.io/aas/API/GetSubmodelElementByPath/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>path</i>	idShortPath via relative Reference/Keys to a submodel element	yes	Key	1..*
<i>serializationModifier</i>	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	<i>GetFileByPath</i>			
Explanation	Returns a specific file from the Submodel at a specified path			
semanticId	https://admin-shell.io/aas/API/GetFileByPath/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>path</i>	idShortPath via relative Reference/Keys to a submodel element	yes	Key	1..*
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested file	yes	File Content	0..1

Operation PutFileByPath

Operation Name	<i>PutFileByPath</i>			
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/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>path</i>	idShortPath via relative Reference/Keys to a submodel element	yes	Key	1..*
<i>payload</i>	Replacing file	yes	File Content	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1

Operation DeleteFileByPath

Operation Name	<i>DeleteFileByPath</i>			
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/3/2			

Name	Description	Mand.	Type	Card.
Input Parameter				
<i>path</i>	idShortPath via relative Reference/Keys to a submodel element	yes	Key	1..*
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1

Operation PutSubmodel

Operation Name	<i>PutSubmodel</i>			
Explanation	Replaces the Submodel			
semanticId	https://admin-shell.io/aas/API/PutSubmodel/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>submodel</i>	Submodel object	yes	Submodel	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Replaced submodel	no	Submodel	1

Operation PatchSubmodel

Operation Name	<i>PatchSubmodel</i>			
Explanation	Updates the Submodel			
semanticId	https://admin-shell.io/aas/API/PatchSubmodel/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>serializationModifier</i>	Defines the format of the input Note: values remain unchanged with content=metadata.	no	SerializationModifier	1
<i>submodel</i>	Submodel object	yes	Submodel	1

Output Parameter				
statusCode	Status code	yes	StatusCode	1
payload	Updated submodel	no	Submodel	1

Operation PostSubmodelElement

Operation Name				
<i>PostSubmodelElement</i>				
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/3/2				
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>submodelElement</i>	Submodel element object	yes	SubmodelElement	
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Created submodel element	yes	SubmodelElement	1

Operation PostSubmodelElementByPath

Operation Name				
<i>PostSubmodelElementByPath</i>				
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/3/2				
Name	Description	Mand.	Type	Card.
Input Parameter				

<i>path</i>	idShortPath via relative Reference/Keys to a submodel element under which the new SubmodelElement shall be added	yes	Key	1..*
<i>submodelElement</i>	Submodel element object	yes	SubmodelElement	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	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	<i>PutSubmodelElementByPath</i>			
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/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>path</i>	idShortPath via relative Reference/Keys to a submodel element which shall be replaced	yes	Key	1..*
<i>submodelElement</i>	Submodel element object	yes	SubmodelElement	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	New state of the submodel element	no	SubmodelElement	1

Operation PatchSubmodelElementByPath

Operation Name	<i>PatchSubmodelElementByPath</i>			
Explanation	Updates an existing submodel element at a specified path within the submodel element hierarchy			
semanticId	https://admin-shell.io/aas/API/PatchSubmodelElementByPath/3/2			
Name	Description	Mand.	Type	Card.

Input Parameter				
<i>serializationModifier</i>	Defines the format of the input	no	SerializationModifier	1
	Note: values remain unchanged with content=metadata.			
<i>path</i>	idShortPath via relative Reference/Keys to a submodel element	yes	Key	1..*
<i>submodelElement</i>	Submodel element object	yes	SubmodelElement	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Updated submodel element	no	SubmodelElement	1

Operation GetSubmodelElementValueByPath

Operation Name		<i>GetSubmodelElementValueByPath</i>		
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/3/2		
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>path</i>	idShortPath via relative Reference/Keys to a submodel element	yes	Key	1..*
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested submodel element value	yes	SubmodelElement	1

Operation PatchSubmodelElementValueByPath

Operation Name		<i>PatchSubmodelElementValueByPath</i>		
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/3/2		
Name	Description	Mand.	Type	Card.

Input Parameter				
<i>path</i>	idShortPath via relative Reference/Keys to a submodel element	yes	Key	1..*
<i>payload</i>	The new value of the submodel element	yes	SubmodelElement	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1

Operation DeleteSubmodelElementByPath

Operation Name		<i>DeleteSubmodelElementByPath</i>		
Explanation		Deletes a submodel element at a specified path within the submodel elements hierarchy		
semanticId		https://admin-shell.io/aas/API/DeleteSubmodelElementByPath/3/2		
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>path</i>	idShortPath via relative Reference/Keys to a submodel element	yes	Key	1..*
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1

Operation InvokeOperationSync

Operation Name		<i>InvokeOperationSync</i>		
Explanation		Synchronously invokes an Operation at a specified path		
semanticId		https://admin-shell.io/aas/API/InvokeOperationSync/3/2		
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>path</i>	idShortPath via relative Reference/Keys to a submodel element, in this case an operation	yes	Key	1..*
<i>inputArgument</i>	Input argument	no	OperationVariable	1..*
<i>inoutputArgument</i>	Inoutput argument	no	OperationVariable	1..*

Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	The Operation Result	yes	OperationResult	1

Operation InvokeOperationAsync

Operation Name	<i>InvokeOperationAsync</i>			
Explanation	Asynchronously invokes an Operation at a specified path			
semanticId	https://admin-shell.io/aas/API/InvokeOperationAsync/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>path</i>	idShortPath via relative Reference/Keys to a submodel element, in this case an operation	yes	Key	1..*
<i>inputArgument</i>	Input argument	no	OperationVariable	1..*
<i>inoutputArgument</i>	Inoutput argument	no	OperationVariable	1..*
<i>clientTimeoutDuration</i>	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				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	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	<i>GetOperationAsyncStatus</i>			
Explanation	Returns the current status of an asynchronously invoked operation			
semanticId	https://admin-shell.io/aas/API/GetOperationAsnycStatus/3/2			
Name	Description	Mand.	Type	Card.

Input Parameter				
<i>operationHandle</i>	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				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Execution state of the operation	yes	BaseOperationResult	1

Operation GetOperationAsyncResult

Operation Name		<i>GetOperationAsyncResult</i>		
Explanation		Returns the OperationResult of an asynchronously invoked operation		
semanticId		https://admin-shell.io/aas/API/GetOperationAsyncResult/3/2		
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>operationHandle</i>	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				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Operation Result	yes	OperationResult	1

Serialization Interface and Operations

Serialization Interface

Interface: <i>Serialization</i>	
Operation Name	Description
GenerateSerializationByIds	Returns an appropriate serialization based on the specified format (see SerializationFormat).

Operation GenerateSerializationByIds

Operation Name	<i>GenerateSerializationByIds</i>
----------------	-----------------------------------

Explanation	Returns an appropriate serialization based on the specified format (see SerializationFormat).			
semanticId	https://admin-shell.io/aas/API/GenerateSerializationByIds/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>aasIds</i>	The unique ids of the Asset Administration Shells to be contained in the serialization	no	Identifier	1..*
<i>submodelIds</i>	The unique ids of the Submodels to be contained in the serialization	no	Identifier	1..*
<i>includeConceptDescriptions</i>	Include all concept descriptions that are referenced in the Submodels contained in the serialization and that are known to the server Note: This shall also include the concept descriptions that are indirectly referenced by a Submodel, e.g., via the <i>valueList</i> attribute from the IEC 63260 data specification template.	no	boolean	1
<i>serializationFormat</i>	Denotes in which serialization format the requested content shall be delivered	no	SerializationFormat	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Serialization of the requested Asset Administration Shells and/or Submodels with or without ConceptDescriptions in specified <i>SerializationFormat</i> .	yes	Environment	1

Enumeration:	<i>SerializationFormat</i>
Explanation:	Determines the format of serialization, for example JSON or XML. The values are media types conformant to RFC 2046 and registered as described in RFC 6838 (IANA).
Set of:	—
Literal	Explanation

<i>application/json</i>	JSON serialization of the requested data object inside an AAS Environment structure
<i>application/xml</i>	XML serialization of the requested data object inside an AAS Environment structure (default)
<i>application/aasx+xml</i>	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
<i>GetAllAASXPackageIds</i>	Returns a list of available AASX packages at the server
<i>GetAASXByPackageId</i>	Returns a specific AASX package from the server
<i>PostAASXPackage</i>	Creates an AASX package at the server
<i>PutAASXByPackageId</i>	Replaces or creates the AASX package at the server
<i>DeleteAASXByPackageId</i>	Deletes a specific AASX package
<i>PostAsyncAASXPackage</i>	Initiates an asynchronous upload of an AASX package at the server.
<i>GetAasxAsyncStatus</i>	Returns the status of an asynchronously invoked AASX Operation.
<i>GetOperationAsyncResult</i>	Returns the Operation result of an asynchronously invoked AASX operation.

Operation GetAllAASXPackageIds

Operation Name	<i>GetAllAASXPackageIds</i>			
Explanation	Returns a list of available AASX packages at the server			
semanticId	https://admin-shell.io/aas/API/GetAllAASXPackageIds/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>aasId</i>	Identifier of the AAS which must exist in each matching AASX package	no	<u>Identifier</u>	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				

<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	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	<i>GetAASXByPackageId</i>			
Explanation	Returns a specific AASX package from the server			
semanticId	https://admin-shell.io/aas/API/GetAASXByPackageId/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>packageId</i>	Requested package ID from the package list	yes	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>filename</i>	Filename of the AASX package	yes	string	1
<i>payload</i>	Requested AASX package	yes	AASX package	1

Operation PostAASXPackage

Operation Name	<i>PostAASXPackage</i>			
Explanation	Creates an AASX package at the server			
semanticId	https://admin-shell.io/aas/API/PostAASXPackage/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>aasIds</i>	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..*

<i>file</i>	New AASX package	yes	AASX package	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>packageId</i>	New Package ID	yes	string	1

Operation PutAASXByPackageId

Operation Name	<i>PutAASXByPackageId</i>			
Explanation	Replaces or creates the AASX package at the server			
semanticId	https://admin-shell.io/aas/API/PutAASXByPackageId/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>packageId</i>	Package ID from the package list	yes	string	1
<i>aasIds</i>	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..*
<i>file</i>	New AASX package	yes	AASX package	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1

Operation DeleteAASXByPackageId

Operation Name	<i>DeleteAASXByPackageId</i>			
Explanation	Deletes a specific AASX package from the server			
semanticId	https://admin-shell.io/aas/API/DeleteAASXByPackageId/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>packageId</i>	Package ID from the package list	yes	string	1
Output Parameter				

<i>statusCode</i>	Status code	yes	StatusCode	1
-------------------	-------------	-----	----------------------------	---

Operation PostAsyncAASXPackage

Operation Name	<i>PostAsyncAASXPackage</i>			
Explanation	Initiates an asynchronous upload of an AASX package at the server			
semanticId	https://admin-shell.io/aas/API/PostAsyncAASXPackage/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>file</i>	New AASX package	yes	AASX package	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	The returned handle of an operation's asynchronous invocation used to request the current state of the operation's execution	yes	OperationHandle	1

Operation GetAasxAsyncStatus

Operation Name	<i>GetAasxAsyncStatus</i>			
Explanation	Returns the status of an asynchronously invoked AASX Operation			
semanticId	https://admin-shell.io/aas/API/GetAasxAsyncStatus/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>handleId</i>	The returned handle ID of an operation's asynchronous invocation used to request the current state of the operation's execution (UTF8-BASE64-URL-encoded)	yes	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Execution state of the operation	yes	BaseOperationResult	1

Operation GetAasxAsyncResult

Operation Name	GetAasxAsyncResult			
Explanation	Returns the Operation result of an asynchronously invoked AASX operation			
semanticId	https://admin-shell.io/aas/API/GetAasxAsyncResult/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>operationHandle</i>	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				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Operation Result	yes	OperationResult	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

Interface: Asset Administration Shell Registry

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	<i>GetAllAssetAdministrationShellDescriptors</i>			
Explanation	Returns all Asset Administration Shell Descriptors fulfilling the criteria. All parameters are evaluated with AND semantics, except that <i>CreatedFrom</i> and <i>UpdatedFrom</i> are grouped together and evaluated with OR semantics within that group. <i>key</i> and <i>keyIdentifier</i> shall be both available or both missing, they are presenting a key-value-pair.			
semanticId	https://admin-shell.io/aas/API/GetAllAssetAdministrationShellDescriptors/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>assetKind</i>	The kind of the assets to retrieve (Type, Instance, Role, NotApplicable)	no	AssetKind	1
<i>assetType</i>	The type of the assets to retrieve, encoded as unique ID	no	Identifier	1
<i>key</i>	The key of the AssetId The name of the specific asset identifier or the predefined name " <i>globalAssetId</i> " that would refer to the <i>AssetInformation/globalAssetId</i>	no	string	1
<i>keyIdentifier</i>	The key identifier object	no	string	1
<i>createdFrom</i>	Return Shells created at or after the given date	no	DateTimeUtc	1

<i>updatedFrom</i>	Return Shells updated at or after the given date	no	DateTimeUtc	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
<i>payload</i>	List of Asset Administration Shell Descriptors	no	AssetAdministrationShellDescriptor	1..*

Operation GetAssetAdministrationShellDescriptorById

Operation Name	<i>GetAssetAdministrationShellDescriptorById</i>			
Explanation	Returns a specific Asset Administration Shell Descriptor			
semanticId	https://admin-shell.io/aas/API/GetAssetAdministrationShellDescriptorById/3/2			
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
<i>payload</i>	Requested Asset Administration Shell Descriptor	yes	AssetAdministrationShellDescriptor	1

Operation PostAssetAdministrationShellDescriptor

Operation Name	<i>PostAssetAdministrationShellDescriptor</i>			
Explanation	Creates a new Asset Administration Shell Descriptor, i.e., registers an AAS			
semanticId	https://admin-shell.io/aas/API/PostAssetAdministrationShellDescriptor/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
shellDescriptor	Object containing the Asset Administration Shell's identification and endpoint information	yes	AssetAdministrationShellDescriptor	1

Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Created Asset Administration Shell Descriptor	yes	AssetAdministrationShellDescriptor	1

Operation CreateBulkAssetAdministrationShellDescriptors

Operation Name		<i>CreateBulkAssetAdministrationShellDescriptors</i>		
Explanation		Creates multiple new Asset Administration Shell Descriptors, i.e., registers multiple Asset Administration Shells		
semanticId		https://admin-shell.io/aas/API/CreateBulkAssetAdministrationShellDescriptors/3/2		
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		<i>PutAssetAdministrationShellDescriptorById</i>		
Explanation		Creates or replaces an existing Asset Administration Shell Descriptor, i.e., replaces registration information		
semanticId		https://admin-shell.io/aas/API/PutAssetAdministrationShellDescriptorById/3/2		
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
<i>payload</i>	New state of the Asset Administration Shell Descriptor	no	AssetAdministrationShellDescriptor	1

Operation PutBulkAssetAdministrationShellDescriptorsById

Operation Name	<i>PutBulkAssetAdministrationShellDescriptorsById</i>			
Explanation	Updates multiple existing Asset Administration Shell Descriptors or creates them if they do not exist			
semanticId	https://admin-shell.io/aas/API/PutBulkAssetAdministrationShellDescriptorsById/3/2			
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	<i>DeleteAssetAdministrationShellDescriptorById</i>			
Explanation	Deletes an Asset Administration Shell Descriptor, i.e., de-registers an AAS			
semanticId	https://admin-shell.io/aas/API/DeleteAssetAdministrationShellDescriptorById/3/2			
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	<i>DeleteBulkAssetAdministrationShellDescriptorsById</i>			
Explanation	Deletes multiple Asset Administration Shell Descriptors, i.e., de-registers multiple Asset Administration Shells			
semanticId	https://admin-shell.io/aas/API/DeleteBulkAssetAdministrationShellDescriptorsById/3/2			
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/3/2			
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
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

Interface: Submodel Registry	
<u>PutBulkSubmodelDescriptorsByld</u>	Creates or replaces one or more existing submodel descriptors, i.e., replaces registration information of several submodels
<u>DeleteSubmodelDescriptorByld</u>	Deletes a submodel descriptor, i.e., de-registers a submodel
<u>DeleteBulkSubmodelDescriptorsByld</u>	Deletes one or more submodel descriptors, i.e., de-registers several submodels
<u>QuerySubmodelDescriptors</u>	Returns all Submodel Descriptors that conform to a certain input query.

Operation GetAllSubmodelDescriptors

Operation Name		<i>GetAllSubmodelDescriptors</i>		
Explanation		Returns all submodel descriptors fulfilling the criteria. All parameters are evaluated with AND semantics, except that CreatedFrom and UpdatedFrom are grouped together and evaluated with OR semantics within that group.		
semanticId		https://admin-shell.io/aas/API/GetAllSubmodelDescriptors/3/2		
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>createdFrom</i>	Return Submodel Descriptors created at or after the given date	no	<u>DateTimeUtc</u>	1
<i>updatedFrom</i>	Return Submodel Descriptors updated at or after the given date	no	<u>DateTimeUtc</u>	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	<u>StatusCode</u>	1
<i>payload</i>	List of requested submodel descriptors	no	<u>SubmodelDescriptor</u>	1..*

Operation GetSubmodelDescriptorByld

Operation Name		<i>GetSubmodelDescriptorByld</i>		
Explanation		Returns a specific Submodel Descriptor		
semanticId		https://admin-shell.io/aas/API/GetSubmodelDescriptorByld/3/2 =		
Name	Description	Mand.	Type	Card.

Input Parameter				
<i>submodelIdentifier</i>	The Submodel's unique ID	yes	Identifier	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested submodel descriptor	yes	SubmodelDescriptor	1

Operation PostSubmodelDescriptor

Operation Name	<i>PostSubmodelDescriptor</i>			
Explanation	Creates a new submodel descriptor, i.e. registers a submodel			
semanticId	https://admin-shell.io/aas/API/PostSubmodelDescriptor/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>submodel Descriptor</i>	Object containing the Submodel's identification and endpoint information	yes	SubmodelDescriptor	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Created submodel descriptor	yes	SubmodelDescriptor	1

Operation PostBulkSubmodelDescriptors

Operation Name	<i>PostBulkSubmodelDescriptors</i>			
Explanation	Creates one or more new submodel descriptors, i.e., registers several submodels			
semanticId	https://admin-shell.io/aas/API/PostBulkSubmodelDescriptors/3/2			
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	<i>PutSubmodelDescriptorById</i>			
Explanation	Replaces an existing submodel descriptor, i.e., replaces registration information			
semanticId	https://admin-shell.io/aas/API/PutSubmodelDescriptorById/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>submodel Descriptor</i>	Object containing the Submodel's identification and endpoint information	yes	SubmodelDescriptor	1
Output Parameter				
StatusCode	Status code	yes	StatusCode	1
<i>payload</i>	Replaced submodel descriptor	no	SubmodelDescriptor	1

Operation PutBulkSubmodelDescriptorsById

Operation Name	<i>PutBulkSubmodelDescriptorsById</i>			
Explanation	Replaces one or more existing submodel descriptors, i.e., replaces registration information of several submodels			
semanticId	https://admin-shell.io/aas/API/PutBulkSubmodelDescriptorsById/3/2			
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
<i>payload</i>	Replaced submodel descriptor	yes	SubmodelDescriptor	1

Operation DeleteSubmodelDescriptorById

Operation Name	<i>DeleteSubmodelDescriptorById</i>			
Explanation	Deletes a Submodel Descriptor, i.e., de-registers a submodel			
semanticId	https://admin-shell.io/aas/API/DeleteSubmodelDescriptorById/3/2			
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		<i>DeleteBulkSubmodelDescriptorsById</i>		
Explanation		Deletes one or more submodel descriptors, i.e., de-registers several submodels		
semanticId		https://admin-shell.io/aas/API/DeleteBulkSubmodelDescriptorsById/3/2		
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		<i>QuerySubmodelDescriptors</i>		
Explanation		Returns all Submodel Descriptors that conform to a certain input query.		
semanticId		https://admin-shell.io/aas/API/QuerySubmodelDescriptors/3/2		
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
<i>payload</i>	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
GetAllAssetAdministrationShellsRecentChanges	Returns information about recently changed 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
GetAssetAdministrationShellVersionByIdAndDate	Returns a specific Asset Administration Shell at a specific point in time.
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

Interface: Asset Administration Shell Repository[GetAssetAdministrationShellByIdSigned](#)

Returns the JWS for an Asset Administration Shell

Operation GetAllAssetAdministrationShells

Operation Name	<i>GetAllAssetAdministrationShells</i>			
Explanation	Returns all Asset Administration Shells All parameters are evaluated with AND semantics, except that CreatedFrom and UpdatedFrom are grouped together and evaluated with OR semantics within that group. <i>key</i> and <i>keyIdentifier</i> shall be both available or both missing, they are presenting a key-value-pair.			
semanticId	https://admin-shell.io/aas/API/GetAllAssetAdministrationShells/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>key</i>	The key of the AssetId The name of the specific asset identifier or the predefined name "<i>globalAssetId</i>" that would refer to the <i>AssetInformation/globalAssetId</i>	no	string	1
<i>keyIdentifier</i>	The key identifier object	no	string	1
<i>createdFrom</i>	Return Shells created at or after the given date	no	<u>DateTimeUtc</u>	1
<i>updatedFrom</i>	Return Shells updated at or after the given date	no	<u>DateTimeUtc</u>	1
<i>serializationModifier</i>	Defines the format of the response	yes	<u>SerializationModifier</u>	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	<u>StatusCode</u>	1
<i>payload</i>	List of Asset Administration Shells	no	<u>AssetAdministrationShell</u>	1..*

Operation GetAllAssetAdministrationShellsRecentChanges

Operation Name	GetAllAssetAdministrationShellsRecentChanges <<Experimental>>			
Explanation	Returns information about recently changed Asset Administration Shells. All parameters are evaluated with AND semantics, except that CreatedFrom and UpdatedFrom are grouped together and evaluated with OR semantics within that group. key and keyIdentifier shall be both available or both missing, they are presenting a key-value-pair.			
semanticId	https://admin-shell.io/aas/API/GetAllAssetAdministrationShellsRecentChanges/3/2			
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	no	string	1
keyIdentifier	The key identifier object	no	string	1
createdFrom	Return Shells created at or after the given date	no	DateTimeUtc	1
updatedFrom	Return Shells updated at or after the given date	no	DateTimeUtc	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 Change Infos	yes	AssetAdministrationShellRecentChange	1

Operation GetAssetAdministrationShellById

Operation Name	GetAssetAdministrationShellById			
Explanation	Returns a specific Asset Administration Shell			
semanticId	https://admin-shell.io/aas/API/GetAssetAdministrationShellById/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				

<i>id</i>	The Asset Administration Shell's unique ID	yes	Identifier	1
<i>serializationModifier</i>	Defines the format of the response	yes	SerializationModifier	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Asset Administration Shell	yes	AssetAdministrationShell	1

Operation GetAllAssetAdministrationShellsByAssetId

Operation Name	<i>GetAllAssetAdministrationShellsByAssetId</i>			
Explanation	Returns all Asset Administration Shells that are linked to a globally unique asset identifier or to specific asset ID. <i>key</i> and <i>keyIdentifier</i> shall be both available or both missing, they are presenting a key-value-pair.			
semanticId	https://admin-shell.io/aas/API/GetAllAssetAdministrationShellsByAssetId/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>key</i>	The key of the AssetId The name of the specific asset identifier or the predefined name " <i>globalAssetId</i> " that would refer to the <i>AssetInformation/globalAssetId</i>	yes	string	1
<i>keyIdentifier</i>	The key identifier object	yes	string	1
<i>serializationModifier</i>	Defines the format of the response	yes	SerializationModifier	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Asset Administration Shells	no	AssetAdministrationShell	1..*

Operation GetAllAssetAdministrationShellsByIdShort

Operation Name	GetAllAssetAdministrationShellsByIdShort			
Explanation	Returns all Asset Administration Shells with a specific <i>idShort</i>			
semanticId	https://admin-shell.io/aas/API/GetAllAssetAdministrationShellsByIdShort/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>idShort</i>	The Asset Administration Shell's <i>idShort</i>	yes	NameType	1
<i>serializationModifier</i>	Defines the format of the response	yes	SerializationModifier	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Asset Administration Shells	no	AssetAdministrationShell	1..*

Operation GetAssetAdministrationShellVersionByIdAndDate

Operation Name	GetAssetAdministrationShellVersionByIdAndDate			
Explanation	Returns a specific Asset Administration Shell at a specific point in time. If no date is provided, the latest version is returned. Note 1: Asset Administration Shell version in the operation name shall not be confused with the attribute version in the AdministrativeInformation class. The version attribute is not relevant for the versioning of Asset Administration Shells in the context of this operation. Note 2: Any kind of change in an Asset Administration Shell object may lead to a new version of the Asset Administration Shell. The exact definition of what constitutes a new version is not defined by this specification.			
semanticId	https://admin-shell.io/aas/API/GetAssetAdministrationShellVersionByIdAndDate/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>id</i>	The Asset Administration Shell's unique ID	yes	Identifier	1

date	The point in time at which the Asset Administration Shell version is requested	no	DateTimeUtc	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
payload	Requested Asset Administration Shell Version at the specified date	no	AssetAdministrationShell	1

Operation PostAssetAdministrationShell

Operation Name	<i>PostAssetAdministrationShell</i>			
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/3/2			
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	<i>PutAssetAdministrationShellById</i>			
Explanation	Creates or replaces an existing Asset Administration Shell			
semanticId	https://admin-shell.io/aas/API/PutAssetAdministrationShellById/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
id	The Asset Administration Shell's unique ID	yes	Identifier	1

aas	Asset Administration Shell object	yes	AssetAdministrationShell	1
Output Parameter ===== Note: The id of the contained Asset Administration Shell object has to be exactly the same as in the id input parameter. A server shall respond with a ClientErrorBadRequest otherwise. =====				
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/3/2			
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/3/2			
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..*

Operation GetAssetAdministrationShellByIdSigned

Operation Name	<i>GetAssetAdministrationShellByIdSigned</i>			
Explanation	Returns a JWS for a specific Asset Administration Shell.			
semanticId	https://admin-shell.io/aas/API/GetAssetAdministrationShellByIdSigned/3/2			
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
<i>payload</i>	JWS string for requested Asset Administration Shell	yes	string	1

Submodel Repository Interface and Operations

Submodel Repository Interface

Interface: Submodel Repository	
Operation Name	Description
<u>GetAllSubmodels</u>	Returns all Submodels
<u>GetAllSubmodelsRecentChanges</u>	Returns information about recently changed Submodels
<u>GetSubmodelById</u>	Returns a specific Submodel
<u>GetSubmodelVersionByIdAndDate</u>	Returns a specific version of a Submodel from a given point in time
<u>GetAllSubmodelsBySemanticId</u>	Returns all Submodels with a specific SemanticId
<u>GetAllSubmodelsByIdShort</u>	Returns all Submodels with a specific idShort
<u>PostSubmodel</u>	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.
<u>PutSubmodelById</u>	Creates or replaces an existing Submodel

Interface: Submodel Repository	
<u>PatchSubmodelById</u>	Updates an existing submodel
<u>DeleteSubmodelById</u>	Deletes a Submodel
<u>QuerySubmodels</u>	Returns all Submodels that conform to a certain input query
<u>GetSubmodelByIdSigned</u>	Returns the JWS for a specific Submodel

Operation GetAllSubmodels

Operation Name	<i>GetAllSubmodels</i>			
Explanation	Returns all Submodels			
semanticId	https://admin-shell.io/aas/API/GetAllSubmodels/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>semanticId</i>	Identifier of the semantic definition	no	<u>Reference</u>	1
<i>createdFrom</i>	Return Submodels created at or after the given date	no	<u>DateTimeUtc</u>	1
<i>updatedFrom</i>	Return Submodels updated at or after the given date	no	<u>DateTimeUtc</u>	1
<i>serializationModifier</i>	Defines the format of the response	yes	<u>SerializationModifier</u>	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	<u>StatusCode</u>	1
<i>payload</i>	List of Submodels	no	<u>Submodel</u>	1..*

Operation GetAllSubmodelsRecentChanges

Operation Name	<i>GetAllSubmodelsRecentChanges</i> <<Experimental>>
Explanation	Returns information about all Submodels that have been changed recently. All parameters are evaluated with AND semantics, except that CreatedFrom and UpdatedFrom are grouped together and evaluated with OR semantics within that group.
semanticId	https://admin-shell.io/aas/API/GetAllSubmodelsRecentChanges/3/2

Name	Description	Mand.	Type	Card.
Input Parameter				
<i>semanticId</i>	Identifier of the semantic definition	yes	Reference	1
<i>idShort</i>	The Submodel's idShort	yes	NameType	1
<i>createdFrom</i>	Return Submodels created at or after the given date	no	DateTimeUtc	1
<i>updatedFrom</i>	Return Submodels updated at or after the given date	no	DateTimeUtc	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Submodel Change Infos	yes	SubmodelRecentChange	1

Operation GetSubmodelById

Operation Name	<i>GetSubmodelById</i>			
Explanation	Returns a specific Submodel			
semanticId	https://admin-shell.io/aas/API/GetSubmodelById/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>id</i>	The Submodel's unique ID	yes	Identifier	1
<i>serializationModifier</i>	Defines the format of the response	yes	SerializationModifier	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Submodel	yes	Submodel	1

Operation GetSubmodelVersionByIdAndDate

Operation Name	<i>GetSubmodelVersionByIdAndDate</i>
-----------------------	--------------------------------------

Explanation	Returns a specific version of a Submodel from a given point in time. If no date is provided, the latest version is returned.			
	Note 1: Submodel version in the operation name shall not be confused with the attribute version in the AdministrativeInformation of a Submodel. The version attribute is not relevant for the versioning of Submodels in the context of this operation. Note 2: Any kind of change in a Submodel object may lead to a new version of the Submodel. The exact definition of what constitutes a new version is not defined by this specification.			
semanticId	https://admin-shell.io/aas/API/GetSubmodelVersionByIdAndDate/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
id	The Submodel's unique ID	yes	Identifier	1
date	The point in time at which the Submodel version is requested	no	DateTimeUtc	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
<i>payload</i>	Requested Submodel Version at the specified date	no	Submodel	1

Operation GetAllSubmodelsBySemanticId

Operation Name	<i>GetAllSubmodelsBySemanticId</i>			
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/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>semanticId</i>	Identifier of the semantic definition	yes	Reference	1
<i>serializationModifier</i>	Defines the format of the response	yes	SerializationModifier	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1

Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Submodels	no	Submodel	1..*

Operation GetAllSubmodelsByIdShort

Operation Name		<i>GetAllSubmodelsByIdShort</i>		
Explanation		Returns all Submodels with a specific <i>idShort</i>		
semanticId		https://admin-shell.io/aas/API/GetAllSubmodelsByIdShort/3/2		
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>idShort</i>	The Submodel's idShort	yes	NameType	1
<i>serializationModifier</i>	Defines the format of the response	yes	SerializationModifier	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Submodels	no	Submodel	1..*

Operation PostSubmodel

Operation Name		<i>PostSubmodel</i>		
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/3/2		
Name	Description	Mand.	Type	Card.
Input Parameter				
submodel	Submodel object	yes	Submodel	1
Output Parameter				

statusCode	Status code	yes	StatusCode	1
<i>payload</i>	Created Submodel	yes	Submodel	1

Operation PutSubmodelById

Operation Name	<i>PutSubmodelById</i>			
Explanation	Creates a new or replaces an existing Submodel			
semanticId	https://admin-shell.io/aas/API/PutSubmodelById/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
submodel	Submodel object	yes	Submodel	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
<i>payload</i>	New state of the Submodel	no	Submodel	1

Operation PatchSubmodelById

Operation Name	<i>PatchSubmodelById</i>			
Explanation	Updates an existing Submodel			
semanticId	https://admin-shell.io/aas/API/PatchSubmodelById/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>SerializationModifier</i>	Defines the format of the input Note: values remain unchanged with content=metadata.	yes	SerializationModifier	1
<i>submodel</i>	Submodel object	yes	Submodel	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
<i>payload</i>	Updated submodel	no	Submodel	1

Operation DeleteSubmodelById

Operation Name	DeleteSubmodelById			
Explanation	Deletes a Submodel			
semanticId	https://admin-shell.io/aas/API/DeleteSubmodelById/3/2			
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/3/2			
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..*

Operation GetSubmodelByIdSigned

Operation Name	GetSubmodelByIdSigned			
Explanation	Returns the JWS for a specific Submodel			
semanticId	https://admin-shell.io/aas/API/GetSubmodelByIdSigned/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
id	The Submodel's unique ID	yes	Identifier	1

Output Parameter				
statusCode	Status code	yes	StatusCode	1
payload	JWS of the requested Submodel	yes	string	1

Concept Description Repository Interface and Operations

Concept Description Repository Interface

Interface: <i>Concept Description Repository</i>	
Operation Name	Description
GetAllConceptDescriptions	Returns all Concept Descriptions
GetAllConceptDescriptionsRecentChanges	Returns recently changed Concept Descriptions
GetConceptDescriptionById	Returns a specific Concept Description
GetAllConceptDescriptionsByIdShort	Returns all Concept Descriptions with a specific <i>idShort</i>
GetAllConceptDescriptionsByIsCaseOf	Returns all Concept Descriptions with a specific <i>IsCaseOf</i> -reference
GetAllConceptDescriptionsByDataSpecificationReference	Returns all Concept Descriptions with a specific <i>dataSpecification</i> 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 <i>idShort</i> 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
GetConceptDescriptionByIdSigned	Returns the JWS for a specific Concept Description

Operation GetAllConceptDescriptions

Operation Name	<i>GetAllConceptDescriptions</i>
Explanation	Returns all Concept Descriptions
semanticId	https://admin-shell.io/aas/API/GetAllConceptDescriptions/3/2

Name	Description	Mand.	Type	Card.
Input Parameter				
<i>createdFrom</i>	Return Concept Descriptions created at or after the given date	no	DateTimeUtc	1
<i>updatedFrom</i>	Return Concept Descriptions updated at or after the given date	no	DateTimeUtc	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	List of Concept Descriptions	no	ConceptDescription	1..*

Operation GetAllConceptDescriptionsRecentChanges

Operation Name	<i>GetAllConceptDescriptionsRecentChanges</i> <<Experimental>>			
Explanation	Returns all Concept Descriptions that have been changed recently. All parameters are evaluated with AND semantics, except that CreatedFrom and UpdatedFrom are grouped together and evaluated with OR semantics within that group.			
semanticId	https://admin-shell.io/aas/API/GetAllConceptDescriptionsRecentChanges/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>createdFrom</i>	Return Concept Descriptions created at or after the given date	no	DateTimeUtc	1
<i>updatedFrom</i>	Return Concept Descriptions updated at or after the given date	no	DateTimeUtc	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	List of Concept Descriptions	no	ConceptDescriptionRecentChange	1..*

Operation GetConceptDescriptionById

Operation Name	<i>GetConceptDescriptionById</i>			
Explanation	Returns a specific Concept Description			
semanticId	https://admin-shell.io/aas/API/GetConceptDescriptionById/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>id</i>	The Concept Description's unique ID	yes	Identifier	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Concept Description	yes	ConceptDescription	1

Operation GetAllConceptDescriptionsByIdShort

Operation Name	<i>GetAllConceptDescriptionsByIdShort</i>			
Explanation	Returns all Concept Descriptions with a specific <i>idShort</i>			
semanticId	https://admin-shell.io/aas/API/GetAllConceptDescriptionsByIdShort/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>idShort</i>	The Concept Description's idShort	yes	NameType	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Concept Descriptions	no	ConceptDescription	1..*

Operation GetAllConceptDescriptionsByIsCaseOf

Operation Name	<i>GetAllConceptDescriptionsByIsCaseOf</i>			
Explanation	Returns all Concept Descriptions with a specific <i>IsCaseOf</i> reference			
semanticId	https://admin-shell.io/aas/API/GetAllConceptDescriptionsByIsCaseOf/3/2			
Name	Description	Mand.	Type	Card.

Input Parameter				
<i>isCaseOf</i>	IsCaseOf reference	yes	Reference	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Concept Descriptions	no	ConceptDescription	1..*

Operation GetAllConceptDescriptionsByDataSpecificationReference

Operation Name		<i>GetAllConceptDescriptionsByDataSpecificationReference</i>		
Explanation		Returns all Concept Descriptions with a specific <i>dataSpecification</i> reference		
semanticId		https://admin-shell.io/aas/API/GetAllConceptDescriptionsByDataSpecificationReference/3/2		
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>dataSpecification-Reference</i>	<i>DataSpecification</i> reference	yes	Reference	1
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested Concept Descriptions	no	ConceptDescription	1..*

Operation PostConceptDescription

Operation Name		<i>PostConceptDescription</i>		
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/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
conceptDescription	Concept Description object	yes	ConceptDescription	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
<i>payload</i>	Created Concept Description	yes	ConceptDescription	1

Operation PutConceptDescriptionById

Operation Name	<i>PutConceptDescriptionById</i>			
Explanation	Creates or replaces an existing Concept Description			
semanticId	https://admin-shell.io/aas/API/PutConceptDescriptionById/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
conceptDescription	Concept Description object	yes	ConceptDescription	1
Output Parameter				
statusCode	Status code	yes	StatusCode	1
<i>payload</i>	New state of the Concept Description	no	ConceptDescription	1

Operation DeleteConceptDescriptionById

Operation Name	<i>DeleteConceptDescriptionById</i>			
Explanation	Deletes a Concept Description			
semanticId	https://admin-shell.io/aas/API/DeleteConceptDescriptionById/3/2			
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/3/2			
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..*

Operation GetConceptDescriptionByIdSigned

Operation Name	GetConceptDescriptionByIdSigned			
Explanation	Returns the JWS for a specific Concept Description			
semanticId	https://admin-shell.io/aas/API/GetConceptDescriptionByIdSigned/3/2			
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	JWS for the requested Concept Description	yes	string	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

Interface: Asset Administration Shell Basic Discovery	
Operation Name	Description
<u>GetAllAssetAdministrationShellIdsByAssetLink</u>	Returns a list of Asset Administration Shell ids based on asset identifier key-value-pairs
<u>GetAllAssetLinksById</u>	Returns a list of asset identifier key-value-pairs based on a given Asset Administration Shell id
<u>SearchAllAssetAdministrationShellIdsByAssetLink</u>	Returns a list of Asset Administration Shell IDs linked to specific asset identifiers or the global asset ID
<u>PostAllAssetLinksById</u>	Creates or replaces all asset links associated to the Asset Administration Shell.
<u>DeleteAllAssetLinksById</u>	Deletes all asset identifier key-value-pair linked to an Asset Administration Shell

Operation GetAllAssetAdministrationShellIdsByAssetLink

Operation Name		<i>GetAllAssetAdministrationShellIdsByAssetLink</i> <<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/3/2		
Name	Description	Mand.	Type	Card.
Input Parameter				
<i>assetIds</i>	The specific assetId of an asset identifier, which could be the globalAssetId or specificAssetIds. Note: the name for the globalAssetId is defined in Constraint AASd-116 in IDTA-0001. It is the predefined key "globalAssetId" that would refer to the AssetInformation/globalAssetId.	no	<u>SpecificAssetId</u>	1..*
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				

<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	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	<i>GetAllAssetLinksById</i>			
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/3/2			
Name	Description	Mand.	Type	Card.
Input Parameter				
aasIdentifier	The Asset Administration Shell's unique ID	yes	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	Requested asset identifier, which could be the globalAssetId or specificAssetIds. Note: the name for the globalAssetId is defined in Constraint 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	<i>SearchAllAssetAdministrationShellIdsByAssetLink</i>			
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/3/2			
Name	Description	Mand.	Type	Card.

Input Parameter				
<i>assetLinks</i>	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..*
<i>limit</i>	The maximum size of the result set	no	nonNegativeInteger	1
<i>cursor</i>	The position from which to resume a result listing	no	string	1
Output Parameter				
<i>statusCode</i>	Status code	yes	StatusCode	1
<i>payload</i>	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		<i>PostAllAssetLinksById</i>		
Explanation		Creates or replaces all asset links associated to the Asset Administration Shell.		
semanticId		https://admin-shell.io/aas/API/PostAllAssetLinksById/3/2		
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/3/2			
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: <i>Self-Description</i>	
Operation Name	Description
GetSelfDescription	Returns a description object containing the capabilities and supported features of the server.

Operation GetSelfDescription

Operation Name	<i>GetSelfDescription</i>			
Explanation	Returns a description object containing the capabilities and supported features of the server.			
semanticId	https://admin-shell.io/aas/API/GetSelfDescription/3/2			
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.

Descriptor

Class Name	<i>Descriptor</i>
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/3/2

Attribute	Explanation	Type	Card.
<i>description</i>	Description or comments on the element The description can be provided in several languages	MultiLanguageTextType	0..1
<i>displayName</i>	Display name; can be provided in several languages	MultiLanguageNameType	0..1
<i>extension</i>	An extension of the element	Extension	0..*

AssetAdministrationShellDescriptor

Class Name	<i>AssetAdministrationShellDescriptor</i>		
Explanation	Descriptor of an Asset Administration Shell		
Inherits from	Descriptor		
semanticId	https://admin-shell.io/aas/API/DataTypes/AssetAdministrationShellDescriptor/3/2		
Attribute	Explanation	Type	Card.
<i>administration</i>	Administrative information of the Asset Administration Shell	AdministrativeInformation	0..1
<i>assetKind</i>	Denotes whether the asset of the described Asset Administration Shell is of kind "Type", "Instance", "Role", or "NotApplicable"	AssetKind	0..1
<i>assetType</i>	The type of the asset described by the Asset Administration Shell of this Descriptor. See AssetInformation/assetType for further information.	Identifier	0..1
<i>endpoint</i>	Endpoint of the network resource	Endpoint	0..*
<i>globalAssetId</i>	Global reference to the asset the AAS is representing	Identifier	0..1
<i>idShort</i>	Short name of the Asset Administration Shell	NameType	0..1
<i>id</i>	Globally unique identification of the Asset Administration Shell	Identifier	1
<i>specificAssetId</i>	Specific asset identifier	SpecificAssetId	0..*
<i>submodelDescriptor</i>	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/3/2		
Attribute	Explanation	Type	Card.
<i>administration</i>	Administrative information of the Submodel	AdministrativeInformation	0..1
<i>endpoint</i>	Endpoint of the network resource	Endpoint	1..*
<i>idShort</i>	Short name of the Submodel	NameType	0..1
<i>id</i>	Globally unique identification of the Submodel	Identifier	1
<i>semanticId</i>	Identifier of the semantic definition of the Submodel	Reference	0..1
<i>supplementalSemanticId</i>	Identifier of a supplemental semantic definition of the element called supplemental semantic ID of the element	Reference	0..*

Endpoint

Class Name	<i>Endpoint</i>		
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/3/2		
Attribute	Explanation	Type	Card.
<i>protocolInformation</i>	Protocol information of the network resource endpoint	ProtocolInformation	1
<i>interface</i>	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 “3.2”, e.g. the entry for the Asset Administration Shell Interface is “AAS-3.2”.

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* serialisation:

```
"endpoints": [
  {
    "protocolInformation": {
      "endpointAddress": "https://localhost:1234/api/3/submodel",
      "endpointProtocolVersion": [ "1.1" ]
    }
  }
]
```

```

    },
    "interface": "SUBMODEL-3.2"
  },
  {
    "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-3.2"
  }
]

```

ProtocolInformation

Class Name	<i>ProtocolInformation</i>		
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/3/2		
Attribute	Explanation	Type	Card.
<i>href</i>	The endpoint address as a URL	LocatorType	1
<i>endpointProtocol</i>	Either scheme of endpointAddress or scheme + further information. Scheme denotes the highest level of doubtless transmission.	SchemeType	0..1
<i>endpointProtocolVersion</i>	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..*
<i>subprotocol</i>	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

<i>subprotocolBody</i>	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
<i>subprotocolBodyEncoding</i>	If subprotocolBody is present, the encoding might be explicitly defined, otherwise it shall default to subprotocols encoding scheme	NameType	0..1
<i>securityAttributes</i>	Array of securityAttribute objects, each attribute has 3 properties: <pre>{</pre> 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 <pre>}</pre> The securityAttribute objects are treated as possible alternatives (logical “or”)	SecurityAttributeObject	0..*

Class Name	<i>SecurityAttributeObject</i>		
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/3/2		
Attribute	Explanation	Type	Card.
<i>type</i>	Enum security type or standard	SecurityTypeEnum	1
<i>key</i>	Security attribute key according to standard definitions of the security type	string	1
<i>value</i>	Security attribute value e.g. DANE TLSA Ressource Record	string	1

Enumeration	<i>SecurityTypeEnum</i>		
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/3/2		
Literal	ID		
	Explanation		
<i>NONE</i>	https://admin-shell.io/aas/API/DataTypes/SecurityTypeEnum/NONE/3/2		
	No predefined security type available		
<i>RFC_TLSA</i>	https://admin-shell.io/aas/API/DataTypes/SecurityTypeEnum/RFC_TLSA/3/2		
	TLSA according to RFC 6698		
<i>W3C_DID</i>	https://admin-shell.io/aas/API/DataTypes/SecurityTypeEnum/W3C_DID/3/2		
	Decentralized Identifiers according to the W3C Recommendation [7]		

AssetLink

Class Name	<i>AssetLink</i>		
Explanation	Asset identifier derieved from either SpecificAssetId or GlobalAssetId		
Inherits from	-		
semanticId	https://admin-shell.io/aas/API/DataTypes/AssetLink/3/2		
Attribute	Explanation	Type	Card.
<i>name</i>	Name of the Asset identifier, i.e., "globalAssetId", a serial number, manufacturer part ID, or customer part IDs	LabelType	1
<i>value</i>	Value of the Asset Identifier	Identifier	1

ServiceDescription

Class Name	<i>ServiceDescription</i>
-------------------	---------------------------

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/3/2		
Attribute	Explanation	Type	Card.
<i>profiles</i>	List of implemented server specification profiles.	ServiceSpecificationProfile Enum	1..*

Enumeration	<i>ServiceSpecificationProfileEnum</i>		
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/3/2		
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/3/1/AssetAdministrationShellServiceSpecification/SSP-001	Indicates that the server implemented all features of the Asset Administration Shell Service Specification Full Profile in version 3.1.		
https://admin-shell.io/aas/API/3/2/AssetAdministrationShellServiceSpecification/SSP-001	Indicates that the server implemented all features of the Asset Administration Shell Service Specification Full Profile in version 3.2.		
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/3/1/AssetAdministrationShellServiceSpecification/SSP-002	Indicates that the server implemented all features of the Asset Administration Shell Service Specification Read Profile in version 3.1.		

https://admin-shell.io/aas/API/3/2/AssetAdministrationShellServiceSpecification/SSP-002	Indicates that the server implemented all features of the Asset Administration Shell Service Specification Read Profile in version 3.2.
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/3/1/SubmodelServiceSpecification/SSP-001	Indicates that the server implemented all features of the Submodel Service Specification Full Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/SubmodelServiceSpecification/SSP-001	Indicates that the server implemented all features of the Submodel Service Specification Full Profile in version 3.2.
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/3/1/SubmodelServiceSpecification/SSP-002	Indicates that the server implemented all features of the Submodel Service Specification Value Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/SubmodelServiceSpecification/SSP-002	Indicates that the server implemented all features of the Submodel Service Specification Value Profile in version 3.2.
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/3/1/SubmodelServiceSpecification/SSP-003	Indicates that the server implemented all features of the Submodel Service Specification Read Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/SubmodelServiceSpecification/SSP-003	Indicates that the server implemented all features of the Submodel Service Specification Read Profile in version 3.2.
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/3/1/AasxFileServerServiceSpecification/SSP-001	Indicates that the server implemented all details of the AASX File Server Service Specification Full Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/AasxFileServerServiceSpecification/SSP-001	Indicates that the server implemented all details of the AASX File Server Service Specification Full Profile in version 3.2.

https://admin-shell.io/aas/API/3/1/AasxFileServerServiceSpecification/SSP-002	Indicates that the server implemented all details of the AASX File Server Service Specification Read Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/AasxFileServerServiceSpecification/SSP-002	Indicates that the server implemented all details of the AASX File Server Service Specification Read Profile in version 3.2.
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/3/1/AssetAdministrationShellRegistryServiceSpecification/SSP-001	Indicates that the server implemented all details of the Asset Administration Shell Registry Service Specification Full Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRegistryServiceSpecification/SSP-001	Indicates that the server implemented all details of the Asset Administration Shell Registry Service Specification Full Profile in version 3.2.
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-002	Indicates that the server implemented all details of the Asset Administration Shell Registry Service Specification Read Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRegistryServiceSpecification/SSP-002	Indicates that the server implemented all details of the Asset Administration Shell Registry Service Specification Read Profile in version 3.2.
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/3/2/AssetAdministrationShellRegistryServiceSpecification/SSP-003	Indicates that the server implemented all details of the Asset Administration Shell Registry Service Specification Bulk Profile in version 3.2.
https://admin-shell.io/aas/API/3/1/AssetAdministrationShellRegistryServiceSpecification/SSP-004	Indicates that the server implemented all details of the Asset Administration Shell Registry Service Specification Query Profile in version 3.1.

https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRegistryServiceSpecification/SSP-004	Indicates that the server implemented all details of the Asset Administration Shell Registry Service Specification Query Profile in version 3.2.
https://admin-shell.io/aas/API/3/1/AssetAdministrationShellRegistryServiceSpecification/SSP-005	Indicates that the server implemented all details of the Asset Administration Shell Registry Service Specification Minimal Read Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRegistryServiceSpecification/SSP-005	Indicates that the server implemented all details of the Asset Administration Shell Registry Service Specification Minimal Read Profile in version 3.2.
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/3/1/SubmodelRegistryServiceSpecification/SSP-001	Indicates that the server implemented all details of the Submodel Registry Service Specification Full Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/SubmodelRegistryServiceSpecification/SSP-001	Indicates that the server implemented all details of the Submodel Registry Service Specification Full Profile in version 3.2.
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-002	Indicates that the server implemented all details of the Submodel Registry Service Specification Read Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/SubmodelRegistryServiceSpecification/SSP-002	Indicates that the server implemented all details of the Submodel Registry Service Specification Read Profile in version 3.2.
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/3/2/SubmodelRegistryServiceSpecification/SSP-003	Indicates that the server implemented all details of the Submodel Registry Service Specification Bulk Profile in version 3.2.

https://admin-shell.io/aas/API/3/1/SubmodelRegistryServiceSpecification/SSP-004	Indicates that the server implemented all details of the Submodel Registry Service Specification Query Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/SubmodelRegistryServiceSpecification/SSP-004	Indicates that the server implemented all details of the Submodel Registry Service Specification Query Profile in version 3.2.
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/3/1/DiscoveryServiceSpecification/SSP-001	Indicates that the server implemented all details of the Discovery Service Specification Full Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/DiscoveryServiceSpecification/SSP-001	Indicates that the server implemented all details of the Discovery Service Specification Full Profile in version 3.2.
https://admin-shell.io/aas/API/3/2/DiscoveryServiceSpecification/SSP-002	Indicates that the server implemented all details of the Discovery Service Specification Read Profile in version 3.1.
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/3/1/AssetAdministrationShellRepositoryServiceSpecification/SSP-001	Indicates that the server implemented all details of the Asset Administration Shell Repository Service Specification Full Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRepositoryServiceSpecification/SSP-001	Indicates that the server implemented all details of the Asset Administration Shell Repository Service Specification Read Profile in version 3.2.
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/3/1/AssetAdministrationShellRepositoryServiceSpecification/SSP-002	Indicates that the server implemented all details of the Asset Administration Shell Repository Service Specification Read Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRepositoryServiceSpecification/SSP-002	Indicates that the server implemented all details of the Asset Administration Shell Repository Service Specification Read Profile in version 3.2.

https://admin-shell.io/aas/API/3/1/AssetAdministrationShellRepositoryServiceSpecification/SSP-003	Indicates that the server implemented all details of the Asset Administration Shell Repository Service Specification Query Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRepositoryServiceSpecification/SSP-003	Indicates that the server implemented all details of the Asset Administration Shell Repository Service Specification Query Profile in version 3.2.
https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRepositoryServiceSpecification/SSP-004	Indicates that the server implemented all details of the Asset Administration Shell Repository Service Specification Signature Profile in version 3.2.
https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRepositoryServiceSpecification/SSP-005	Indicates that the server implemented all details of the Asset Administration Shell Repository Service Specification Identifiable Profile in version 3.2.
https://admin-shell.io/aas/API/3/0/SubmodelRepositoryServiceSpecification/SP-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/3/1/SubmodelRepositoryServiceSpecification/SP-001	Indicates that the server implemented all details of the Submodel Service Repository Specification Full Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/SubmodelRepositoryServiceSpecification/SP-001	Indicates that the server implemented all details of the Submodel Service Repository Specification Full Profile in version 3.2.
https://admin-shell.io/aas/API/3/0/SubmodelRepositoryServiceSpecification/SP-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/3/1/SubmodelRepositoryServiceSpecification/SP-002	Indicates that the server implemented all details of the Submodel Service Repository Specification Read Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/SubmodelRepositoryServiceSpecification/SP-002	Indicates that the server implemented all details of the Submodel Service Repository Specification Read Profile in version 3.2.
https://admin-shell.io/aas/API/3/0/SubmodelRepositoryServiceSpecification/SP-003	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/SubmodelRepositoryServiceSpecification/SP-003	Indicates that the server implemented all details of the Submodel Service Repository Specification Template Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/SubmodelRepositoryServiceSpecification/SP-003	Indicates that the server implemented all details of the Submodel Service Repository Specification Template Profile in version 3.2.
https://admin-shell.io/aas/API/3/0/SubmodelRepositoryServiceSpecification/SP-004	Indicates that the server implemented all details of the Submodel Service Repository Specification Template Read Profile in version 3.0.
https://admin-shell.io/aas/API/3/1/SubmodelRepositoryServiceSpecification/SP-004	Indicates that the server implemented all details of the Submodel Service Repository Specification Template Read Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/SubmodelRepositoryServiceSpecification/SP-004	Indicates that the server implemented all details of the Submodel Service Repository Specification Template Read Profile in version 3.2.
https://admin-shell.io/aas/API/3/1/SubmodelRepositoryServiceSpecification/SP-005	Indicates that the server implemented all details of the Submodel Service Repository Specification Query Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/SubmodelRepositoryServiceSpecification/SP-005	Indicates that the server implemented all details of the Submodel Service Repository Specification Query Profile in version 3.2.
https://admin-shell.io/aas/API/3/2/SubmodelRepositoryServiceSpecification/SP-006	Indicates that the server implemented all details of the Submodel Service Repository Specification Signature Profile in version 3.2.
https://admin-shell.io/aas/API/3/0/ConceptDescriptionRepositoryServiceSpecification/SSP-001	Indicates that the server implemented all details of the Concept Description Repository Service Specification 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 Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/ConceptDescriptionRepositoryServiceSpecification/SSP-002	Indicates that the server implemented all details of the Concept Description Repository Service Specification Query Profile in version 3.2.

https://admin-shell.io/aas/API/3/1/ConceptDescriptionRepositoryServiceSpecification/SSP-002	Indicates that the server implemented all details of the Concept Description Repository Service Specification Query Profile in version 3.1.
https://admin-shell.io/aas/API/3/2/ConceptDescriptionRepositoryServiceSpecification/SSP-002	Indicates that the server implemented all details of the Concept Description Repository Service Specification Query Profile in version 3.2.
https://admin-shell.io/aas/API/3/2/ConceptDescriptionRepositoryServiceSpecification/SSP-003	Indicates that the server implemented all details of the Concept Description Repository Service Specification Signature Profile in version 3.2.

An example ServiceDescription object might look like the following, indicating that the server supports two profiles at the same time, the Discovery Service Specification Full Profile and the AAS Registry Service Specification Full Profile (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/3/2/DiscoveryServiceSpecification/SSP-001",
    "https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRegistryServiceSpecification/SSP-001"
  ]
}
```

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 4. Simple Data Types used for API-specific Classes

Query Result	Definition	Value Examples
--------------	------------	----------------

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

Recent Changes

Finding recently added or changed elements in the repository is required in several use cases. Since the number of changed objects can be very large the result of such a query might be paginated and only contain the identifiers of the changed objects and a few supporting attributes. The following table describes the data types used for such a result.

Class Name	<i>RecentChange</i>		
Explanation	This class is not part of the metamodel.		
Inherits from	—		
semanticId	https://admin-shell.io/aas/API/DataTypes/RecentChange/3/2		
Attribute	Explanation	Type	Card.
<i>createdAt</i>	The point in time at which the Identifiable object was created	DateTimeUtc	1
<i>updatedAt</i>	The point in time at which the Identifiable object was recently updated	DateTimeUtc	1

Class Name	<i>AssetAdministrationShellRecentChange</i>		
Explanation	This class is not part of the metamodel.		
Inherits from	RecentChange		
semanticId	https://admin-shell.io/aas/API/DataTypes/AssetAdministrationShellRecentChange/3/2		

Attribute	Explanation	Type	Card.
<i>id</i>	Globally unique identification of the Asset Administration Shell	Identifier	1
<i>globalAssetId</i>	Global reference to the asset the AAS is representing	Identifier	0..1
<i>specificAssetId</i>	Specific asset identifier	SpecificAssetId	0..*

Class Name	<i>ConceptDescriptionRecentChange</i>		
Explanation	This class is not part of the metamodel.		
Inherits from	RecentChange		
semanticId	https://admin-shell.io/aas/API/DataTypes/ConceptDescriptionRecentChange/3/2		
Attribute	Explanation	Type	Card.
<i>id</i>	Globally unique identification of the Submodel	Identifier	1

Class Name	<i>SubmodelRecentChange</i>		
Explanation	This class is not part of the metamodel.		
Inherits from	RecentChange		
semanticId	https://admin-shell.io/aas/API/DataTypes/SubmodelRecentChange/3/2		
Attribute	Explanation	Type	Card.
<i>id</i>	Globally unique identification of the Submodel	Identifier	1
<i>semanticId</i>	Identifier of the semantic definition of the Submodel	Reference	0..1
<i>supplementalSemanticId</i>	Identifier of a supplemental semantic definition of the element called supplemental semantic ID of the element	Reference	0..*

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 5](#).

Table 5. Simple Data Types used for API-specific Classes

Primitive	Definition	Value Examples
-----------	------------	----------------

NonNegativeInteger	The <i>nonNegativeInteger</i> datatype as defined by XML Schema Part 2 in version 1.0 [2]	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 6. Primitive Data Types used for the API-specific Classes

Primitive	Definition	Value Examples
CodeType	<i>string</i> with max 32 and min 1 characters	"409" "Bad_UserAccessDenied"
ShortIdType	same as <i>NameType</i> (string with max 128 and min 1 characters) Note: ShortIdType is <i>not</i> 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 <i>NameType</i> (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 7](#) 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 7. 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
ServerInternalServerError	Unexpected error	Yes
ServerNotImplemented	The server has not implemented this API Operation. Intended for cases where API Operations beyond the supported service profiles are requested.	No
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	<i>Result</i>		
Explanation	The result object		
Inherits from	—		
semanticId	https://admin-shell.io/aas/API/DataTypes/Result/3/2		
Attribute	Explanation	Type	Card.
<i>message</i>	Additional message containing information for the requester	Message	0..*

Class Name	<i>Message</i>		
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/3/2		
Attribute	Explanation	Type	Card.
<i>messageType</i>	The message type	MessageTypeEnum	1
<i>text</i>	The message text	string	1
<i>code</i>	Technology-dependent status or error code	CodeType	0..1
<i>correlationId</i>	Identifier to relate several result messages throughout several systems	ShortIdType	0..1
<i>timestamp</i>	Timestamp of the message	dateTime	0..1

Enumeration	<i>MessageTypeEnum</i>		
Explanation	The message type		
semanticId	https://admin-shell.io/aas/API/DataTypes/MessageTypeEnum/3/2		
Literal	Explanation		
<i>Info</i>	Used to inform the user about a certain fact		
<i>Warning</i>	Used for warnings; warnings may lead to errors in the subsequent execution		
<i>Error</i>	Used for handling errors		
<i>Exception</i>	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	<i>OperationRequest</i>		
Explanation	The operation request object		
Inherits from	—		
semanticId*	https://admin-shell.io/aas/API/DataTypes/OperationRequest/3/2		

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	<i>OperationRequestValueOnly</i>		
Explanation	The operation request object		
Inherits from	—		
semanticId*	https://admin-shell.io/aas/API/DataTypes/OperationRequestValueOnly/3/2		
Attribute	Explanation	Type	Card.
inputArguments	Input argument	ValueOnly	0..*
inoutputArguments	InOutput argument	ValueOnly	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	<i>BaseOperationResult</i>		
Explanation	The object containing the intermediate state of an operation		
Inherits from	Result		
semanticId	https://admin-shell.io/aas/API/DataTypes/BaseOperationResult/3/2		
Attribute	Explanation	Type	Card.
<i>executionState</i>	Execution state	ExecutionState	1
<i>success</i>	Flag indicating whether the business operation behind the operation was successful (true) or not (false)	boolean	0..1

OperationResult

Class Name	<i>OperationResult</i>		
Explanation	The operation's invocation result object		
Inherits from	BaseOperationResult		
semanticId	https://admin-shell.io/aas/API/DataTypes/OperationResult/3/2		
Attribute	Explanation	Type	Card.
<i>outputArguments</i>	Output argument	OperationVariable	0..*
<i>inoutputArguments</i>	InOutput argument	OperationVariable	0..*

OperationResultValueOnly

Class Name	<i>OperationResultValueOnly</i>		
Explanation	The operation's invocation result object in the ValueOnly notation.		
Inherits from	BaseOperationResult		
semanticId	https://admin-shell.io/aas/API/DataTypes/OperationResultValueOnly/3/2		
Attribute	Explanation	Type	Card.
<i>outputArguments</i>	Output argument	ValueOnly	0..*
<i>inoutputArguments</i>	InOutput argument	ValueOnly	0..*

Enumeration ExecutionState

Enumeration	<i>ExecutionState</i>		
Explanation	The operation's invocation result state		
semanticId	https://admin-shell.io/aas/API/DataTypes/ExecutionState/3/2		
Literal	Explanation		
<i>Initiated</i>	The operation is ready to be executed (initial state)		
<i>Running</i>	The operation is running		
<i>Completed</i>	The operation is completed		
<i>Canceled</i>	The operation was cancelled externally		
<i>Failed</i>	The operation failed		
<i>Timeout</i>	The operation has timed out due to given client or server timeout		

OperationHandle

Class Name	<i>OperationHandle</i>		
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/3/2		
Attribute	Explanation	Type	Card.
<i>handleId</i>	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 8. *Level Parameters*

Value	Explanation
<i>Deep</i> (Default)	All elements of a requested hierarchy level and all children on all sublevels are returned. Children in this sense are <i>SubmodelElements</i> which are contained at the 'submodelElements' field of <i>Submodels</i> , the 'value' field of <i>SubmodelElementCollections</i> or <i>SubmodelElementLists</i> , the 'statements' field of <i>Entities</i> , or the 'annotations' field of <i>AnnotatedRelationshipElements</i> .
<i>Core</i>	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/v3.2/mappings/mappings.html#_format_normal_in_xml and https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.2/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/v3.2/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/v3.2/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/v3.2/mappings/mappings.html#_format_reference in Part 1 for details.

For Content equal to Path see Clause https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.2/mappings/mappings.html#_format_path_idshortpath_serialization_in_json in Part 1 for details.

Table 9. Content Parameters

Value	Explanation
<i>Normal</i> (Default)	The standard serialization of the model element or child elements is applied.
<i>Metadata</i>	Only metadata of an element or child elements is returned; the value is not .
<i>Value</i>	Only the raw value of the model element or child elements is returned; it is commonly referred to as <i>ValueOnly</i> -serialization.
<i>Reference</i>	Only applicable to Referables. Only the reference to the found element is returned; potential child elements are ignored.
<i>Path</i>	Returns the <i>idShort</i> of the requested element and a list of <i>idShort</i> 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 10. Extent Parameters

Value	Explanation
<i>WithoutBLOBValue</i> (Default)	Only applicable to BLOB-elements; the BLOB content is not returned.
<i>WithBLOBValue</i>	Only applicable to BLOB-elements; the BLOB content is returned as <i>base64</i> -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 11. 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
Concept Description	No	No	No
AssetAdministrationShellDescriptor	No	No	No
SubmodelDescriptor	No	No	No
SubmodelElements			
SubmodelElementCollection	Deep/Core	Normal/ Metadata/ Value/ Reference/ Path	WithoutBLOBValue/ WithBLOBValue

Resource Name	Level Modifier	Content Modifier	Extent Modifier
SubmodelElementList	Deep/Core	Normal/ Metadata/ Value/ Reference/ Path	WithoutBLOBValue/ WithBLOBValue
Entity	Deep/Core	Normal/ Metadata/ Value/ Reference/ Path	WithoutBLOBValue/ WithBLOBValue
BasicEventElement	No	Normal/ Metadata/ Value/ Reference	No
Capability	No	Normal/Reference	No
Operation	No	Normal/Reference	No
DataElements			
Property	No	Normal/ Metadata/ Value/ Reference	No
MultiLanguageProperty	No	Normal/ Metadata/ Value/ Reference	No
Range	No	Normal/ Metadata/ Value/ Reference	No
ReferenceElement	No	Normal/ Metadata/ Value/ Reference	No
RelationshipElement	No	Normal/ Metadata/ Value/ Reference	No
AnnotatedRelationshipElement	No	Normal/ Metadata/ Value/ Reference	No
Blob	No	Normal/ Metadata/ Value/ Reference	WithoutBLOBValue/ WithBLOBValue
File	No	Normal/ Metadata/ Value/ Reference	No

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

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

[2] <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 [here](#) and, for legacy reasons, also at [SwaggerHub](#) until version v3.1.3.

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://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../Part1-MetaModel-Schemas/openapi_3.2.yaml&version=v3.2.0

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://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../Part2-API-Schemas/openapi.yaml&version=v3.2.0>

This schema includes general objects, which are used in the further defined APIs.

The specific OpenAPI documents for the different APIs are listed in the sections for the respective service specifications and profiles in Clause [\[http-rest-api::service-specifications-and-profiles::service-specifications-and-profiles\]](#).

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, hosted by SwaggerHub (legacy versions) and rendered via Github Pages 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 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=W3sibmFtZSI6ICJnbG9iYWxBc3NldElkIiwidmFsdWUiOiAiHR0cDovL2V4YW1wbGUuY29tcGFueS9tZW50In0seyJuYW1lIjogIm15T3duSW50ZXJuYWxBc3NldElkIiwidmFsdWUiOiAiMTIzNDVBQkMifV0"`. 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^[4]. 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 [\[index::metamodel-versions\]](#). 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.

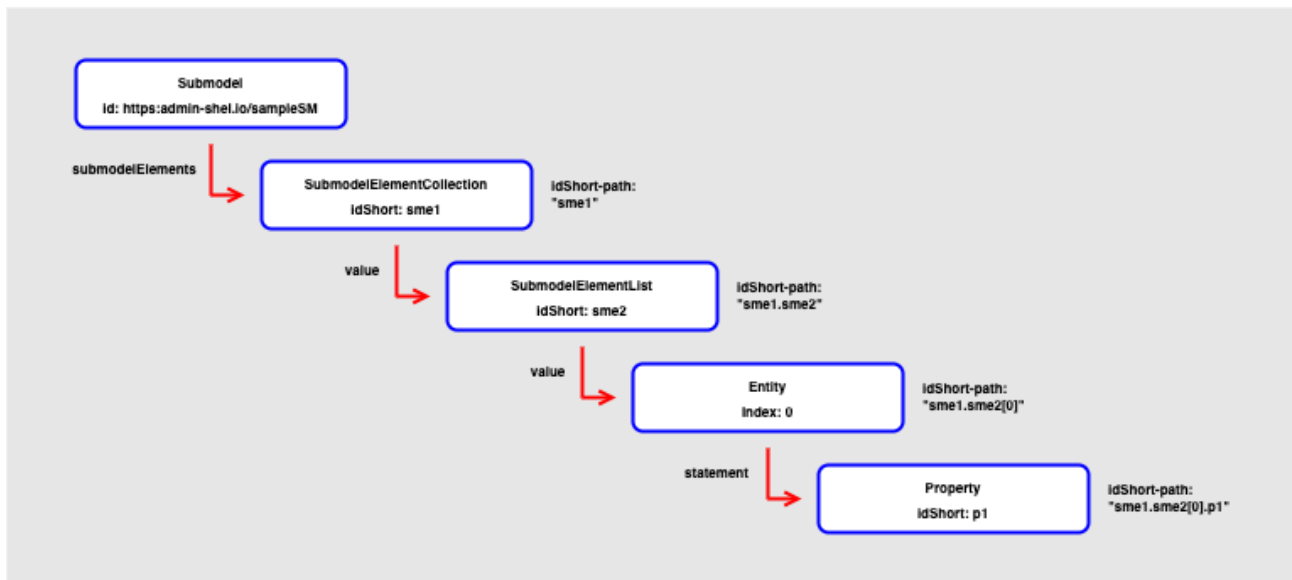


Figure 7. Example Hierarchy of Submodel Elements

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 12. 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 [6] 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 13. Parameters for Pagination

Parameter	Values	Default	Explanation
<i>Cursor</i>	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.
<i>Limit</i>	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	<i>Result</i>		
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.
<i>result</i>	List of returned items. Any kind of Referables is possible, depending on the endpoint which has been requested.	Referable	0..*
<i>paging_metadata</i>	Additional information for the client to, e.g. fetch the next part of the result set.	PagingMetadata	1

Class Name	<i>PagingMetadata</i>		
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.
<i>cursor</i>	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 the [Descriptor Definition](#).

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 14](#) 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 14. 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 '<code><baseUrl>/aas/submodel-refs/{submodelIdentifier}</code>' 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 date={dateTime} is the date and time for which the version of the AAS shall be retrieved, e.g. "2024-01-01T12:00:00Z" for January 1st, 2024 at 12:00:00 UTC.
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
GetAssetAdministrationShellVersionByIdAndDate	GET	/shells/{aasIdentifier}/\$history	base64url-encoded identifier
GetAssetAdministrationShellByIdSigned	GET	/shells/{aasIdentifier}/\$signed	base64url-encoded identifier
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
GetAllSubmodelsRecentChanges	GET	/submodels/\$recent-changes	Pagination

Operation Name	HTTP Verb	REST-Path	Comment (e.g. optional query parameters)
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 date={dateTime} is the date and time for which the version of the Submodel shall be retrieved, e.g. "2024-01-01T12:00:00Z" for January 1st, 2024 at 12:00:00 UTC. Only applicable for the 'normal' and 'value' operations.
PostSubmodel	POST	/submodels	
PutSubmodelById	PUT	/submodels/{submodelIdentifier}	base64url-encoded identifier
PatchSubmodelById	PATCH	/submodels/{submodelIdentifier}	path-suffix=\$metadata/\$value or no suffix for normal
DeleteSubmodelById	DELETE	/submodels/{submodelIdentifier}	base64url-encoded identifier
SubmodelInterface	*	/submodels/{submodelIdentifier}/*	superpath as defined in service specification or profile
QuerySubmodels	POST	/query/submodels	Input query in the request body
GetSubmodelVersionByIdAndDate	GET	/submodels/{submodelIdentifier}/\$history	base64url-encoded identifier
GetSubmodelByIdSigned	GET	/submodels/{submodelIdentifier}/\$signed	base64url-encoded identifier
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

Operation Name	HTTP Verb	REST-Path	Comment (e.g. optional query parameters)
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
GetConceptDescriptionByIdSigned	GET	/concept-descriptions/{cdIdentifier}/\$signed	base64url-encoded identifier
AASX File Server Interface			
GetAllAASXPackageIds	GET	/packages	base64url-encoded identifier Pagination
PostAASXPackage	POST	/packages	
GetAASXByPackageId	GET	/packages/{packageId}	base64url-encoded identifier
PutAASXByPackageId	PUT	/packages/{packageId}	base64url-encoded identifier
DeleteAASXByPackageId	DELETE	/packages/{packageId}	base64url-encoded identifier
PostAsyncAASXPackage	POST	/packages-async	
GetAasxAsyncStatus	GET	/packages-async/status/{handleId}	
GetAasxAsyncResult	GET	/packages-async/result/{handleId}	
Serialization Interface			
GenerateSerializationByIds	GET	/serialization	base64url-encoded identifier; AcceptHeader: application/aasx+xml or application/json oder application/xml
AAS Basic Discovery Interface			

Operation Name	HTTP Verb	REST-Path	Comment (e.g. optional query parameters)
GetAllAssetAdministrationShellIdsByAssetLink	GET	/lookup/shells	base64url-encoded JSON-serialized key-value-pairs ?assetids=... Pagination
GetAllAssetAdministrationShellRecentChanges	GET	/shells/\$recent-changes	key and <i>keyIdentifier</i> are mapped to 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	key and <i>keyIdentifier</i> are mapped to base64url-encoded JSON-serialized key-value-pairs ?assetids=... 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
Submodel Registry Interface	*	/shell-descriptors/{aasIdentifier}/submodelDescriptors/*	superpath as defined in Service Specification or Profile

Operation Name	HTTP Verb	REST-Path	Comment (e.g. optional query parameters)
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
DeleteSubmodelDescriptorById	DELETE	/submodel-descriptors/{submodelIdentifier}	base64url-encoded identifier
QuerySubmodelDescriptors	POST	/query/submodel-descriptors	Input query in the request body

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

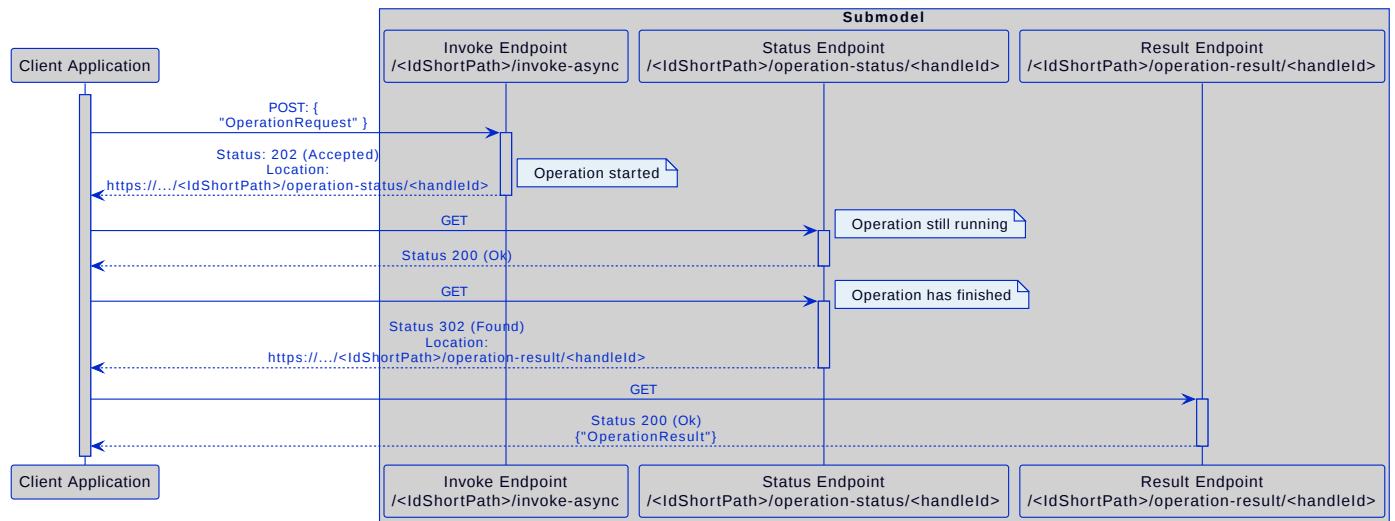


Figure 8. Sequence for asynchronous invocations of the SubmodelElement 'Operation'

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.

Asynchronous Invocation of an AASX File Upload

The upload of an AASX file is only defined as an asynchronous operation in the current version of the specification. The expected behavior is therefore explained in detail. The client sends a POST request to the "/packages-async" endpoint with the AASX file in the request body. The server immediately responds with an Accepted (status code: 202) message containing an OperationHandle in the response body and the link to the status endpoint in the Location response header. This status endpoint is located at "/packages-async/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

a message 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 provided at the "/packages-async/result/{handleId}" endpoint. 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 result object with status code 200 and set the attribute "executionState" to "Failed".

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"](#).

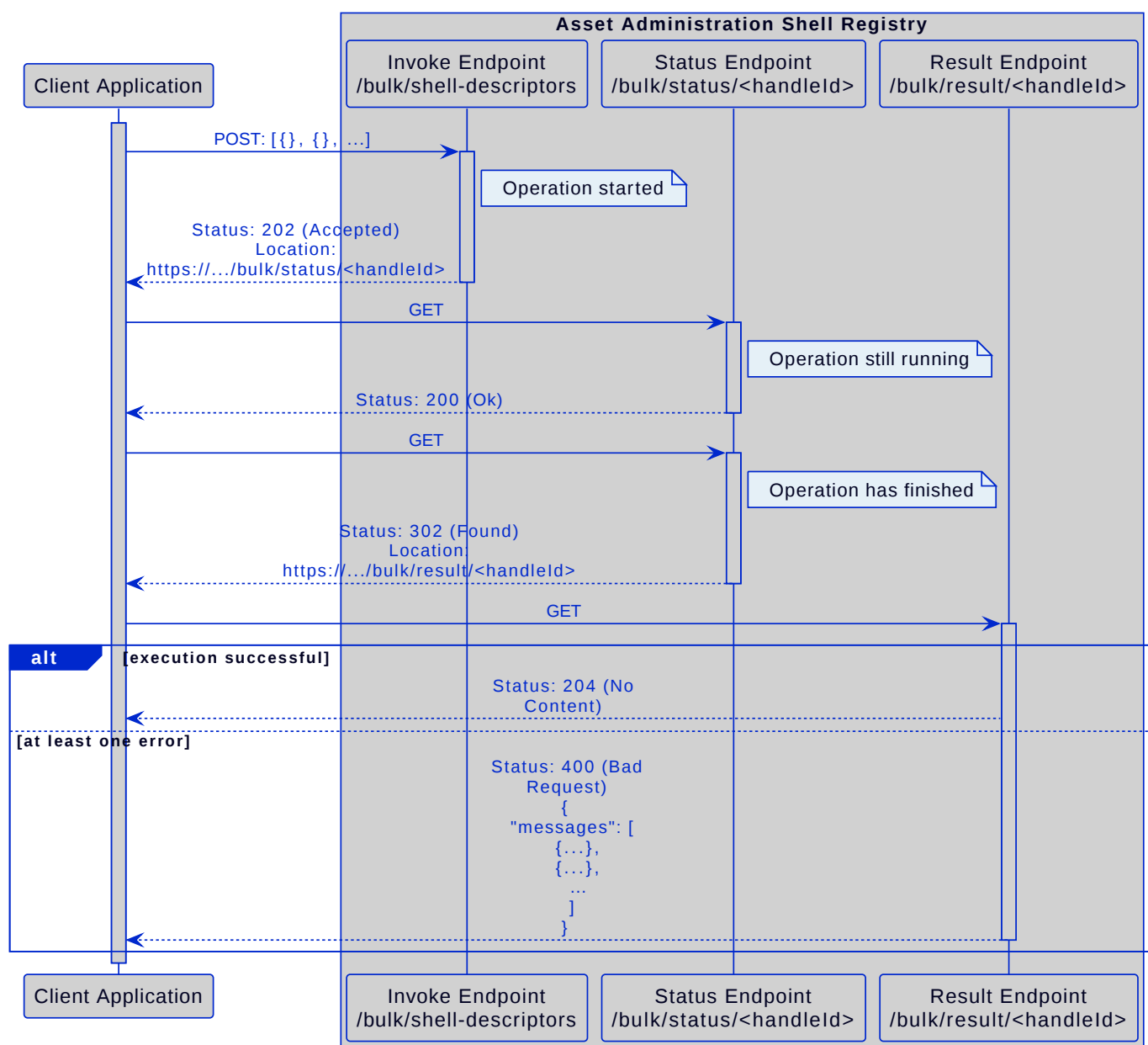


Figure 9. Sequence of a Bulk API Operation

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:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Common JSON Schema for AAS Queries and Access Rules",
  "description": "This schema contains all classes that are shared between the AAS Query Language and the AAS Access Rule Language.",
  "oneOf": [
    {
      "$ref": "#/definitions/Query"
    },
    {
      "$ref": "#/definitions/AllAccessPermissionRules"
    }
  ],
  "definitions": {
    "standardString": {
      "type": "string",
      "minLength": 1
    },
    "ReferenceIdentifier": {
      "type": "string",
      "pattern": "^(?:\\$aas\\(\\\"(?:[\\x09\\x0A\\x0D\\x20-\\x21\\x23-\\x5B\\x5D-\\x5E\\x5F\\x60-\\x6F\\x70-\\x7F\\x80-\\x8F\\x90-\\x9F\\xA0-\\xA9\\xAA-\\xAF\\xB0-\\xBF\\xC0-\\xCF\\xD0-\\xDF\\xE0-\\xEF\\xF0-\\xFF\\x100-\\x10F\\x110-\\x11F\\x120-\\x12F\\x130-\\x13F\\x140-\\x14F\\x150-\\x15F\\x160-\\x16F\\x170-\\x17F\\x180-\\x18F\\x190-\\x19F\\x1A0-\\x1AF\\x1B0-\\x1BF\\x1C0-\\x1CF\\x1D0-\\x1DF\\x1E0-\\x1EF\\x1F0-\\x1FF\\x200-\\x20F\\x210-\\x21F\\x220-\\x22F\\x230-\\x23F\\x240-\\x24F\\x250-\\x25F\\x260-\\x26F\\x270-\\x27F\\x280-\\x28F\\x290-\\x29F\\x2A0-\\x2AF\\x2B0-\\x2BF\\x2C0-\\x2CF\\x2D0-\\x2DF\\x2E0-\\x2EF\\x2F0-\\x2FF\\x300-\\x30F\\x310-\\x31F\\x320-\\x32F\\x330-\\x33F\\x340-\\x34F\\x350-\\x35F\\x360-\\x36F\\x370-\\x37F\\x380-\\x38F\\x390-\\x39F\\x3A0-\\x3AF\\x3B0-\\x3BF\\x3C0-\\x3CF\\x3D0-\\x3DF\\x3E0-\\x3EF\\x3F0-\\x3FF\\x400-\\x40F\\x410-\\x41F\\x420-\\x42F\\x430-\\x43F\\x440-\\x44F\\x450-\\x45F\\x460-\\x46F\\x470-\\x47F\\x480-\\x48F\\x490-\\x49F\\x4A0-\\x4AF\\x4B0-\\x4BF\\x4C0-\\x4CF\\x4D0-\\x4DF\\x4E0-\\x4EF\\x4F0-\\x4FF\\x500-\\x50F\\x510-\\x51F\\x520-\\x52F\\x530-\\x53F\\x540-\\x54F\\x550-\\x55F\\x560-\\x56F\\x570-\\x57F\\x580-\\x58F\\x590-\\x59F\\x5A0-\\x5AF\\x5B0-\\x5BF\\x5C0-\\x5CF\\x5D0-\\x5DF\\x5E0-\\x5EF\\x5F0-\\x5FF\\x600-\\x60F\\x610-\\x61F\\x620-\\x62F\\x630-\\x63F\\x640-\\x64F\\x650-\\x65F\\x660-\\x66F\\x670-\\x67F\\x680-\\x68F\\x690-\\x69F\\x6A0-\\x6AF\\x6B0-\\x6BF\\x6C0-\\x6CF\\x6D0-\\x6DF\\x6E0-\\x6EF\\x6F0-\\x6FF\\x700-\\x70F\\x710-\\x71F\\x720-\\x72F\\x730-\\x73F\\x740-\\x74F\\x750-\\x75F\\x760-\\x76F\\x770-\\x77F\\x780-\\x78F\\x790-\\x79F\\x7A0-\\x7AF\\x7B0-\\x7BF\\x7C0-\\x7CF\\x7D0-\\x7DF\\x7E0-\\x7EF\\x7F0-\\x7FF\\x800-\\x80F\\x810-\\x81F\\x820-\\x82F\\x830-\\x83F\\x840-\\x84F\\x850-\\x85F\\x860-\\x86F\\x870-\\x87F\\x880-\\x88F\\x890-\\x89F\\x8A0-\\x8AF\\x8B0-\\x8BF\\x8C0-\\x8CF\\x8D0-\\x8DF\\x8E0-\\x8EF\\x8F0-\\x8FF\\x900-\\x90F\\x910-\\x91F\\x920-\\x92F\\x930-\\x93F\\x940-\\x94F\\x950-\\x95F\\x960-\\x96F\\x970-\\x97F\\x980-\\x98F\\x990-\\x99F\\x9A0-\\x9AF\\x9B0-\\x9BF\\x9C0-\\x9CF\\x9D0-\\x9DF\\x9E0-\\x9EF\\x9F0-\\x9FF\\xA00-\\xA0F\\xA10-\\xA1F\\xA20-\\xA2F\\xA30-\\xA3F\\xA40-\\xA4F\\xA50-\\xA5F\\xA60-\\xA6F\\xA70-\\xA7F\\xA80-\\xA8F\\xA90-\\xA9F\\xAA0-\\xAAF\\xAB0-\\xABF\\xAC0-\\xACF\\xAD0-\\xADF\\xAE0-\\xAEF\\xAF0-\\xAFF\\xB00-\\xB0F\\xB10-\\xB1F\\xB20-\\xB2F\\xB30-\\xB3F\\xB40-\\xB4F\\xB50-\\xB5F\\xB60-\\xB6F\\xB70-\\xB7F\\xB80-\\xB8F\\xB90-\\xB9F\\xBA0-\\xBAF\\xBB0-\\xBBF\\xBC0-\\xBCF\\xBD0-\\xBDF\\xBE0-\\xBEF\\xBF0-\\xBFF\\xC00-\\xC0F\\xC10-\\xC1F\\xC20-\\xC2F\\xC30-\\xC3F\\xC40-\\xC4F\\xC50-\\xC5F\\xC60-\\xC6F\\xC70-\\xC7F\\xC80-\\xC8F\\xC90-\\xC9F\\xCA0-\\xC AF\\xCB0-\\xC BF\\xCC0-\\xC CF\\xCD0-\\xC DF\\xCE0-\\xC EF\\xCF0-\\xC FF\\xD00-\\xD0F\\xD10-\\xD1F\\xD20-\\xD2F\\xD30-\\xD3F\\xD40-\\xD4F\\xD50-\\xD5F\\xD60-\\xD6F\\xD70-\\xD7F\\xD80-\\xD8F\\xD90-\\xD9F\\xDA0-\\xDAF\\xDB0-\\xDBF\\xDC0-\\xDCF\\xDD0-\\xDDF\\xDE0-\\xDE F\\xDF0-\\xDF F\\xE00-\\xE0F\\xE10-\\xE1F\\xE20-\\xE2F\\xE30-\\xE3F\\xE40-\\xE4F\\xE50-\\xE5F\\xE60-\\xE6F\\xE70-\\xE7F\\xE80-\\xE8F\\xE90-\\xE9F\\xEA0-\\xE AF\\xEB0-\\xEB F\\xEC0-\\xEC F\\xED0-\\xED F\\xEE0-\\xEE F\\xEF0-\\xEF F\\xF00-\\xF0F\\xF10-\\xF1F\\xF20-\\xF2F\\xF30-\\xF3F\\xF40-\\xF4F\\xF50-\\xF5F\\xF60-\\xF6F\\xF70-\\xF7F\\xF80-\\xF8F\\xF90-\\xF9F\\xFA0-\\xFAF\\xFB0-\\xFB F\\xFC0-\\xFC F\\xFD0-\\xFD F\\xFE0-\\xFE F\\xFF0-\\xFF F\\x1000-\\x100F\\x1010-\\x101F\\x1020-\\x102F\\x1030-\\x103F\\x1040-\\x104F\\x1050-\\x105F\\x1060-\\x106F\\x1070-\\x107F\\x1080-\\x108F\\x1090-\\x109F\\x10A0-\\x10AF\\x10B0-\\x10BF\\x10C0-\\x10CF\\x10D0-\\x10DF\\x10E0-\\x10EF\\x10F0-\\x10FF\\x1100-\\x110F\\x1110-\\x111F\\x1120-\\x112F\\x1130-\\x113F\\x1140-\\x114F\\x1150-\\x115F\\x1160-\\x116F\\x1170-\\x117F\\x1180-\\x118F\\x1190-\\x119F\\x11A0-\\x11AF\\x11B0-\\x11BF\\x11C0-\\x11CF\\x11D0-\\x11DF\\x11E0-\\x11EF\\x11F0-\\x11FF\\x1200-\\x120F\\x1210-\\x121F\\x1220-\\x122F\\x1230-\\x123F\\x1240-\\x124F\\x1250-\\x125F\\x1260-\\x126F\\x1270-\\x127F\\x1280-\\x128F\\x1290-\\x129F\\x12A0-\\x12AF\\x12B0-\\x12BF\\x12C0-\\x12CF\\x12D0-\\x12DF\\x12E0-\\x12EF\\x12F0-\\x12FF\\x1300-\\x130F\\x1310-\\x131F\\x1320-\\x132F\\x1330-\\x133F\\x1340-\\x134F\\x1350-\\x135F\\x1360-\\x136F\\x1370-\\x137F\\x1380-\\x138F\\x1390-\\x139F\\x13A0-\\x13AF\\x13B0-\\x13BF\\x13C0-\\x13CF\\x13D0-\\x13DF\\x13E0-\\x13EF\\x13F0-\\x13FF\\x1400-\\x140F\\x1410-\\x141F\\x1420-\\x142F\\x1430-\\x143F\\x1440-\\x144F\\x1450-\\x145F\\x1460-\\x146F\\x1470-\\x147F\\x1480-\\x148F\\x1490-\\x149F\\x14A0-\\x14AF\\x14B0-\\x14BF\\x14C0-\\x14CF\\x14D0-\\x14DF\\x14E0-\\x14EF\\x14F0-\\x14FF\\x1500-\\x150F\\x1510-\\x151F\\x1520-\\x152F\\x1530-\\x153F\\x1540-\\x154F\\x1550-\\x155F\\x1560-\\x156F\\x1570-\\x157F\\x1580-\\x158F\\x1590-\\x159F\\x15A0-\\x15AF\\x15B0-\\x15BF\\x15C0-\\x15CF\\x15D0-\\x15DF\\x15E0-\\x15EF\\x15F0-\\x15FF\\x1600-\\x160F\\x1610-\\x161F\\x1620-\\x162F\\x1630-\\x163F\\x1640-\\x164F\\x1650-\\x165F\\x1660-\\x166F\\x1670-\\x167F\\x1680-\\x168F\\x1690-\\x169F\\x16A0-\\x16AF\\x16B0-\\x16BF\\x16C0-\\x16CF\\x16D0-\\x16DF\\x16E0-\\x16EF\\x16F0-\\x16FF\\x1700-\\x170F\\x1710-\\x171F\\x1720-\\x172F\\x1730-\\x173F\\x1740-\\x1
```

```

\\uFFFF]|\\\\\\\\"\\\\\\")}{1,2048}\\\\"\\")$"
},
"ReferableIdentifier": {
  "type": "string",
  "pattern": "^\\\\$sme\\\\\\\\"(?:[\\\\x09\\\\x0A\\\\x0D\\\\x20-\\\\x21\\\\x23-\\\\x5B\\\\x5D-
\\\\uD7FF\\\\uE000-\\\\uFFFF]|\\\\uDBFF]|\\\\uDC00-\\\\uDFFF]|[^\\\\u0000-
\\\\uFFFF]|\\\\\\\\\\\\"\\\\\\\\\\")}{1,2048}\\\\"\\\\". [A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-9_])?(?:\\\\\\[(?:0|[1-
9][0-9]*)?\\\\\\])*(?:\\\\\\.[A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-9_])?(?:\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\])*)*$"
},
"FragmentIdentifier": {
  "$ref": "#/definitions/FragmentFieldIdentifier"
},
"DescriptorIdentifier": {
  "type": "string",
  "pattern": "^\\\\$(?:aasdesc|smdesc)\\\\\\\\"(?:[\\\\x09\\\\x0A\\\\x0D\\\\x20-\\\\x21\\\\x23-
\\\\x5B\\\\x5D-\\\\uD7FF\\\\uE000-\\\\uFFFF]|\\\\uDBFF]|\\\\uDC00-\\\\uDFFF]|[^\\\\u0000-
\\\\uFFFF]|\\\\\\\\\\\\"\\\\\\\\\\")}{1,2048}\\\\"\\\\"$"
},
"FieldIdentifier": {
  "type": "string",
  "pattern":
"^ (?:\\\\$aas#(?:idShort|id|assetInformation\\\\.assetKind|assetInformation\\\\.assetType|assetIn
formation\\\\.globalAssetId|assetInformation\\\\.specificAssetIds\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:name|value|externalSubjectId(?:\\\\. (?:type|keys\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:type|value)))?)|submodels\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\] (?:\\\\. (?:type|keys\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:type|value)))?)|\\\\$sm#(?:semanticId(?:\\\\. (?:type|keys\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:type|value)))?)|supplementalSemanticIds(?:\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\])?(?:\\\\. (?:type|keys\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:type|value)))?)|idShort|id)|\\\\$sme(?:\\\\. [A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-
9_])?(?:\\\\\\[(?:0|[1-9][0-9]*)?\\\\\\])*)(?:\\\\\\.[A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-
9_])?(?:\\\\\\[(?:0|[1-9][0-9]*)?\\\\\\])*)*)?#(?:semanticId(?:\\\\. (?:type|keys\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:type|value)))?)|supplementalSemanticIds(?:\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\])?(?:\\\\. (?:type|keys\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:type|value)))?)|idShort|value|valueType|language)|\\\\$cd#(?:idShort|id)|\\\\$aasd
esc#(?:idShort|id|assetKind|assetType|globalAssetId|specificAssetIds\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:name|value|externalSubjectId(?:\\\\. (?:type|keys\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:type|value)))?)|endpoints\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:interface|protocolInformation\\\\.href)|submodelDescriptors\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:semanticId(?:\\\\. (?:type|keys\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:type|value)))?)|supplementalSemanticIds(?:\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\])?(?:\\\\. (?:type|keys\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:type|value)))?)|idShort|id|endpoints\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:interface|protocolInformation\\\\.href)))|\\\\$smdesc#(?:semanticId(?:\\\\. (?:type|
keys\\\\\\[(?:0|[1-9][0-9]*)?\\\\\\]\\\\. (?:type|value)))?)|supplementalSemanticIds(?:\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\])?(?:\\\\. (?:type|keys\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:type|value)))?)|idShort|id|endpoints\\\\\\[(?:0|[1-9][0-
9]*)?\\\\\\]\\\\. (?:interface|protocolInformation\\\\.href)))$"
},
"FragmentFieldIdentifier": {
  "type": "string",

```

```

    "pattern":
    "^(?:\\$aas#(?:idShort|assetInformation\\.assetType|assetInformation\\.globalAssetId|assetI
nformation\\.specificAssetIds(?:\\[(?:0|[1-9])[0-
9]*)?\\)(?:\\.externalSubjectId(?:\\.keys(?:\\[(?:0|[1-9])[0-
9]*)?\\))?)?)?|submodels(?:\\[(?:0|[1-9])[0-9]*)?\\)(?:\\.keys(?:\\[(?:0|[1-9])[0-
9]*)?\\))?)?)|\\$sm#(?:semanticId(?:\\.keys(?:\\[(?:0|[1-9])[0-
9]*)?\\))?)?|supplementalSemanticIds(?:\\[(?:0|[1-9])[0-9]*)?\\)(?:\\.keys(?:\\[(?:0|[1-
9])[0-9]*)?\\))?)?|idShort)|\\$sme(?:\\.([A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-
9_])?(?:\\[(?:0|[1-9])[0-9]*)?\\))*?(?:\\.([A-Za-z](?:[A-Za-z0-9_-]*[A-Za-z0-
9_])?(?:\\[(?:0|[1-9])[0-9]*)?\\))*?)?(?:#(?:semanticId(?:\\.keys(?:\\[(?:0|[1-9])[0-
9]*)?\\))?)?|supplementalSemanticIds(?:\\[(?:0|[1-9])[0-9]*)?\\)(?:\\.keys(?:\\[(?:0|[1-
9])[0-
9]*)?\\))?)?)?|idShort|value|valueType|language))?)|\\$cd#idShort|\\$aasdesc#(?:idShort|desc
ription|displayName|extension|administration|assetKind|assetType|globalAssetId|specificAsse
tIds(?:\\[(?:0|[1-9])[0-9]*)?\\)(?:\\.externalSubjectId(?:\\.keys(?:\\[(?:0|[1-9])[0-
9]*)?\\))?)?)?|endpoints(?:\\[(?:0|[1-9])[0-9]*)?\\])?|submodelDescriptors(?:\\[(?:0|[1-
9])[0-9]*)?\\)(?:\\.semanticId(?:\\.keys(?:\\[(?:0|[1-9])[0-
9]*)?\\))?)?|supplementalSemanticIds(?:\\[(?:0|[1-9])[0-9]*)?\\)(?:\\.keys(?:\\[(?:0|[1-
9])[0-9]*)?\\))?)?)?|idShort|endpoints(?:\\[(?:0|[1-9])[0-
9]*)?\\))?)?)?)|\\$smdesc#(?:semanticId(?:\\.keys(?:\\[(?:0|[1-9])[0-
9]*)?\\))?)?|supplementalSemanticIds(?:\\[(?:0|[1-9])[0-9]*)?\\)(?:\\.keys(?:\\[(?:0|[1-
9])[0-9]*)?\\))?)?)?|idShort|endpoints(?:\\[(?:0|[1-9])[0-9]*)?\\))?)?)$"
    },
    "hexLiteralPattern": {
        "type": "string",
        "pattern": "^16#[0-9A-F]+$"
    },
    "dateTimeLiteralPattern": {
        "type": "string",
        "pattern": "^(?:(?:[0-9]{4}-(?:(?:0[13578]|1[02])-(?:0[1-9]|1[12])[0-
9]|3[01])|(?:0[469]|11)-(?:0[1-9]|1[12])[0-9]|30|02-(?:0[1-9]|1[0-9]|2[0-8]))|(?:(?:[0-
9]{2}(?:0[48]|2[468][048]|1[3579][26])|(?:0[2468][048]|1[3579][26])00)-02-
29))T(?:(?:[01][0-9]|2[0-3]):[0-5][0-9](?::[0-5][0-9])?(?:\\.([0-
9]+)?|24:00(?::00)?(?:\\.0+)?(?:Z|[-+](?:0[0-9]|1[0-3]):[0-5][0-9]|14:00)))?$"
    },
    "timeLiteralPattern": {
        "type": "string",
        "pattern": "^(?:(?:[01][0-9]|2[0-3]):[0-5][0-9](?::[0-5][0-9])?(?:\\.([0-
9]+)?|24:00(?::00)?(?:\\.0+)?(?:Z|[-+](?:0[0-9]|1[0-3]):[0-5][0-9]|14:00)))?$"
    },
    "dateTimeAttributeItem": {
        "type": "object",
        "properties": {
            "GLOBAL": {
                "type": "string",
                "enum": [
                    "LOCALNOW",
                    "UTCNOW",
                    "CLIENTNOW"
                ]
            }
        }
    }
}

```

```

    "required": [
      "GLOBAL"
    ],
    "additionalProperties": false
  },
  "dateTimeOperand": {
    "type": "object",
    "properties": {
      "$dateTimeVal": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      },
      "$dateTimeCast": {
        "$ref": "#/definitions/stringValue"
      },
      "$attribute": {
        "$ref": "#/definitions/dateTimeAttributeItem"
      }
    },
    "oneOf": [
      {
        "required": [
          "$dateTimeVal"
        ]
      },
      {
        "required": [
          "$dateTimeCast"
        ]
      },
      {
        "required": [
          "$attribute"
        ]
      }
    ],
    "additionalProperties": false
  },
  "fieldOperand": {
    "type": "object",
    "properties": {
      "$field": {
        "$ref": "#/definitions/FieldIdentifier"
      }
    },
    "required": [
      "$field"
    ],
    "additionalProperties": false
  },
  "numericalOperand": {
    "type": "object",
    "properties": {

```

```

"$numVal": {
  "type": "number"
},
"$numCast": {
  "$ref": "#/definitions/Value"
},
"$dayOfWeek": {
  "$ref": "#/definitions/dateTimeOperand"
},
"$dayOfMonth": {
  "$ref": "#/definitions/dateTimeOperand"
},
"$month": {
  "$ref": "#/definitions/dateTimeOperand"
},
"$year": {
  "$ref": "#/definitions/dateTimeOperand"
}
},
"oneOf": [
  {
    "required": [
      "$numVal"
    ]
  },
  {
    "required": [
      "$numCast"
    ]
  },
  {
    "required": [
      "$dayOfWeek"
    ]
  },
  {
    "required": [
      "$dayOfMonth"
    ]
  },
  {
    "required": [
      "$month"
    ]
  },
  {
    "required": [
      "$year"
    ]
  }
],
"additionalProperties": false

```

```

},
"hexOperand": {
  "type": "object",
  "properties": {
    "$hexVal": {
      "$ref": "#/definitions/hexLiteralPattern"
    },
    "$hexCast": {
      "$ref": "#/definitions/Value"
    }
  }
},
"oneOf": [
  {
    "required": [
      "$hexVal"
    ]
  },
  {
    "required": [
      "$hexCast"
    ]
  }
],
"additionalProperties": false
},
"boolOperand": {
  "type": "object",
  "properties": {
    "$boolean": {
      "type": "boolean"
    },
    "$boolCast": {
      "$ref": "#/definitions/Value"
    }
  }
},
"oneOf": [
  {
    "required": [
      "$boolean"
    ]
  },
  {
    "required": [
      "$boolCast"
    ]
  }
],
"additionalProperties": false
},
"timeOperand": {
  "type": "object",
  "properties": {

```

```

"$timeVal": {
  "$ref": "#/definitions/timeLiteralPattern"
},
"$timeCast": {
  "anyOf": [
    {
      "$ref": "#/definitions/stringValue"
    },
    {
      "$ref": "#/definitions/dateTimeOperand"
    }
  ]
},
"oneOf": [
  {
    "required": [
      "$timeVal"
    ]
  },
  {
    "required": [
      "$timeCast"
    ]
  }
],
"additionalProperties": false
},
"Value": {
  "type": "object",
  "properties": {
    "$field": {
      "$ref": "#/definitions/FieldIdentifier"
    },
    "$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/stringValue"
},
"$timeCast": {
  "anyOf": [
    {
      "$ref": "#/definitions/stringValue"
    },
    {
      "$ref": "#/definitions/dateTimeOperand"
    }
  ]
},
"$dayOfWeek": {
  "$ref": "#/definitions/dateTimeOperand"
},
"$dayOfMonth": {
  "$ref": "#/definitions/dateTimeOperand"
},
"$month": {
  "$ref": "#/definitions/dateTimeOperand"
},
"$year": {
  "$ref": "#/definitions/dateTimeOperand"
}
},
"oneOf": [
  {
    "required": [
      "$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/FieldIdentifier"
        },
        "$strVal": {
            "$ref": "#/definitions/standardString"
        },
        "$strCast": {
            "$ref": "#/definitions/Value"
        },
        "$attribute": {
            "$ref": "#/definitions/attributeItem"
        }
    }
},
"oneOf": [
    {
        "required": [
            "$field"
        ]
    }
]

```

```

    },
    {
      "required": [
        "$strVal"
      ]
    },
    {
      "required": [
        "$strCast"
      ]
    },
    {
      "required": [
        "$attribute"
      ]
    }
  ],
  "additionalProperties": false
},
"stringNonFieldValue": {
  "type": "object",
  "properties": {
    "$strVal": {
      "$ref": "#/definitions/standardString"
    },
    "$strCast": {
      "$ref": "#/definitions/Value"
    },
    "$attribute": {
      "$ref": "#/definitions/attributeItem"
    }
  },
  "oneOf": [
    {
      "required": [
        "$strVal"
      ]
    },
    {
      "required": [
        "$strCast"
      ]
    },
    {
      "required": [
        "$attribute"
      ]
    }
  ],
  "additionalProperties": false
},
"comparisonItems": {

```

```

    "$ref": "#/definitions/equalityComparisonItems"
  },
  "stringItems": {
    "anyOf": [
      {
        "type": "array",
        "minItems": 2,
        "maxItems": 2,
        "items": [
          {
            "$ref": "#/definitions/stringValue"
          },
          {
            "$ref": "#/definitions/stringNonFieldValue"
          }
        ]
      },
      {
        "type": "array",
        "minItems": 2,
        "maxItems": 2,
        "items": [
          {
            "$ref": "#/definitions/stringNonFieldValue"
          },
          {
            "$ref": "#/definitions/stringValue"
          }
        ]
      },
      {
        "type": "array",
        "minItems": 2,
        "maxItems": 2,
        "items": [
          {
            "$ref": "#/definitions/fieldOperand"
          },
          {
            "$ref": "#/definitions/fieldOperand"
          }
        ]
      }
    ]
  },
  "numericalComparisonItems": {
    "oneOf": [
      {
        "type": "array",
        "minItems": 2,
        "maxItems": 2,
        "items": {

```

```

        "$ref": "#/definitions/numerical0operand"
    }
},
{
    "type": "array",
    "minItems": 2,
    "maxItems": 2,
    "items": [
        {
            "$ref": "#/definitions/numerical0operand"
        },
        {
            "$ref": "#/definitions/field0operand"
        }
    ]
},
{
    "type": "array",
    "minItems": 2,
    "maxItems": 2,
    "items": [
        {
            "$ref": "#/definitions/field0operand"
        },
        {
            "$ref": "#/definitions/numerical0operand"
        }
    ]
}
],
},
"hexComparisonItems": {
    "oneOf": [
        {
            "type": "array",
            "minItems": 2,
            "maxItems": 2,
            "items": {
                "$ref": "#/definitions/hex0operand"
            }
        },
        {
            "type": "array",
            "minItems": 2,
            "maxItems": 2,
            "items": [
                {
                    "$ref": "#/definitions/hex0operand"
                },
                {
                    "$ref": "#/definitions/field0operand"
                }
            ]
        }
    ]
}

```

```

    ]
  },
  {
    "type": "array",
    "minItems": 2,
    "maxItems": 2,
    "items": [
      {
        "$ref": "#/definitions/fieldOperand"
      },
      {
        "$ref": "#/definitions/hexOperand"
      }
    ]
  }
]
},
"boolComparisonItems": {
  "oneOf": [
    {
      "type": "array",
      "minItems": 2,
      "maxItems": 2,
      "items": {
        "$ref": "#/definitions/boolOperand"
      }
    },
    {
      "type": "array",
      "minItems": 2,
      "maxItems": 2,
      "items": [
        {
          "$ref": "#/definitions/boolOperand"
        },
        {
          "$ref": "#/definitions/fieldOperand"
        }
      ]
    },
    {
      "type": "array",
      "minItems": 2,
      "maxItems": 2,
      "items": [
        {
          "$ref": "#/definitions/fieldOperand"
        },
        {
          "$ref": "#/definitions/boolOperand"
        }
      ]
    }
  ]
}

```

```

    }
  ]
},
"dateTimeComparisonItems": {
  "oneOf": [
    {
      "type": "array",
      "minItems": 2,
      "maxItems": 2,
      "items": {
        "$ref": "#/definitions/dateTimeOperand"
      }
    },
    {
      "type": "array",
      "minItems": 2,
      "maxItems": 2,
      "items": [
        {
          "$ref": "#/definitions/dateTimeOperand"
        },
        {
          "$ref": "#/definitions/fieldOperand"
        }
      ]
    },
    {
      "type": "array",
      "minItems": 2,
      "maxItems": 2,
      "items": [
        {
          "$ref": "#/definitions/fieldOperand"
        },
        {
          "$ref": "#/definitions/dateTimeOperand"
        }
      ]
    }
  ]
},
"timeComparisonItems": {
  "oneOf": [
    {
      "type": "array",
      "minItems": 2,
      "maxItems": 2,
      "items": {
        "$ref": "#/definitions/timeOperand"
      }
    },
    {

```

```

        "type": "array",
        "minItems": 2,
        "maxItems": 2,
        "items": [
            {
                "$ref": "#/definitions/timeOperand"
            },
            {
                "$ref": "#/definitions/fieldOperand"
            }
        ]
    },
    {
        "type": "array",
        "minItems": 2,
        "maxItems": 2,
        "items": [
            {
                "$ref": "#/definitions/fieldOperand"
            },
            {
                "$ref": "#/definitions/timeOperand"
            }
        ]
    }
]
},
"equalityComparisonItems": {
    "anyOf": [
        {
            "$ref": "#/definitions/stringItems"
        },
        {
            "$ref": "#/definitions/numericalComparisonItems"
        },
        {
            "$ref": "#/definitions/hexComparisonItems"
        },
        {
            "$ref": "#/definitions/boolComparisonItems"
        },
        {
            "$ref": "#/definitions/dateTimeComparisonItems"
        },
        {
            "$ref": "#/definitions/timeComparisonItems"
        }
    ]
},
"orderedComparisonItems": {
    "anyOf": [
        {

```

```

    "$ref": "#/definitions/stringItems"
  },
  {
    "$ref": "#/definitions/numericalComparisonItems"
  },
  {
    "$ref": "#/definitions/hexComparisonItems"
  },
  {
    "$ref": "#/definitions/dateTimeComparisonItems"
  },
  {
    "$ref": "#/definitions/timeComparisonItems"
  }
]
},
"matchExpression": {
  "type": "object",
  "properties": {
    "$match": {
      "type": "array",
      "minItems": 1,
      "items": {
        "$ref": "#/definitions/matchExpression"
      }
    },
    "$eq": {
      "$ref": "#/definitions/equalityComparisonItems"
    },
    "$ne": {
      "$ref": "#/definitions/equalityComparisonItems"
    },
    "$gt": {
      "$ref": "#/definitions/orderedComparisonItems"
    },
    "$ge": {
      "$ref": "#/definitions/orderedComparisonItems"
    },
    "$lt": {
      "$ref": "#/definitions/orderedComparisonItems"
    },
    "$le": {
      "$ref": "#/definitions/orderedComparisonItems"
    },
    "$contains": {
      "$ref": "#/definitions/stringItems"
    },
    "$starts-with": {
      "$ref": "#/definitions/stringItems"
    },
    "$ends-with": {
      "$ref": "#/definitions/stringItems"
    }
  }
}

```

```

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

```

```

        "required": [
            "$regex"
        ],
    },
    {
        "required": [
            "$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/equalityComparisonItems"
        },
        "$ne": {
            "$ref": "#/definitions/equalityComparisonItems"
        },
        "$gt": {
            "$ref": "#/definitions/orderedComparisonItems"
        },
        "$ge": {
            "$ref": "#/definitions/orderedComparisonItems"
        },
        "$lt": {

```

```

    "$ref": "#/definitions/orderedComparisonItems"
  },
  "$le": {
    "$ref": "#/definitions/orderedComparisonItems"
  },
  "$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": {
        "$ref": "#/definitions/ReferenceIdentifier"
    }
},
"additionalProperties": false
},
"objectItem": {
    "oneOf": [
        {
            "required": [
                "ROUTE"
            ]
        },
        {
            "required": [
                "IDENTIFIABLE"
            ]
        },
        {
            "required": [
                "REFERABLE"
            ]
        }
    ],
    {
        "required": [

```

```

        "FRAGMENT"
    ]
},
{
    "required": [
        "DESCRIPTOR"
    ]
}
],
"properties": {
    "ROUTE": {
        "type": "string"
    },
    "IDENTIFIABLE": {
        "$ref": "#/definitions/IdentifiableIdentifier"
    },
    "REFERABLE": {
        "$ref": "#/definitions/ReferableIdentifier"
    },
    "FRAGMENT": {
        "$ref": "#/definitions/FragmentIdentifier"
    },
    "DESCRIPTOR": {
        "$ref": "#/definitions/DescriptorIdentifier"
    }
},
"additionalProperties": false
},
"rightsEnum": {
    "type": "string",
    "enum": [
        "CREATE",
        "READ",
        "UPDATE",
        "DELETE",
        "EXECUTE",
        "VIEW",
        "ALL"
    ],
    "additionalProperties": false
},
"ACL": {
    "type": "object",
    "properties": {
        "ATTRIBUTES": {
            "type": "array",
            "items": {
                "$ref": "#/definitions/attributeItem"
            }
        },
        "USEATTRIBUTES": {
            "type": "array",

```

```

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

```

```

"USEOBJECTS": {
  "type": "array",
  "items": {
    "type": "string"
  }
},
"FORMULA": {
  "$ref": "#/definitions/logicalExpression",
  "additionalProperties": false
},
"USEFORMULA": {
  "type": "string"
},
"FILTER": {
  "$ref": "#/definitions/SecurityQueryFilter",
  "additionalProperties": false
},
"FILTERLIST": {
  "type": "array",
  "items": {
    "$ref": "#/definitions/SecurityQueryFilter"
  }
}
},
"allOf": [
  {
    "oneOf": [
      {
        "required": [
          "ACL"
        ]
      },
      {
        "required": [
          "USEACL"
        ]
      }
    ]
  },
  {
    "anyOf": [
      {
        "required": [
          "OBJECTS"
        ]
      },
      {
        "required": [
          "USEOBJECTS"
        ]
      }
    ]
  }
]

```

```

    },
    {
      "oneOf": [
        {
          "required": [
            "FORMULA"
          ]
        },
        {
          "required": [
            "USEFORMULA"
          ]
        }
      ]
    }
  ],
  "additionalProperties": false
},
"SecurityQueryFilter": {
  "type": "object",
  "properties": {
    "FRAGMENT": {
      "$ref": "#/definitions/FragmentFieldIdentifier"
    },
    "CONDITION": {
      "$ref": "#/definitions/logicalExpression"
    },
    "USEFORMULA": {
      "type": "string"
    }
  },
  "required": [
    "FRAGMENT"
  ],
  "oneOf": [
    {
      "required": [
        "CONDITION"
      ]
    },
    {
      "required": [
        "USEFORMULA"
      ]
    }
  ],
  "additionalProperties": false
},
"QueryFilter": {
  "type": "object",
  "properties": {
    "$fragment": {

```

```

        "$ref": "#/definitions/FragmentFieldIdentifier"
    },
    "$condition": {
        "$ref": "#/definitions/logicalExpression"
    }
},
"required": [
    "$fragment",
    "$condition"
],
"additionalProperties": false
},
"Query": {
    "type": "object",
    "properties": {
        "$select": {
            "type": "string",
            "pattern": "^id$"
        },
        "$condition": {
            "$ref": "#/definitions/logicalExpression"
        },
        "$filters": {
            "type": "array",
            "items": {
                "$ref": "#/definitions/QueryFilter"
            }
        }
    },
    "required": [
        "$condition"
    ],
    "additionalProperties": false
},
"AllAccessPermissionRules": {
    "type": "object",
    "properties": {
        "DEFATTRIBUTES": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "name": {
                        "type": "string"
                    }
                },
                "attributes": {
                    "type": "array",
                    "items": {
                        "$ref": "#/definitions/attributeItem"
                    }
                }
            }
        },
        "USEATTRIBUTES": {

```

```

        "type": "array",
        "items": {
            "type": "string"
        }
    },
    "required": [
        "name"
    ],
    "oneOf": [
        {
            "required": [
                "attributes"
            ]
        },
        {
            "required": [
                "USEATTRIBUTES"
            ]
        }
    ],
    "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 15. 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. <code>SubmodelElement</code>
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

	Meaning	HTTP status code	Explanation
ClientForbidden	Authorization has been refused	403 (Forbidden)	The request content is basically valid and understood by the server, but the server refuses the action due to certain restrictions, e.g. profiles or roles
ClientErrorResourceNotFound	Resource not found	404 (Not Found)	The requested resource was not found
ClientMethodNotAllowed	Operation request is not allowed	405 (Method Not Allowed)	The server rejected the request for the requested resource, e.g. /invoke only for the operation submodel element
ClientResourceConflict	Conflict-creating resource (resource already exists)	409 (Conflict)	A resource already exists; might occur if a Submodel or SubmodelElement with the same Identifier or ShortId is contained in a POST request.
ServerInternalError	Unexpected error	500 (Internal Server Error)	General server-internal error due to an unexpected condition
ServerNotImplemented	Not implemented	501 (Not Implemented)	The server does not support the functionality to fulfill the request
ServerErrorBadGateway	Bad Gateway	502 (Bad Gateway)	The primarily addressed server that was acting as gateway or proxy received an invalid response from subsequent systems/servers

Additional Data Types for Payload for HTTP/REST

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

PackageDescription

Class Name	<i>PackageDescription</i>		
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.
<i>packageId</i>	File server specific package id	ShortIdType	1
<i>aasId</i>	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/V3.2_SSP-002” contains the API Operation “GetAllSubmodels” but not “[PostSubmodelElementByPath](#) +”, while the more comprehensive “RepositoryServiceSpecification/V3.2_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 16. 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

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
Discovery Service Specification									X		X
Asset Administration Shell Repository Serv. Spec.	S	S				X	S			X	X
Submodel Repository Service Specification		S					X			X	X
ConceptDescription Repository Service Spec.								X		X	X

x: Service Specification contains API at the root

s: Service Specification contains API through [superpaths](#)

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/3/1/SubmodelRegistryServiceSpecification/SSP-002" and "https://admin-shell.io/aas/API/3/2/SubmodelRegistryServiceSpecification/SSP-002" are supported, the server may only include "https://admin-shell.io/aas/API/3/2/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/3/2/AssetAdministrationShellServiceSpecification/SSP-001".

FieldIdentifier Applicability per Profile

The AAS Query Language (see [Query Language](#)) and the Security Access Rule Model (see the Security specification, IDTA-01004) use a set of FieldIdentifier prefixes (\$aas, \$sm, \$sme, \$cd, \$aasdesc, \$smdesc). Not every prefix is meaningful in every profile, because the underlying data is only exposed by specific service specifications.

The following table summarises which FieldIdentifier prefix is applicable per service specification family:

Service Specification (family)	\$aas	\$sm	\$sme	\$cd	\$aasdesc	\$smdesc
AssetAdministrationShellRepositoryServiceSpecification	↯	↯	↯	—	—	—
SubmodelRepositoryServiceSpecification	—	↯	↯	—	—	—
ConceptDescriptionRepositoryServiceSpecification	—	—	—	↯	—	—
AssetAdministrationShellRegistryServiceSpecification	—	—	—	—	↯	↯
SubmodelRegistryServiceSpecification	—	—	—	—	—	↯
AssetAdministrationShellServiceSpecification	↯	↯	↯	—	—	—
SubmodelServiceSpecification	—	↯	↯	—	—	—
DiscoveryServiceSpecification	—	—	—	—	—	—

Queries and access rules that reference a prefix marked with "—" for the chosen profile are treated as **not applicable** in that profile: they MUST NOT cause a parse or evaluation error, they simply do not match any object in the deployment. Implementations SHOULD clearly indicate in their deployment documentation which profiles they expose so that operators can scope their rules accordingly.

Applicability describes which data types may be evaluated by access rules; it does not imply that every profile in a service specification family exposes the Query API.

This table is referenced by IDTA-01004 § "Descriptor FieldIdentifier Applicability".

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/3/2/AssetAdministrationShellServiceSpecification/SSP-001
Feature	Appearance

Name:	AAS Full Profile
APIs and API Operations	<i>Asset Administration Shell API:</i> GetAssetAdministrationShell PutAssetAdministrationShell GetAllSubmodelReferences PostSubmodelReference DeleteSubmodelReference GetAssetInformation PutAssetInformation GetThumbnail PutThumbnail DeleteThumbnail <i>Submodel API as superpath:</i> GetSubmodelById GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath PutFileByPath DeleteFileByPath PutSubmodelById PatchSubmodelById PostSubmodelElement PostSubmodelElementByPath PostSubmodelElementByPath PutSubmodelElementByPath PutSubmodelElementByPath PatchSubmodelElementByPath DeleteSubmodelElementByPath InvokeOperationSync InvokeOperationAsync GetOperationAsyncStatus GetOperationAsyncResult <i>Serialization API:</i> GenerateSerializationByIds <i>Description API:</i> GetDescription
SerializationModifier	Level: Core, Deep Content: Normal, Metadata, Value, Reference, Path Extent: WithBLOBValue, WithoutBLOBValue
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AssetAdministrationShellServiceSpecification/V3.2_SSP-001.yaml&version=v3.2.0

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/3/2/AssetAdministrationShellServiceSpecification/SSP-002
Feature	Appearance
API Operations	<i>Asset Administration Shell API:</i> GetAssetAdministrationShell GetAllSubmodelReferences GetAssetInformation GetThumbnail <i>Submodel API as superpath:</i> GetSubmodelById GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath <i>Description API:</i> GetDescription
SerializationModifier	Level: Core, Deep Content: Normal, Metadata, Value, Reference, Path Extent: WithBLOBValue, WithoutBLOBValue
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AssetAdministrationShellServiceSpecification/V3.2_SSP-002.yaml&version=v3.2.0

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/3/2/SubmodelServiceSpecification/SSP-001
Feature	Appearance
APIs and API Operations	<i>Submodel API:</i> GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath PutFileByPath DeleteFileByPath PutSubmodel PatchSubmodel PostSubmodelElement PostSubmodelElementByPath PutSubmodelElementByPath PatchSubmodelElementByPath PatchSubmodelElementByPath InvokeOperationSync InvokeOperationAsync GetOperationAsyncStatus GetOperationAsyncResult <i>Serialization API:</i> GenerateSerializationByIds <i>Description API:</i> GetDescription
SerializationModifier	Level: Core, Deep Content: Normal, Metadata, Value, Reference, Path Extent: WithBLOBValue, WithoutBLOBValue
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../SubmodelServiceSpecification/V3.2_SSP-001.yaml&version=v3.2.0

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/3/2/SubmodelServiceSpecification/SSP-002
Feature	Appearance

Name:	Submodel Read Profile
API and API Operations	<i>Submodel API:</i> GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath <i>Serialization API:</i> GenerateSerializationByIds <i>Description API:</i> GetDescription
SerializationModifier	Level: Core, Deep Content: Normal, Metadata, Value, Reference, Path Extent: WithBLOBValue, WithoutBLOBValue
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../SubmodelServiceSpecification/V3.2_SSP-002.yaml&version=v3.2.0

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/3/2/SubmodelServiceSpecification/SSP-003
Feature	Appearance
APIs and API Operations	<i>Submodel API:</i> GetSubmodel InvokeOperationSync <i>Description API:</i> GetDescription
SerializationModifier	Level: Deep Content: Normal, Value Extent: WithBLOBValue, WithoutBLOBValue
SerializationFormat	JSON
Pagination	not supported

See: <https://industrialdigitaltwin.io/aas-specs-api/docs/swagger->

AASX File Server Service Specification

Service Specification / Profiles	Description
AasxFileServerServiceSpecification/SSP-001 The full feature set of the AASX File Server Service Specification	AasxFileServerServiceSpecification/SSP-002 The async feature set of the AASX File Server Service Specification

AASX File Server Service Specification – Full Profile

Name:	AASX File Server Full Profile
Profile Identifier	https://admin-shell.io/aas/API/3/2/AasxFileServerServiceSpecification/SSP-001
Feature	Appearance
APIs and API Operations	<i>File Server API:</i> GetAllAASXPackageIds GetAASXByPackageId PostAASXPackage PutAASXByPackageId DeleteAASXByPackageId <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON for descriptions and error messages AASX for packages
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AasxFileServerServiceSpecification/V3.2_SSP-001.yaml&version=v3.2.0

AASX File Server Service Specification – Async Profile

Name:	AASX File Server Async Profile
Profile Identifier	https://admin-shell.io/aas/API/3/2/AasxFileServerServiceSpecification/SSP-002
Feature	Appearance
APIs and API Operations	<i>File Server API:</i> InvokeOperationAsync GetAasxAsyncStatus GetAasxAsyncResult <i>Description API:</i> GetDescription

Name:	AASX File Server Async Profile
SerializationModifier	not applicable
SerializationFormat	JSON for descriptions and error messages AASX for packages
Pagination	not supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AasxFileServerServiceSpecification/V3.2_SSP-002.yaml&version=v3.2.0

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/3/2/AssetAdministrationShellRegistryServiceSpecification/SSP-001
Feature	Appearance

Name:	Asset Administration Shell Registry Full Profile
APIs and API Operations	<i>AAS Registry API:</i> GetAllAssetAdministrationShellDescriptors GetAssetAdministrationShellDescriptorById PostAssetAdministrationShellDescriptor PutAssetAdministrationShellDescriptorById DeleteAssetAdministrationShellDescriptorById <i>Submodel Registry API as superpath:</i> GetAllSubmodelDescriptors GetSubmodelDescriptorById PostSubmodelDescriptor PutSubmodelDescriptorById DeleteSubmodelDescriptorById <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	Supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AssetAdministrationShellRegistryServiceSpecification/V3.2_SSP-001.yaml&version=v3.2.0

Asset Administration Shell Registry Service Specification – Read Profile

Name:	AAS Registry Read Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRegistryServiceSpecification/SSP-002
Feature	Appearance
APIs and API Operations	<i>AAS Registry API:</i> GetAllAssetAdministrationShellDescriptors GetAssetAdministrationShellDescriptorById <i>Submodel Registry API as superpath:</i> GetAllSubmodelDescriptors GetDescription <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AssetAdministrationShellRegistryServiceSpecification/V3.2_SSP-002.yaml&version=v3.2.0

Asset Administration Shell Registry Service Specification – Bulk Profile

Name:	AAS Registry Bulk Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRegistryServiceSpecification/SSP-003
Feature	Appearance
APIs and API Operations	<i>AAS Registry API:</i> CreateBulkAssetAdministrationShellDescriptors PutBulkAssetAdministrationShellDescriptorsById DeleteBulkAssetAdministrationShellDescriptorsById <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	Not supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AssetAdministrationShellRegistryServiceSpecification/V3.2_SSP-003.yaml&version=v3.2.0

Asset Administration Shell Registry Service Specification – Query Profile

Name:	AAS Registry Bulk Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRegistryServiceSpecification/SSP-004
Feature	Appearance
APIs and API Operations	<i>AAS Registry API:</i> QueryAssetAdministrationShellDescriptors <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AssetAdministrationShellRegistryServiceSpecification/V3.2_SSP-004.yaml&version=v3.2.0

Asset Administration Shell Registry Service Specification – Minimal Read Profile

Name:	AAS Registry Minimal Read Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRegistryServiceSpecification/SSP-005
Feature	Appearance
APIs and API Operations	<i>AAS Registry API:</i> GetAllAssetAdministrationShellDescriptors GetAssetAdministrationShellDescriptorById <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AssetAdministrationShellRegistryServiceSpecification/V3.2_SSP-005.yaml&version=v3.2.0

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/3/2/SubmodelRegistryServiceSpecification/SSP-001
Feature	Appearance

Name:	Submodel Registry Full Profile
APIs and API Operations	<i>Submodel Registry API:</i> GetAllSubmodelDescriptors GetDescription PostSubmodelDescriptor PutSubmodelDescriptorById DeleteSubmodelDescriptorById <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../SubmodelRegistryServiceSpecification/V3.2_SSP-001.yaml&version=v3.2.0

Submodel Registry Profile – Read Profile

Name:	Submodel Registry Read Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/SubmodelRegistryServiceSpecification/SSP-002
Feature	Appearance
APIs and API Operations	<i>Submodel Registry API:</i> GetAllSubmodelDescriptors GetSubmodelDescriptorById <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	Supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../SubmodelRegistryServiceSpecification/V3.2_SSP-002.yaml&version=v3.2.0

Submodel Registry Profile – Bulk Profile

Name:	Submodel Registry Bulk Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/SubmodelRegistryServiceSpecification/SSP-003
Feature	Appearance

Name:	Submodel Registry Bulk Profile
APIs and API Operations	<i>Submodel Registry API:</i> PostBulkSubmodelDescriptors PutBulkSubmodelDescriptorsById DeleteBulkSubmodelDescriptorsById <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	Not Supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../SubmodelRegistryServiceSpecification/V3.2_SSP-003.yaml&version=v3.2.0

Submodel Registry Profile – Query Profile

Name:	Submodel Registry Bulk Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/SubmodelRegistryServiceSpecification/SSP-004
Feature	Appearance
APIs and API Operations	<i>Submodel Registry API:</i> QuerySubmodelDescriptors <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	Supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../SubmodelRegistryServiceSpecification/V3.2_SSP-004.yaml&version=v3.2.0

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/3/2/DiscoveryServiceSpecification/SSP-001
Feature	Appearance
API and API Operations	<i>AAS Basic Discovery API:</i> GetAllAssetAdministrationShellIdsByAssetLink (<<deprecated>>) SearchAllAssetAdministrationShellIdsByAssetLink GetAllAssetLinksById PostAllAssetLinksById DeleteAllAssetLinksById <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	Not supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../DiscoveryServiceSpecification/V3.2_SSP-001.yaml&version=v3.2.0

Discovery Service Specification – Read Profile

Name:	Discovery Service Full Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/DiscoveryServiceSpecification/SSP-002
Feature	Appearance
API and API Operations	<i>AAS Basic Discovery API:</i> SearchAllAssetAdministrationShellIdsByAssetLink GetAllAssetLinksById <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	Not supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../DiscoveryServiceSpecification/V3.2_SSP-002.yaml&version=v3.2.0

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
AssetAdministrationShellRepositoryServiceSpecification/SSP-004	Signature operations
AssetAdministrationShellRepositoryServiceSpecification/SSP-005	Operations on Identifiables only; is included in the profile AssetAdministrationShellRepositoryServiceSpecification/SSP-001
AssetAdministrationShellRepositoryServiceSpecification/SSP-006	Retrieves the Asset Administration Shell that was valid at a given date.

Asset Administration Shell Repository Service Specification – Full Profile

Name:	AAS Repository Full Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRepositoryServiceSpecification/SSP-001
Feature	Appearance

Name:	AAS Repository Full Profile
API and API Operations	<i>AAS Repository API:</i> GetAllAssetAdministrationShells GetAllAssetAdministrationShellsRecentChanges GetAssetAdministrationShellById GetAllAssetAdministrationShellsByAssetId GetAllAssetAdministrationShellsByIdShort PostAssetAdministrationShell PutAssetAdministrationShellById DeleteAssetAdministrationShellById <i>AAS API by superpath:</i> GetAllSubmodelReferences PostSubmodelReference DeleteSubmodelReference GetAssetInformation PutAssetInformation GetThumbnail PutThumbnail DeleteThumbnail <i>Submodel Repository API by superpath:</i> GetAllSubmodels GetSubmodelById GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort PostSubmodel PutSubmodelById PatchSubmodelById DeleteSubmodelById <i>Submodel API by superpath:</i> GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath PutFileByPath DeleteFileByPath PostSubmodelElement PostSubmodelElementByPath PutSubmodelElementByPath PatchSubmodelElementByPath PatchSubmodelElementByPath InvokeOperationSync InvokeOperationAsync GetOperationAsyncStatus GetOperationAsyncResult <i>AAS Serialization API:</i> GenerateSerializationByIds <i>Description API:</i> GetDescription

Name:	AAS Repository Full Profile
SerializationModifier	Level: Core, Deep Content: Normal, Metadata, Value, Reference, Path Extent: WithBLOBValue, WithoutBLOBValue
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AssetAdministrationShellRepositoryServiceSpecification/V3.2_SSP-001.yaml&version=v3.2.0

Note: [GetAllSubmodelsRecentChanges](#) is not included in the superpath because the superpath operations operate on a single Asset Administration Shell.

Asset Administration Shell Repository Service Specification – Read Profile

Name:	AAS Repository Read Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRepositoryServiceSpecification/SSP-002
Feature	Appearance

Name:	AAS Repository Read Profile
API and API Operations	<i>AAS Repository API:</i> GetAllAssetAdministrationShells GetAllAssetAdministrationShellsRecentChanges GetAssetAdministrationShellById GetAllAssetAdministrationShellsByAssetId [specification::interfaces::GetAllAssetAdministrationShellsByIdShort] <i>AAS API by superpath:</i> GetAssetAdministrationShell GetAllSubmodelReferences GetAllSubmodelReferences GetAllSubmodelReferences <i>Submodel Repository API by superpath:</i> GetAllSubmodelReferences GetAllSubmodelReferences GetAllSubmodelReferences GetAllSubmodelReferences <i>Submodel API by superpath:</i> GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath <i>Serialization API:</i> GenerateSerializationByIds <i>Description API:</i> GetDescription
SerializationModifier	Level: Core, Deep Content: Normal, Metadata, Value, Reference, Path Extent: WithBLOBValue, WithoutBLOBValue
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AssetAdministrationShellRepositoryServiceSpecification/V3.2_SSP-002.yaml&version=v3.2.0

Asset Administration Shell Repository Service Specification – Query Profile

Name:	AAS Repository Query Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRepositoryServiceSpecification/SSP-003
Feature	Appearance

Name:	AAS Repository Query Profile
API and API Operations	<i>AAS Repository API:</i> QueryAssetAdministrationShells <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AssetAdministrationShellRepositoryServiceSpecification/V3.2_SSP-003.yaml&version=v3.2.0

Asset Administration Shell Repository Service Specification – Signature Profile

Name:	AAS Repository Signature Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRepositoryServiceSpecification/SSP-004
Feature	Appearance
API and API Operations	<i>AAS Repository API:</i> GetAssetAdministrationShellByIdSigned <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	not supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AssetAdministrationShellRepositoryServiceSpecification/V3.2_SSP-004.yaml&version=v3.2.0

Asset Administration Shell Repository Service Specification – Identifiable Profile

Name:	AAS Repository Identifiable Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRepositoryServiceSpecification/SSP-005
Feature	Appearance

Name:	AAS Repository Identifiable Profile
API and API Operations	<i>AAS Repository API:</i> GetAssetAdministrationShellById PostAssetAdministrationShell PutAssetAdministrationShellById DeleteAssetAdministrationShellById <i>AAS API by superpath:</i> GetThumbnail PutThumbnail DeleteThumbnail <i>Submodel Repository API by superpath:</i> GetSubmodelById PostSubmodel PutSubmodelById DeleteSubmodelById <i>Submodel API by superpath:</i> GetFileByPath PutFileByPath DeleteFileByPath <i>Description API:</i> GetDescription
SerializationModifier	Level: Deep Content: Normal Extent: WithoutBLOBValue
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AssetAdministrationShellRepositoryServiceSpecification/V3.2_SSP-005.yaml&version=v3.2.0

Asset Administration Shell Repository Service Specification – The History Profile of the Asset Administration Shell Repository Service Specification

Name:	AAS Repository History Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/AssetAdministrationShellRepositoryServiceSpecification/SSP-006
Feature	Appearance
API and API Operations	<i>AAS Repository API:</i> GetAssetAdministrationShellVersionByIdAndDate <i>Description API:</i> GetDescription

Name:	AAS Repository History Profile
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	not supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../AssetAdministrationShellRepositoryServiceSpecification/V3.2_SSP-006.yaml&version=v3.2.0

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 <i>not</i> 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 <i>not</i> in the profile SubmodelRepositoryServiceSpecification/SSP-001 or the profile SubmodelRepositoryService Specification/SSP-002
SubmodelRepositoryServiceSpecification/SSP-005	Query operations
SubmodelRepositoryServiceSpecification/SSP-006	Signature operations
SubmodelRepositoryServiceSpecification/SSP-007	History Profile: Retrieves the Submodel that was valid at a given date.

Submodel Repository - Full Profile

Name:	Submodel Repository Full Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/SubmodelRepositoryServiceSpecification/SSP-001
Feature	Appearance

Name:	Submodel Repository Full Profile
API and API Operations	<i>Submodel Repository API:</i> GetAllSubmodels GetAllSubmodelsRecentChanges GetSubmodelById GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort PostSubmodel PutSubmodelById PatchSubmodelById DeleteSubmodelById <i>Submodel API by superpath:</i> GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath PutFileByPath DeleteFileByPath PostSubmodelElement PostSubmodelElementByPath PutSubmodelElementByPath PatchSubmodelElementByPath PatchSubmodelElementByPath InvokeOperationSync InvokeOperationAsync GetOperationAsyncStatus GetOperationAsyncResult <i>AAS Serialization API:</i> GenerateSerializationByIds <i>Description API:</i> GetDescription
SerializationModifier	Level: Core, Deep Content: Normal, Metadata, Value, Reference, Path Extent: WithBLOBValue, WithoutBLOBValue
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../SubmodelRepositoryServiceSpecification/V3.2_SSP-001.yaml&version=v3.2.0

Submodel Repository – Read Profile

Name:	Submodel Repository Read Profile
Profile Identifier	https://admin-shell.io/aas/API/3/2/SubmodelServiceSpecification/SSP-002
Feature	Appearance

Name:	Submodel Repository Read Profile
API and API Operations	<i>Submodel Repository API:</i> GetAllSubmodels GetAllSubmodelsRecentChanges GetSubmodelById GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort <i>Submodel API by superpath:</i> GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath <i>Serialization API:</i> GenerateSerializationByIds <i>Description API:</i> GetDescription
SerializationModifier	Level: Core, Deep Content: Normal, Metadata, Value, Reference, Path Extent: WithBLOBValue, WithoutBLOBValue
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../SubmodelRepositoryServiceSpecification/V3.2_SSP-002.yaml&version=v3.2.0

Submodel Repository - Template Profile

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

Constraint AASa-003: A service implementing the [SubmodelRepositoryServiceSpecification/SSP-003](#) must not accept or provide any [Submodel](#) with the attribute <https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/V3.2/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/3/2/SubmodelRepositoryServiceSpecification/SSP-003
Feature	Appearance
API and API Operations	<i>Submodel Repository API:</i> GetAllSubmodels GetAllSubmodelsRecentChanges GetSubmodelById GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort PostSubmodel PutSubmodelById PatchSubmodelById DeleteSubmodelById <i>Submodel API by superpath:</i> GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath PutFileByPath DeleteFileByPath PostSubmodelElement PostSubmodelElementByPath PutSubmodelElementByPath PatchSubmodelElementByPath PatchSubmodelElementByPath <i>AAS Serialization API:</i> GenerateSerializationByIds <i>Description API:</i> GetDescription
SerializationModifier	Level: Core, Deep Content: Normal, Metadata Extent: WithoutBLOBValue
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../SubmodelRepositoryServiceSpecification/V3.2_SSP-003.yaml&version=v3.2.0

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-](https://industrialdigitaltwin.io/aas-specifications/IDTA-01002/V3.2/http-rest-api/service-specifications-and-profiles.html#submodel_repository_template_read_profile)

004 must not accept or provide any _Submodel with the attribute https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/V3.2/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/3/2/SubmodelRepositoryServiceSpecification/SSP-004
Feature	Appearance
API and API Operations	<i>Submodel Repository API:</i> GetAllSubmodels GetAllSubmodelsRecentChanges GetSubmodelById GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort <i>Submodel API by superpath:</i> GetSubmodel GetAllSubmodelElements GetSubmodelElementByPath GetFileByPath <i>Serialization API:</i> GenerateSerializationByIds <i>Description API:</i> GetDescription
SerializationModifier	Level: Core, Deep Content: Normal, Metadata Extent: WithoutBLOBValue
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../SubmodelRepositoryServiceSpecification/V3.2_SSP-004.yaml&version=v3.2.0

Submodel Repository Service Specification – Query Profile

Name:	Submodel Repository Query Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/SubmodelRepositoryServiceSpecification/SSP-005

Name:	Submodel Repository Query Profile
Feature	Appearance
API and API Operations	<i>Submodel Repository API:</i> QuerySubmodels <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../SubmodelRepositoryServiceSpecification/V3.2_SSP-005.yaml&version=v3.2.0

Submodel Repository Service Specification – Signature Profile

Name:	Submodel Repository Signature Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/SubmodelRepositoryServiceSpecification/SSP-006
Feature	Appearance
API and API Operations	<i>Submodel Repository API:</i> GetSubmodelByIdSigned <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../SubmodelRepositoryServiceSpecification/V3.2_SSP-006.yaml&version=v3.2.0

Submodel Repository Service Specification – History Profile

Name:	Submodel Repository History Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/SubmodelRepositoryServiceSpecification/SSP-007
Feature	Appearance

Name:	Submodel Repository History Profile
API and API Operations	<i>Submodel Repository API:</i> GetSubmodelVersionByIdAndDate <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	not supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../SubmodelRepositoryServiceSpecification/V3.2_SSP-007.yaml&version=v3.2.0

Concept Description Repository Service Specification

Service Specification / Profiles	Description
ConceptDescriptionRepositoryServiceSpecification/SSP-001	Full feature set
ConceptDescriptionRepositoryServiceSpecification/SSP-002	Query operations
ConceptDescriptionRepositoryServiceSpecification/SSP-003	Signature operations

Concept Description Repository Service Specification – Full Profile

Name:	Concept Description Repository Full Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/ConceptDescriptionRepositoryServiceSpecification/SSP-001
Feature	Appearance
API and API Operations	<i>ConceptDescription Repository API:</i> GetAllConceptDescriptions GetAllConceptDescriptionsRecentChanges GetConceptDescriptionById GetAllConceptDescriptionsByIdShort GetAllConceptDescriptionsByIsCaseOf GetAllConceptDescriptionsByDataSpecificationReference PostConceptDescription PutConceptDescriptionById DeleteConceptDescriptionById <i>Serialization API:</i> GenerateSerializationByIds <i>Description API:</i> GetDescription

Name:	Concept Description Repository Full Profile
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	Supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../ConceptDescriptionRepositoryServiceSpecification/V3.2_SSP-001.yaml&version=v3.2.0

Concept Description Repository Service Specification – Query Profile

Name:	Concept Description Repository Query Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/ConceptDescriptionRepositoryServiceSpecification/SSP-002
Feature	Appearance
API and API Operations	<i>ConceptDescription Repository API:</i> QueryConceptDescriptions <i>Description API:</i> GetDescription
SerializationModifier	not applicable
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../ConceptDescriptionRepositoryServiceSpecification/V3.2_SSP-002.yaml&version=v3.2.0

Concept Description Repository Service Specification – Signature Profile

Name:	Concept Description Repository Signature Profile
Profile Identifier:	https://admin-shell.io/aas/API/3/2/ConceptDescriptionRepositoryServiceSpecification/SSP-003
Feature	Appearance
API and API Operations	<i>ConceptDescription Repository API:</i> GetConceptDescriptionByIdSigned <i>Description API:</i> GetDescription
SerializationModifier	not applicable

Name:	Concept Description Repository Signature Profile
SerializationFormat	JSON
Pagination	supported

See: https://industrialdigitaltwin.io/aas-specs-api/docs/swagger-ui.html?url=../ConceptDescriptionRepositoryServiceSpecification/V3.2_SSP-003.yaml&version=v3.2.0

Interactions

General

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 valid, 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.

Accessing Submodel Data

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.

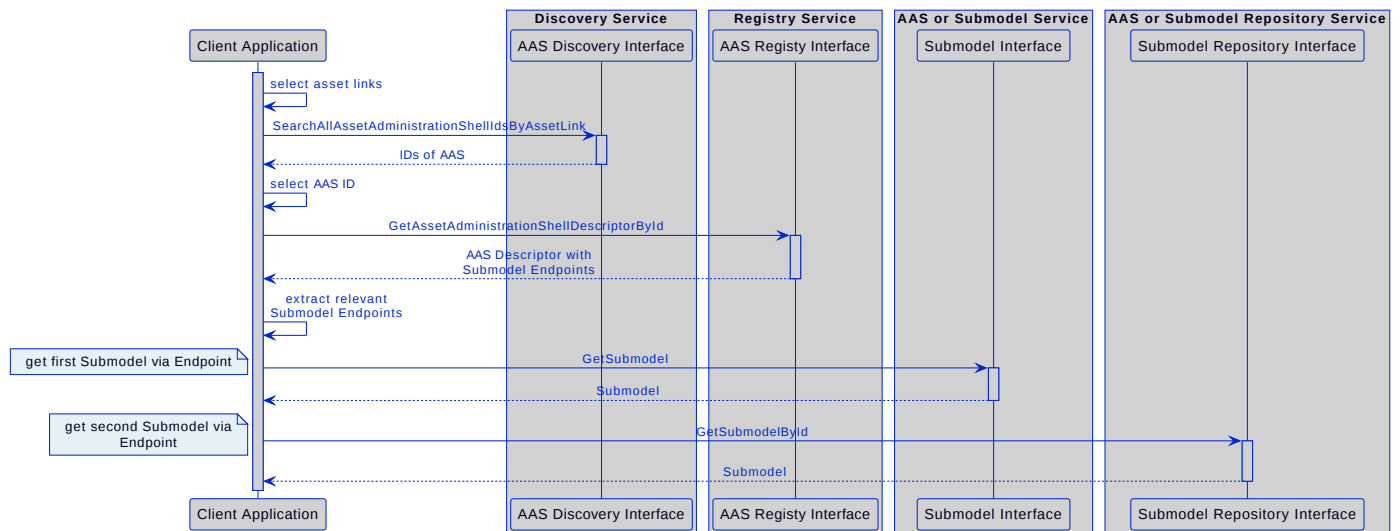


Figure 10. Interaction for Client Application using Submodel Endpoints

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.

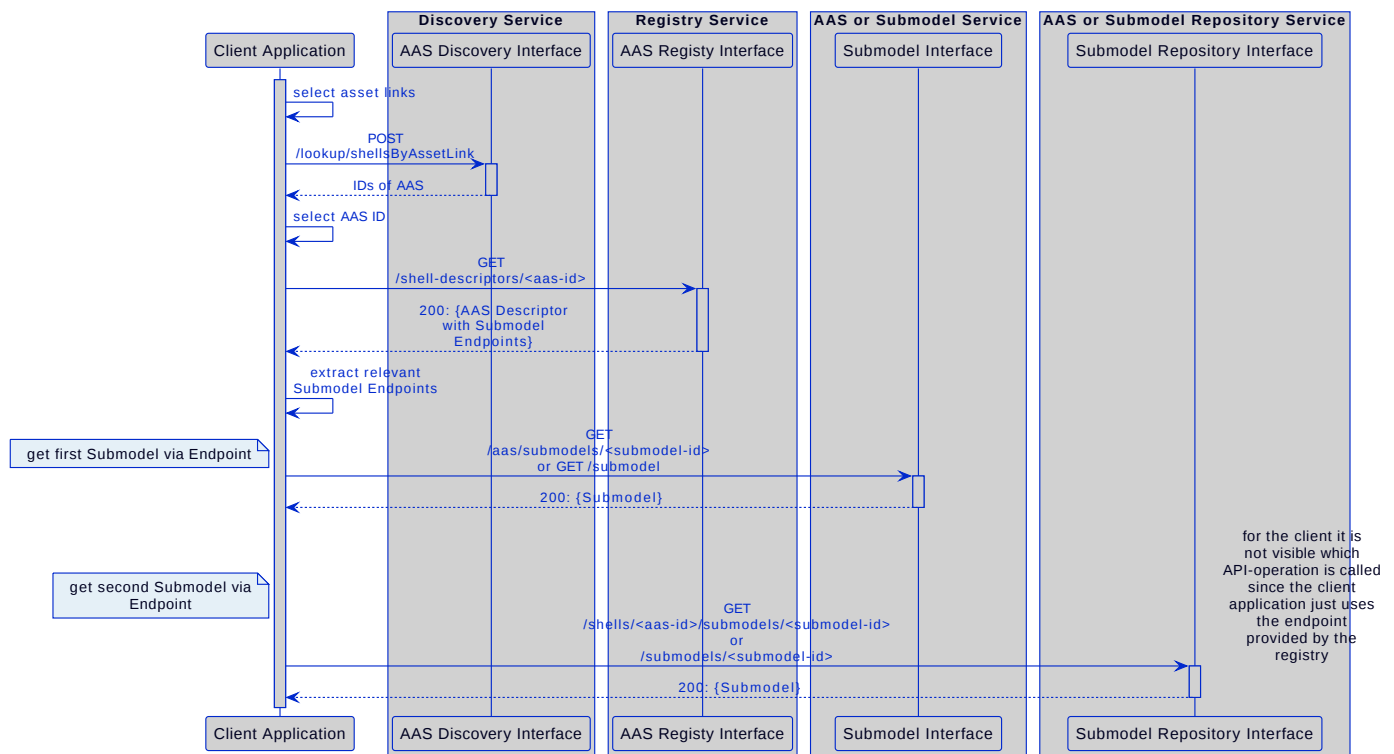


Figure 11. Interaction for Client Application using AAS Endpoints and Repositories

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.

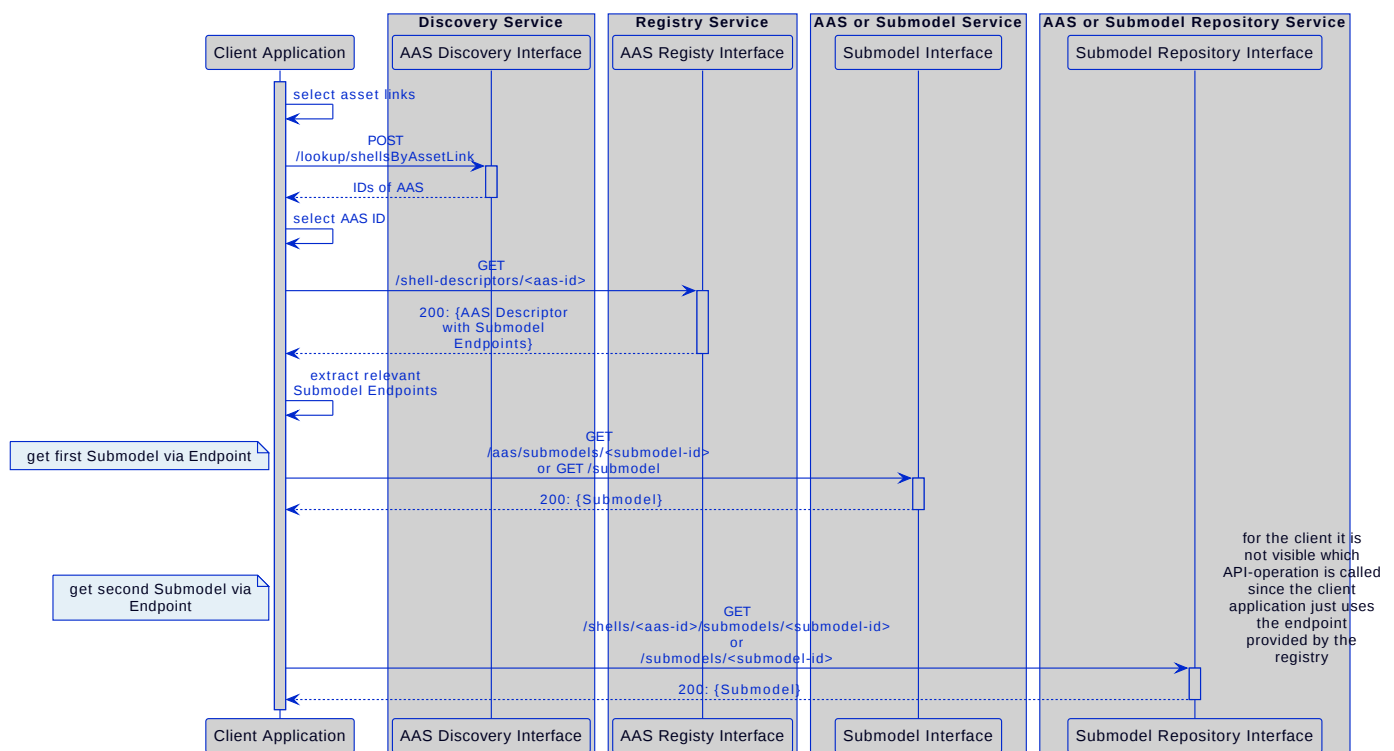


Figure 12. Interaction for Client Application using Submodel Endpoints (for HTTP API Operations)

Accessing recently changes objects

For recently changed Asset Administration Shells or Submodels there are the two API Operations

[GetAllSubmodelsRecentChanges](#) and [GetAllAssetAdministrationShellsRecentChanges](#).

For AAS Registry and Submodel Registry recently changed Asset Administration Shells or Submodels corresponding date input parameters are directly added to [GetAllAssetAdministrationShellDescriptors](#) and [GetAllSubmodelDescriptors](#).

In [Figure 13](#) a sequence interaction is described for an application looking for Asset Administration Shells created after a specified date interacting directly with an AAS repository. In [Figure 14](#) a sequence interaction is described for an application looking for Asset Administration Shells created after a specified date but using the AAS Registry as entry point.

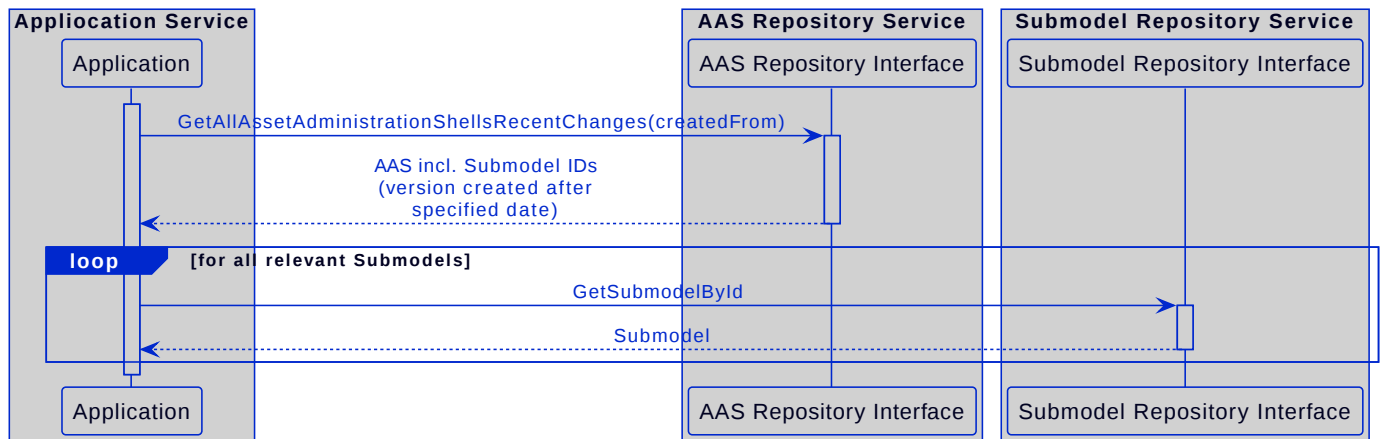


Figure 13. Interaction for Client Application looking for Recent Changes using an Asset Administration Shell Repository

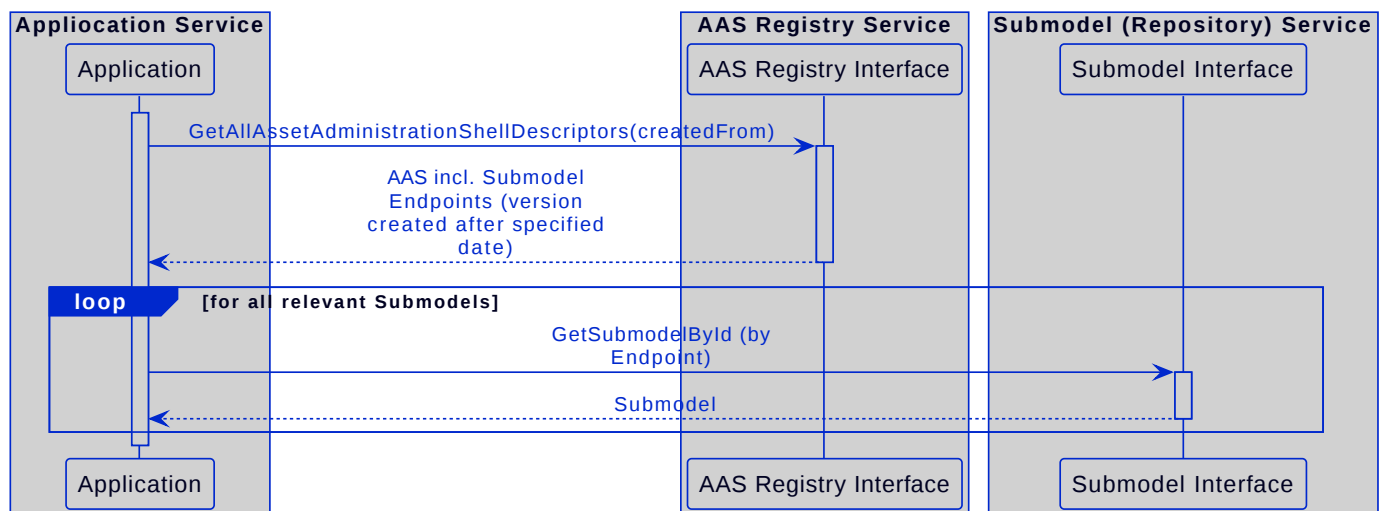


Figure 14. Interaction for Client Application looking for Recent Changes using an Asset Administration Shell Registry

Direct Discovery of AAS Objects based on ID Link and LinkSets

This section describes how a client can discover AAS and related resources by using an ID Link as the globalAssetId combined with RFC 9264 LinkSets. The ID Link is a resolvable HTTP(S) URL representing the asset identity (globalAssetId). The landing endpoint for the ID Link returns a LinkSet (media type application/linkset+json) that contains typed relations to AAS, Submodels, repositories, registries, and Digital Product Passport resources.

Concept: - The globalAssetId is an ID Link, e.g., <https://id.example.com/my-asset-id>. - A GET on the ID Link with Accept: application/linkset+json returns a LinkSet anchored at the ID Link. - The LinkSet provides relations such as: - aas: direct representations/endpoints of the Asset Administration Shell. - aas-service-endpoints: a LinkSet enumerating infrastructure endpoints (AAS Repository, Submodel Repository, Discovery, Registry). - dpp: direct representation of the Digital Product Passport. - dpp-service-endpoints: a LinkSet enumerating infrastructure endpoints for DPP services. - Links may carry metadata (profiles, authentication, versions) to guide client interaction.

Example for the content of the LinkSet returned by the ID Link endpoint:

```
{
  "linkset": [
    {
      "anchor": "https://id.example.com/my-asset-id",
      "aas": [
        {
          "href":
"https://service.example.com/aas/shells/aHR0cHM6Ly9pZC5leGFtcGx1LmNvbS9teS1hc3NldC1pZA",
          "type": "application/json",
          "profiles": {
            "https://admin-
shell.io/aas/API/3/2/AssetAdministrationShellServiceSpecification/SSP-001": [
              "GetAllAssetAdministrationShells",
              "PostAssetAdministrationShells",
              "GetAssetAdministrationShellById"
            ],
            "https://admin-
shell.io/aas/API/3/2/AssetAdministrationShellServiceSpecification/SSP-004": [
              "GetAssetAdministrationShellVersionByIdAndDate"
            ]
          },
          "authentication": [
            {
              "id": "oidc",
              "type": "OIDC",
              "well_known_url": "https://auth.example.com/realm/.well-known/openid-
configuration"
            },
            {
              "id": "another-oauth2",
              "type": "OAuth2",
              "base_url": "https://auth.example.com/realms/aas",
              "flows": ["client-credential-flow"],
              "scopes": ["whatever", "onemore"],
              "well_known_url": "https://auth.example.com/.well-known/oauth-authorization-
server"
            }
          ]
        }
      ]
    },
    {
      "aas-service-endpoints": [
        {
          "href": "https://service.example.com/aas/aas-service-endpoints",
          "type": "application/linkset+json",
          "returns": "linkset"
        }
      ],
      "dpp": [
        {
```

```

    "href": "https://service.example.com/dpp/v1/dpps/https%3A//id.example.com/my-asset-id",
    "type": "application/json",
    "dppSchemaVersion": "1.0", // the concrete url needs a concrete version, that's why single version is given here
    "delegatedActs": [ // I forgot the requested term...
      { "act": "BatteryPassport", "version": "1.0" }
    ]
  },
  "dpp-service-endpoint": {
    "href": "https://service.example.com/dpp", // Official API to append is 'v1/dpps/{dppId}'
    "type": "application/json",
    "dppSchemaVersions": ["1.0"], // This schema allows for multiple versions
    "delegatedActs": [
      { "act": "BatteryPassport", "version": "1.0" }
    ]
  }
}
]
}

```

Example for the content of the LinkSet returned by the AAS Service Endpoints endpoint:

```

{
  "linkset": [
    {
      "anchor": "https://service.example.com/aas/aas-service-endpoints",
      "aas-repository": [
        {
          "href": "https://serviceprovider.example.com/aas/shells",
          "type": "application/json",
          "profiles": {
            "https://admin-shell.io/aas/API/3/2/AssetAdministrationShellServiceSpecification/SSP-001": [
              "GetAllAssetAdministrationShells",
              "PostAssetAdministrationShells",
              "GetAssetAdministrationShellById"
            ],
            "https://admin-shell.io/aas/API/3/2/AssetAdministrationShellServiceSpecification/SSP-004": [
              "GetAssetAdministrationShellVersionByIdAndDate"
            ]
          },
          "authentication": [
            {
              "id": "objective-partner",
              "type": "OAuth2",
              "base_url": "https://auth.example.com/realms/aas",

```

```

        "flow": "client-credential-flow",
        "scope": "<whatever>"
    }
}
],
"sm-repository": [
{
    "href": "https://serviceprovider.example.com/aas/submodels",
    "type": "application/json",
    "profiles": {
        "https://admin-shell.io/aas/API/3/2/SubmodelServiceSpecification/SSP-001": [
            "GetAllSubmodels",
            "PostSubmodels",
            "GetSubmodelById"
        ],
        "https://admin-shell.io/aas/API/3/2/SubmodelServiceSpecification/SSP-004":
["GetSubmodelVersionByIdAndDate"]
    },
    "authentication": [
        {
            "id": "objective-partner",
            "type": "OAuth2",
            "base_url": "https://auth.example.com/realms/aas",
            "flow": "client-credential-flow",
            "scope": "<whatever>"
        }
    ]
}
]
}
]
}
}

```

Example for the content of the LinkSet returned by the ID Link endpoint:

```

{
  "linkset": [
    {
      "anchor": "https://service.example.com/dpp/dpp-service-endpoints",
      "dpp-service-endpoints": [
        {
          "href": "https://service.example.com/dpps/https%3A//id.example.com/my-asset-id",
          "type": "application/json",
          "provides": "ReadDPPByProductId"
        },
        {
          "href":
"https://service.example.com/dppsByProductIdAndDate/https%3A//id.example.com/my-asset-
id?date={date}",
          "type": "application/json",

```

```

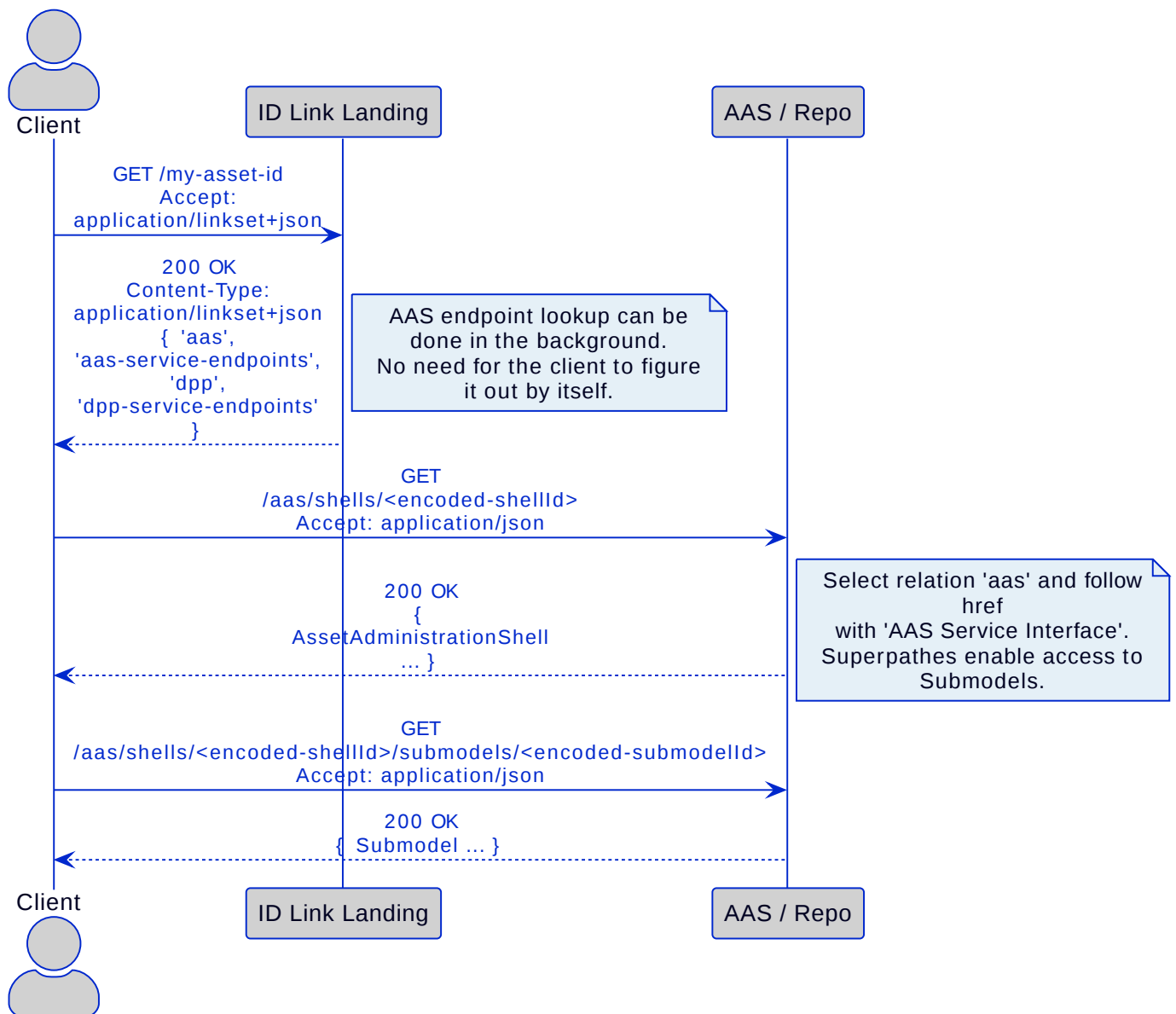
    "provides": "ReadDPPVersionByProductIdAndDate"
  }
]
}
}

```

Client interaction patterns:

1) LinkSet-aware client - Request: - GET <https://id.example.com/my-asset-id> - Accept: application/linkset+json, application/json;q=0.9 - Behavior: - Parse LinkSet and select the appropriate relation (e.g., "aas" for a direct shell, or "aas-service-endpoints" to obtain repositories). - Follow the selected href; apply authentication as indicated. - Example follow-up: - GET <https://service.example.com/aas/shells/aHR0cHM6Ly9pZC5leGFtcGxlMnVbS9teS1hc3NldC1pZA> - Accept: application/json

2) LinkSet-unaware client (fallback via redirect) - If the client requests JSON directly: - Server may respond 302 Found with Location pointing to the specific AAS resource. - Server may also include a Link header with rel="alternate" and type="application/linkset+json" to advertise discovery per RFC 8288.



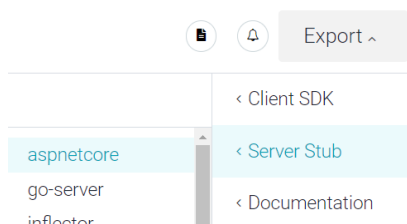
- Servers should support application/linkset+json (RFC 9264).

- Use RFC 8288 relations; include profiles and auth hints where applicable.
- The ID Link acts as a stable proxy; backend changes do not break clients.
- Using referredSemanticId on Submodels is recommended to help clients select relevant Submodels.

References: - RFC 8288 Web Linking: <https://www.rfc-editor.org/rfc/rfc8288.html> - RFC 9264 LinkSet: <https://www.rfc-editor.org/rfc/rfc9264.html>

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

[3] <http://semver.org>

[4] <http://semver.org>

[5] see Chapter 2.4 of RFC 8977

[6] 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, serialization formats for the API-operation "GenerateSerializationByIds" 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.

	Deep (default)	Core
Normal (default)	If applied to the Submodel: <pre> { "modelType": "Submodel", "id": "http://i40.customer.com/type/1/1/7A 7104BDAB57E184", "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" }] }] } </pre>	If applied to the Submodel: <pre> { "modelType": "Submodel", "id": "http://i40.customer.com/type/1/1/7A 7104BDAB57E184", "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" }] } </pre>

	Deep (default)	Core
Metadata	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>	

	Deep (default)	Core
Value	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>	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>

	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: <pre>{ "keys": [{ "type": "Submodel", "value": "http://i40.customer.com/type/1/1/7A7104BDAB57E184" }], "type": "ModelReference" }</pre> If applied to the SubmodelElementCollection, i.e. idShortPath "RotationSpeed": <pre>{ "keys": [{ "type": "Submodel", "value": "http://i40.customer.com/type/1/1/7A7104BDAB57E184" }, { "type": "SubmodelElementCollection", "value": "RotationSpeed" }], "type": "ModelReference" }</pre> If applied to the Property inside the SubmodelElementCollection, i.e. idShortPath “RotationSpeed.MaxRotationSpeed”: <pre>{ "keys": [{ "type": "Submodel", "value": "http://i40.customer.com/type/1/1/7A7104BDAB57E184" }, { "type": "SubmodelElementCollection", "value": "RotationSpeed" }, { "type": "Property", "value": "MaxRotationSpeed" }], }</pre>

	Deep (default)	Core
Path	If applied to the Submodel: Note: IdShortPaths always start at the first SubmodelElement. <pre>["RotationSpeed", "RotationSpeed.MaxRotationSpeed"]</pre> If applied to the SubmodelElementCollection: <pre>["RotationSpeed", "RotationSpeed.MaxRotationSpeed"]</pre> If applied to the Property inside the SubmodelElementCollection: <pre>["RotationSpeed.MaxRotationSpeed"]</pre>	If applied to the Submodel: Note: The SubmodelElementCollection "RotationSpeed" is the only direct child of the Submodel, therefore, it's the only entry. <pre>["RotationSpeed"]</pre> If applied to the SubmodelElementCollection: <pre>["RotationSpeed", "RotationSpeed.MaxRotationSpeed"]</pre> If applied to the Property inside the SubmodelElementCollection: <pre>["RotationSpeed.MaxRotationSpeed"]</pre>

Examples for PATCH Operations

	Deep (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:
	<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":
	5000

Digital Product Passport

Introduction

To support the implementation of DPPs, 8 standards have been developed:

- EN 18219:2026 – Digital product passport – Unique identifiers
- EN 18220:2026 – Digital product passport – Data carriers
- EN 18216:2026 – Digital product passport – Data exchange protocols
- EN 18222: 2026 – Digital Product Passport – Application Programming Interfaces (APIs) for the product passport lifecycle management and searchability
- EN 18223:2026 – Digital Product Passport – System interoperability
- EN 18221:2026 – Digital product passport – data storage, archiving, and data persistence
- EN 18239:2026 – Digital Product Passport – access rights management, information system security, and business confidentiality (not yet released)
- EN 18246:2026 – Digital Product Passport – Data authentication, reliability and integrity (not yet released)

This annex specifies how the requirements as specified in

- EN 18222:2026 – Application Programming Interfaces (APIs) for the product passport lifecycle management and searchability

can be implemented using the Asset Administration Shell (AAS) Services specification. The mapping enables implementations based on AAS infrastructure to expose a fully conformant DPP API.

The DPP operations defined in EN 18222 are organized into three API surfaces:

- **Life Cycle API - main API methods** (EN 18222, Clause 4, Table 16) — Mandatory read operations and CRUD operations for DPP lifecycle management.
- **DPP Registry API** (EN 18222, Clause 5, Table 17) — Registration of DPPs at the DPP Registry.

- **Life Cycle API - fine granular API methods** (EN 18222, Clause 6, Table 18) — Element-level read and update operations within a DPP.

The following sections describe, for each of these API surfaces, how the corresponding API methods shall be mapped to AAS Interface and API operations as well as services.

The terms "method" and "operation" are synonym in this context. In EN 18222 "method" is used, in IDTA-01002 "operation" is used.

The mapping of DPP data elements to AAS metamodel elements (EN 18223) is specified in [IDTA-01001](#). The DPP metadata Submodel Template is specified in [IDTA-02099-1 Digital Product Passport – Part 1: Metadata - Submodel Template Specification](#) with idShort "DppMetadata". A conformant DPP service based on AAS shall implement all aspects described in this annex together with the metamodel mapping and the Submodel Template.

DPP Composition from AAS Infrastructure

A Digital Product Passport as defined in EN 18223 is a single JSON or XML document comprising **header fields** and **content data elements**. When using AAS infrastructure, a DPP shall be composed from the following AAS resources:

- An **Asset Administration Shell** representing the product for which the DPP is issued.
- A **DppMetadata Submodel** (semanticId: <https://admin-shell.io/idta/cds/dppMetadata/1>, as defined in **IDTA-02099-1**) containing the DPP header fields (digitalProductPassportId, uniqueProductIdentifier, granularity, dppSchemaVersion, dppStatus, lastUpdate, economicOperatorId, facilityId, contentSpecificationIds).
- One or more **content Submodels**, each contributing a section of the DPP content. The semanticIds of the contributing content Submodels shall be listed in the contentSpecificationIds property of the DppMetadata Submodel.

The Value-Only serialization (see [IDTA-01001 Part 1 — Format "Value" \(Value-Only Serialization\) in JSON](#)) of the DppMetadata Submodel (IDTA-02099-1) corresponds directly to the DPP header fields defined in EN 18223, Table 1. No additional transformation of the header is required for neither the *compressed* nor the *expanded* representation (EN 18223, Clause 5.2). For the content Submodels however transformation is needed as defined in [Serialization Considerations](#).

Composition Process (Read Path)

To assemble a DigitalProductPassport JSON document, an implementation shall perform the following steps:

- **Resolve the DPP identifier to an AAS:** Determine the corresponding AAS, either by direct lookup of the DPP identifier or by resolving a product identifier through the AAS Discovery Service (see [Discovery of Digital Product Passports via AAS Discovery Sequences](#)).
- **Retrieve the AAS:** Fetch the AAS from the AAS Repository, the AAS Service or via the AAS Registry.
- **Retrieve all referenced Submodels:** Obtain the Submodel references from the AAS or AAS Descriptor and retrieve each Submodel from the Submodel Repository, Submodel Service or via the AAS Registry. If a **date** parameter is provided, the implementation shall retrieve the version of each Submodel that was current at the specified point in time.
- **Identify the DppMetadata Submodel:** Select the Submodel with semanticId <https://admin-shell.io/idta/cds/dppMetadata/1> (IDTA-02099-1). This Submodel provides the DPP header fields, including the Digital Product Passport ID.
- **Identify the content Submodels:** Select all remaining Submodels whose semanticId is listed in the contentSpecificationIds property of the DppMetadata Submodel.

- **Serialize the header:** Serialize the DppMetadata Submodel in Value-Only. The resulting JSON object constitutes the DPP header.
- **Serialize the content:** For each content Submodel, serialize the Submodel in the requested representation format (see [Serialization Considerations](#)). The JSON key for each content section shall be the Submodel's semanticId, equal to the contentSpecificationId, as described in IDTA-01099-1.
- **Assemble the DPP document:** Merge the header fields and the content sections into a single DigitalProductPassport JSON document.

For details see [Discovery of Digital Product Passports via AAS Discovery Sequences](#).

Decomposition Process (Write Path)

When a CreateDPP or UpdateDPPById operation is invoked with a DigitalProductPassport JSON document, an implementation shall decompose the document into the corresponding AAS resources:

- **Parse the DPP header fields:** Extract the header attributes (digitalProductPassportId, uniqueProductIdentifier, granularity, etc.) from the input document.
- **Create or update the AAS:** Map the header fields to the appropriate AAS attributes (e.g., uniqueProductIdentifier to AssetInformation/globalAssetId, granularity to assetKind).
- **Create or update the DppMetadata Submodel:** Persist all header fields as properties of the DppMetadata Submodel.
- **Create or update the content Submodels:** For each top-level content key in the DPP document, create or update the corresponding Submodel with the appropriate semanticId (as referenced in contentSpecificationIds).
- **Register Submodel references:** Ensure all Submodel references are registered in the AAS.

Write operations should be atomic: if any step fails, all preceding changes shall be rolled back to maintain data consistency.

Life Cycle API Operations

The Life Cycle API (EN 18222, Clause 4, Table 16) defines the mandatory read operations and CRUD operations for managing the DPP lifecycle. The following table specifies how each DPP operation shall be mapped to AAS Service operations.

Table 17. Mapping of DPP Life Cycle API Operations to AAS Service Operations

DPP Operation (EN 18222)	AAS Service Operation(s)	Description
ReadDPPById	GetAssetAdministrationShellById / GetAssetAdministrationShell or GetAssetAdministrationShellDescriptorById + GetSubmodel / GetSubmodelById (Value-Only or Normal) for each referenced and relevant Submodel	Shall retrieve the AAS by its identifier, fetch all referenced Submodels, and compose the DPP document as described in Composition Process (Read Path) . The representation query parameter controls the serialization format (see Serialization Considerations).
ReadDPPByProductId	GetAllAssetAdministrationShellsByAssetId / SearchAllAssetAdministrationShellIdsByAssetLink + GetSubmodel (Value-Only or Normal) for each referenced and relevant Submodel	Shall resolve the product identifier to an AAS (via globalAssetId) and compose the DPP document. [Note] === Multiple AAS instances might be associated with the provided product identifier. ===

DPP Operation (EN 18222)	AAS Service Operation(s)	Description
ReadDPPVersionByIdAndDate	GetAssetAdministrationShellVersionByIdAndDate / QueryAssetAdministrationShells / QueryAssetAdministrationShellDescriptors + GetSubmodel (Value-Only or Normal) for each referenced and relevant Submodel	Shall retrieve the version of the DPP that was current at the specified point in time (ISO 8601 UTC timestamp). The implementation shall select the latest version of each Submodel whose updatedAt timestamp is less than or equal to the requested date.
ReadDPPIdsByProductIds	GetAllAssetAdministrationShellIdsByAssetId / SearchAllAssetAdministrationShellIdsByAssetLink	Shall resolve a set of product identifiers to their corresponding DPP identifiers. The operation calls <code>GetAllAssetAdministrationShellIdsByAssetId</code> with the set of product identifiers and returns the union of all discovered DPP identifiers. Pagination via limit and cursor parameters should be supported.
CreateDPP	PostAssetAdministrationShell / PostAssetAdministrationShellDescriptor + CreateSubmodel (for DppMetadata and each content Submodel)	Shall decompose the submitted DPP document and create the corresponding AAS and Submodels as described in Decomposition Process (Write Path) . The DppMetadata Submodel shall be created for every new DPP. Shall return the digitalProductPassportId with HTTP status 201.
UpdateDPPById	PutAssetAdministrationShellById / PutAssetAdministrationShellDescriptorById + PutSubmodelById (for affected Submodels)	Shall accept a partial DPP document (JSON Merge Patch per RFC 7396). If DPP header fields are present, the DppMetadata Submodel shall be updated. If content fields are present, the corresponding content Submodels shall be updated. The lastUpdate field of the DppMetadata Submodel shall be updated accordingly. All changes shall be archived per EN 18221.
DeleteDPPById	DeleteAssetAdministrationShellById / DeleteAssetAdministrationShellDescriptorById (+ DeleteSubmodelReference / DeleteSubmodelDescriptorById + DeleteSubmodelById for the Submodel "DppMetadata")	Shall delete the AAS and Submodels that contain the DPP metadata. Implementations shall ensure compliance with applicable regulations regarding archiving periods and the prohibition of deleting DPPs for products that are still in use. Shall return HTTP status 204.

DPP Registry API Operations

The DPP Registry API (EN 18222, Clause 5, Table 17) defines operations for registering DPPs at a centrally hosted DPP Registry. When an AAS-based service acts as a data source for DPPs, it may also act as a client of the DPP Registry.

Although the AAS supports an AAS Registry this kind of Registry is not necessarily the same as for the DPP Registry and should not be mixed up. An AAS Registry can be used to implement a DPP Registry.

Table 18. Mapping of DPP Registry API Operations to AAS Service Operations

DPP Operation (EN 18222)	AAS Service Operation(s)	Description
RegisterProductDPP	Direct call to the DPP Registry	Upon creation of a new DPP, the implementation should register the DPP at the DPP Registry. So far no Submodel Template Specification is available for posting the DPP registration information.

The DPP Registry is typically a centrally hosted service operated independently from individual DPP data providers. The exact mechanism for registration may depend on the specific DPP Registry implementation and the applicable regulatory framework.

Fine Granular API Operations

The Fine Granular API (EN 18222, Clause 6, Table 18) defines operations for reading and updating individual data elements within a DPP without retrieving or modifying the entire document. The following table specifies how each fine-grained DPP operation shall be mapped to AAS SubmodelElement operations.

Table 19. Mapping of DPP Fine Granular API Operations to AAS SubmodelElement Operations

DPP Operation (EN 18222)	AAS Service Operation(s)	Description
ReadDataElement	GetSubmodelElementByPath or GetSubmodel / GetSubmodelById	Shall retrieve a single data element within a content Submodel by its element path. The <code>elementId</code> in the DPP maps to the <code>idShort</code> -based path in the AAS Submodel (see IDTA-01001 Part 1). In special cases it may also be the complete Submodel itself.
UpdateDataElement	PutSubmodelElementByPath or PutSubmodel / PutSubmodelById	Shall update a single data element within a content Submodel. The <code>lastUpdate</code> field of the <code>DppMetadata</code> Submodel should be updated accordingly.

For the mapping of DPP data element types (`SingleValuedDataElement`, `MultiValuedDataElement`, `DataElementCollection`, `MultiLanguageDataElement`, `RelatedResource`) to AAS SubmodelElement types (`Property`, `SubmodelElementList`, `SubmodelElementCollection`, `MultiLanguageProperty`, `File`), see [IDTA-01001 Part 1 — DPP Metamodel Mapping, Data Element Mapping](#).

Serialization Considerations

EN 18223 defines two JSON serialization formats for DPP content. These map to the existing AAS serialization formats as follows:

Table 20. Mapping of DPP Serialization Formats to AAS Serialization Formats

DPP Format (EN 18223)	AAS Serialization	Description
Compressed (EN 18223, Clause 5.2)	Value-Only (with minor differences as defined in IDTA-01001 Part 1 — DPP Metamodel Mapping, Serialization Compressed vs. Value-Only)	The <code>elementId</code> is used as the JSON key. Metadata (<code>dictionaryReference</code> , <code>valueDataType</code> , <code>objectType</code>) is omitted. This is the default and mandatory format.
Full / Expanded (EN 18223, Annex A)	Normal (with differences as defined in IDTA-01001 Part 1 — DPP Metamodel Mapping, Serialization Expanded vs. Normal)	Uses an <code>elements[]</code> array with explicit <code>elementId</code> , <code>objectType</code> , <code>dictionaryReference</code> , and <code>valueDataType</code> for each <code>DataElement</code> . Note that the AAS Normal serialization differs in structure: attribute names differ (<code>idShort</code> vs. <code>elementId</code> , <code>modelType</code> vs. <code>objectType</code> , <code>semanticId</code> vs. <code>dictionaryReference</code>), <code>semanticId</code> in AAS is a Reference object rather than a plain string, and the XSD data type prefix differs (<code>xs:</code> in AAS vs. <code>xsd:</code> in JTC24).

Implementations shall support the *compressed* representation. The *full* representation may be supported optionally.

The `representation` query parameter (EN 18222, Clause 8) controls which format is returned. If omitted, the *compressed* format shall be used as default.

For the metadata as defined in the Submodel Template Specification IDTA-02099-1 the payload is identical for compressed and expanded serialization. It corresponds to the Value-Only serialization of the AAS.

Discovery of Digital Product Passports via AAS Discovery Sequences

The discovery of Digital Product Passports can be implemented using the AAS discovery steps defined in [IDTA-01002 Part 2 — Figure 2](#). These steps enable clients to discover and access AAS instances based on asset identifiers, including the product identifiers used in Digital Product Passports.

A typical DPP discovery sequence proceeds as follows:

1. A client application obtains a product identifier, for instance by scanning a data carrier (e.g., QR code) on the product. The client executes the DPP operation `ReadDPPByProductId`, passing the product identifier as input.
2. The DPP Service invokes the [SearchAllAssetAdministrationShellIdsByAssetLink](#) operation on the AAS Discovery Service, passing the product identifier as the `globalAssetId`.
3. The Discovery Service returns the identifiers of AAS instances associated with the product identifier.
4. The DPP Service selects the appropriate AAS identifier and invokes [GetAssetAdministrationShellDescriptorById](#) on the AAS Registry Service to obtain the AAS Descriptor, which includes the endpoints for accessing the AAS and/or its Submodels (see sequence [Figure 15](#)), or
5. directly retrieves the AAS using [GetAssetAdministrationShellById](#) on the AAS Repository or Service (see sequence [Figure 16](#)).
6. The DPP Service retrieves the relevant Submodels and composes the DPP document.

To enable discovery, the economic operator responsible for the product shall ensure that the AAS instance representing the DPP is linked to the product identifier via the `globalAssetId` or `specificAssetId` fields in the `AssetInformation` of the AAS. The `uniqueProductIdentifier` in the `DppMetadata` Submodel shall correspond to the `globalAssetId` of the AAS. The explicit mapping is defined in IDTA-02099-1.

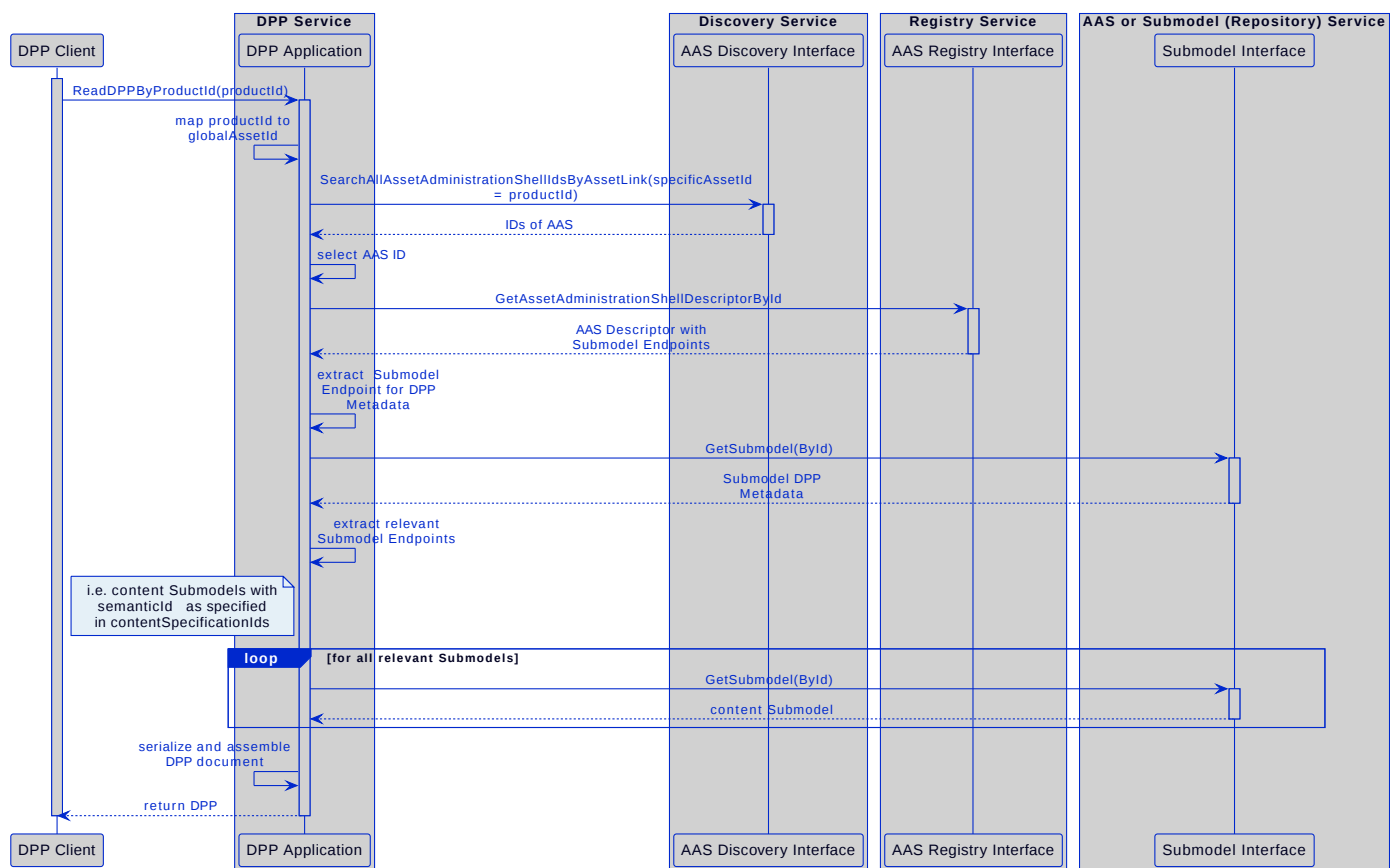


Figure 15. DPP Composition Sequence via AAS Discovery, Registry, and Submodel Services

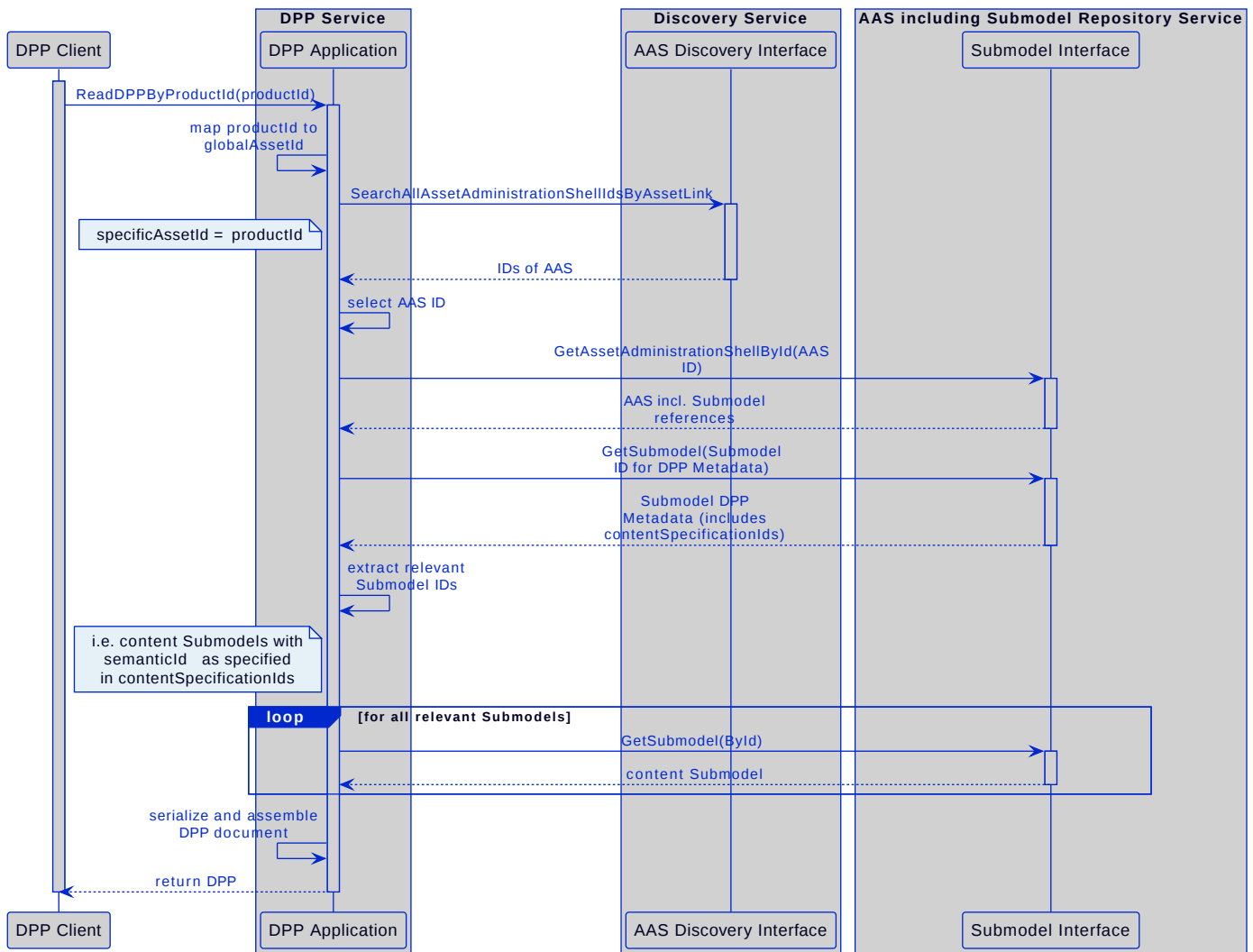


Figure 16. DPP Composition Sequence via AAS Discovery and AAS/Submodel Repository

Figure 15 illustrates the composition process for a `ReadDPPByProductId` interaction. The DPP Application discovers the relevant AAS via the Discovery Service, resolves the Submodel endpoints via the Registry Service, retrieves the relevant Submodels, and merges them into a conformant DPP payload.

Retrieving Historical Versions of DPPs via AAS Versioning Mechanisms ==

EN 18222 defines the `ReadDPPVersionByIdAndDate` operation for retrieving the version of a DPP that was current at a specified point in time. When using AAS infrastructure, this can be implemented by leveraging the versioning and timestamping features of AAS resources.

To implement `ReadDPPVersionByIdAndDate`, an implementation shall perform the following steps:

- Retrieve the AAS and all referenced Submodels as described in [Composition Process \(Read Path\)](#) but considering the `updatedAt` time information within the AAS and its Submodels.
- For each Submodel, select the version whose `updatedAt` timestamp is less than or equal to the requested date. This can be achieved by using the `date` query parameter when retrieving the Submodel, which instructs the AAS Service to return the version that was current at that point in time.
- Compose the DPP document from the selected versions of the Submodels as described in [Composition Process \(Read Path\)](#).

In [Figure 17](#) the steps to be performed by directly accessing an AAS and Submodel Repository is shown. In [Figure 18](#) the steps to be performed when using an AAS Registry with Submodel Endpoints contained in the `AssetAdministrationShellDescriptor` is shown. In [Figure 19](#) the steps to be performed when using an AAS Registry with AAS Endpoints and a Submodel Registry with Submodel Endpoints is shown.

The endpoints contained in the descriptors implicitly map to corresponding [GetAssetAdministrationShellVersionByIdAndDate](#) or [GetSubmodelVersionByIdAndDate](#) operation calls.

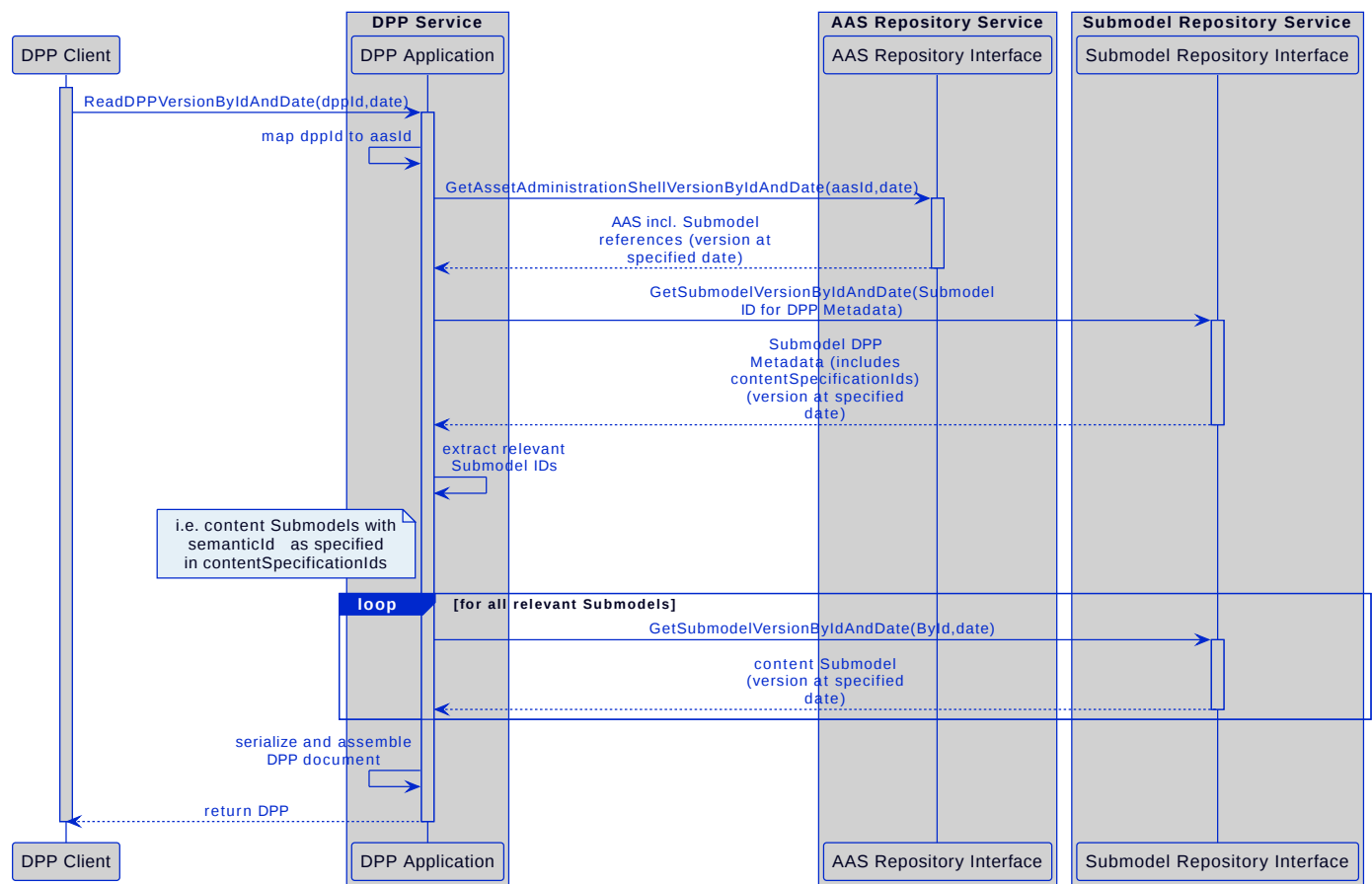


Figure 17. ReadDPPVersionByProductIdAndDate Sequence via AAS Versioning with Repository

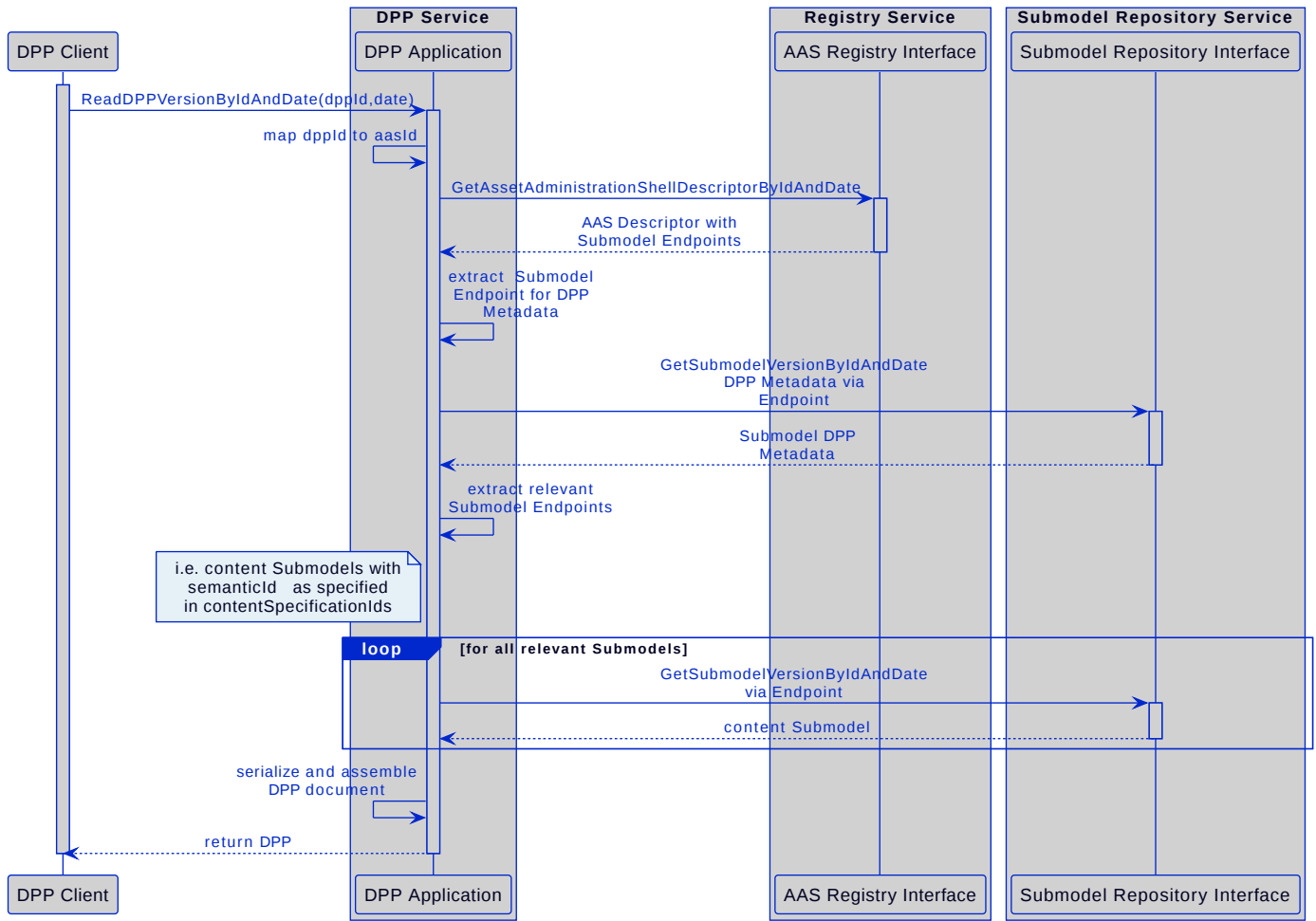


Figure 18. ReadDPPVersionByProductIdAndDate Sequence via AAS Registry

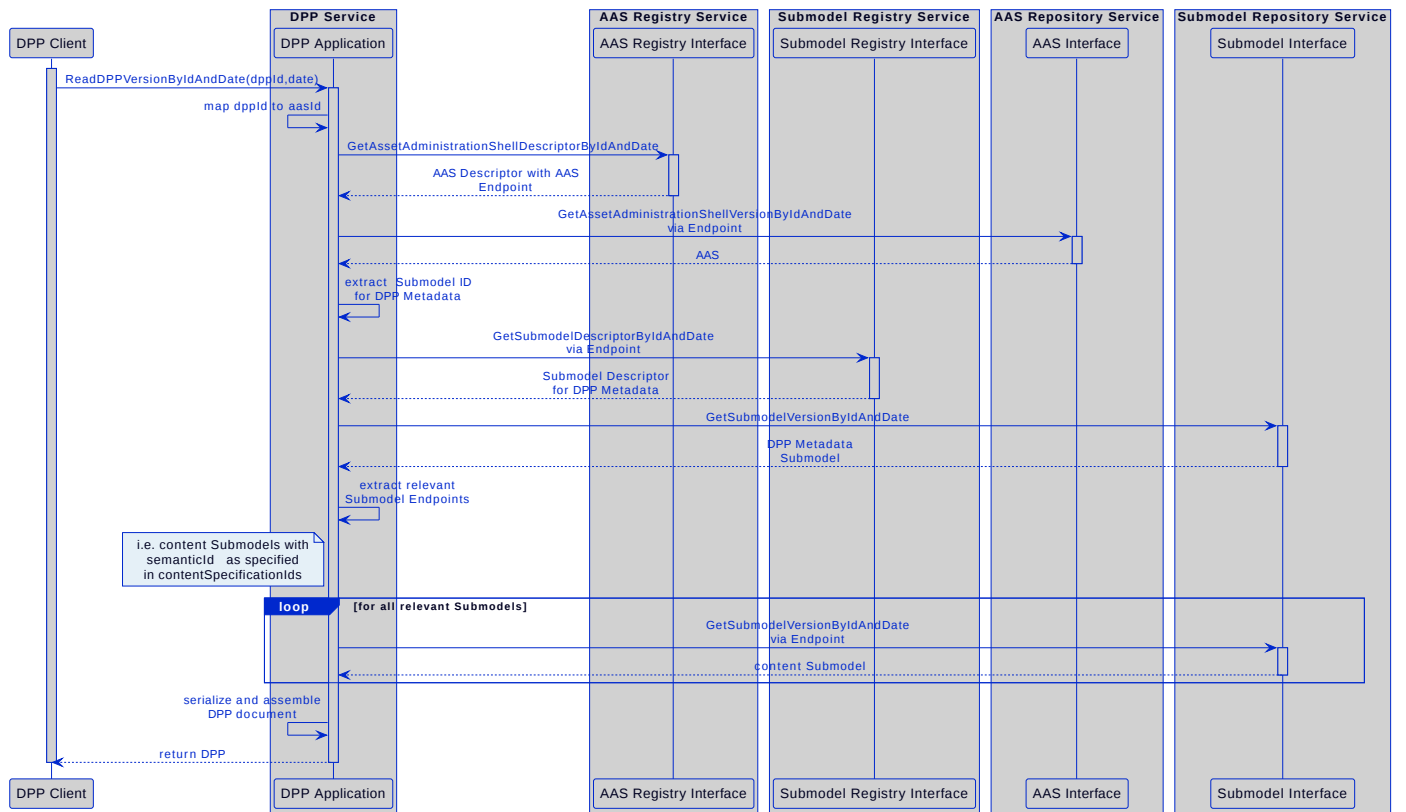


Figure 19. ReadDPPVersionByProductIdAndDate Sequence via AAS and Submodel Registry

HTTP API Mapping

The DPP operations described above shall be exposed through an HTTP REST API following the specifications of the Asset Administration Shell Services (EN 18222, Clause 8). In-path identifiers shall be percent-encoded.

In-path identifiers in AAS are base64url-encoded.

Table 21. Exemplary HTTP Mapping of DPP Operations to AAS Repository Endpoints

DPP Operation	HTTP Method	DPP Path (EN 18222)	AAS Equivalent Endpoint
ReadDPPById	GET	v1/dpps/{dppld}	shells/{aasId} + submodels/{smlId}/\$value (per Submodel)
ReadDPPByProductId	GET	v1/dppsByProductId/{productId}	shells?assetIds={productId} + Submodel retrieval
ReadDPPVersionByIdAnd Date	GET	v1/dppsByIdAndDate/{dppld}?date={timestamp}	shells/{aasId}/\$history + submodels/{smlId}/\$value with date filter
ReadDPPIdsByProductIds	POST	v1/dppsByProductIds	lookup/shells (Discovery) for each product ID
CreateDPP	POST	v1/dpps	shells (POST) + submodels (POST per Submodel)
UpdateDPPById	PATCH	v1/dpps/{dppld}	shells/{aasId} (PUT) + submodels/{smlId} (PUT per Submodel)
DeleteDPPById	DELETE	v1/dpps/{dppld}	shells/{aasId} (DELETE)
RegisterProductDPP	POST	v1/registerDPP	registry/shell-descriptors (POST) or DPP Registry API
ReadDataElement	GET	v1/dpps/{dppld}/elements/{elementPath}	submodels/{smlId}/submodel-elements/{elementPath}/\$value
UpdateDataElement	PATCH	v1/dpps/{dppld}/elements/{elementPath}	submodels/{smlId}/submodel-elements/{elementPath}

Compliance Considerations

Implementations of a DPP service based on AAS infrastructure shall ensure compliance with the applicable regulatory requirements. The following aspects are of particular importance:

Archiving: All changes to a DPP shall be tracked and archived per EN 18221. The AAS Services support this through the `updatedAt` attribute in `AdministrativeInformation` and the `date` query parameter for retrieving historical versions. Implementations shall maintain a complete audit trail of all modifications to a DPP.

Deletion restrictions: Applicable regulations may prohibit the deletion of a DPP while the corresponding product is still in use or within mandatory retention periods. AAS implementations exposing DPPs shall enforce these restrictions

regardless of whether the DeleteDPPById operation is supported. The AAS Services specification provides the DeleteAssetAdministrationShellById operation, but its use shall be governed by the applicable regulatory framework.

Access control: Implementations should enforce appropriate access control policies on DPP operations, in accordance with EN 18222, Clause 4.2–4.5. Different data elements within a DPP may require different levels of authorization depending on the stakeholder and their role in the product lifecycle. For further details see [IDTA-01004 Part 4](#).

Error handling: All error responses shall conform to the Result object structure defined in EN 18222, Clause 7, containing an array of Message objects with messageType, text, code, correlationId, and timestamp. The HTTP status codes shall follow EN 18222, Table 15.

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-Adresse> ::= <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 22. 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 Identifiabiles 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 23. 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 24. 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 25. Interface Description

Interface: <Interface Name>	
Operation Name	Description
<i>Oper1</i>	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.
...	
<i>operN</i> (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 26. Operation Description

Operation Name:	<i>Name of the operation: all individual words in the operation name are capitalized</i>
------------------------	--

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 <i>Type</i>. 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				
<i>inputParameter1</i>	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..*").
...				

<i>inputParameterN</i>	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				
<i>outputParameter1</i>	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..*").
...				
<i>outputParameterN</i>	Human-understandable description of the output parameter N of the operation.	States whether the outputParameterN is mandatory ("yes") or optional ("no")	Type of the output parameter N	The cardinality of type of the outputParameterN, e.g. zero-to-one ("0..1") or at-least-one ("1..*").

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.

[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/V3.2/spec-metamodel/common.html#ModellingKind>.

Constraint AASa-004: A service implementing the [SubmodelRepositoryServiceSpecification/SSP-004](https://industrialdigitaltwin.io/aas-specifications/IDTA-01002/V3.2/http-rest-api/service-specifications-and-profiles.html#submodel_repository_template_read_profile>SubmodelRepositoryServiceSpecification/SSP-004) must not accept or provide any [_Submodel](#) with the attribute <https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/V3.2/spec-metamodel/common.html#ModellingKind>.

Constraint AASa-005: Only the [_SubmodelElements_](#) root declaration can be followed with [IdShortPaths](#).

Operation to RIGHT Mapping

This annex defines the normative mapping from each AAS HTTP/REST API operation to the RIGHT value that a security implementation (see IDTA-01004 Access Rule Model, clause "Rights and operation verbs") MUST use when evaluating access rules for the operation.

The mapping is deterministic: if an implementation of IDTA-01004 denies a request with a given RIGHT, it MUST deny the operations that are listed for that RIGHT below, regardless of HTTP method.

The table uses the following conventions:

- Operation Name corresponds to the `operationId` in the combined OpenAPI document.
- URL Template is the path template per [HTTP/REST API](#).
- RIGHT is the RIGHT value from IDTA-01004 `rightsEnum`. When two RIGHTS are listed (for example for PUT on client-addressable resources), the security implementation MUST resolve the right at request time, based on whether the targeted resource already exists (UPDATE) or not (CREATE). originating RIGHT means that an asynchronous status or result request MUST use the RIGHT that was resolved for the initiating request when its handle was created.
- ROUTE literal example shows a compatible ROUTE: pattern for access rules. In accordance with the IDTA-01004 Access Rule Model [\[3\]](#), wildcards are used only as trailing prefix wildcards.

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
GetAssetAdministrationShell	GET	/aas	READ	"/aas"
PutAssetAdministrationShell	PUT	/aas	UPDATE	"/aas"
GetAssetAdministrationShell-Reference	GET	/aas/\$reference	VIEW	"/aas/\$reference"
GetAssetInformation	GET	/aas/asset-information	READ	"/aas/asset-information"
PutAssetInformation	PUT	/aas/asset-information	UPDATE	"/aas/asset-information"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
GetThumbnail	GET	/aas/asset-information/thumbnail	READ	"/aas/asset-information/thumbnail"
PutThumbnail	PUT	/aas/asset-information/thumbnail	UPDATE	"/aas/asset-information/thumbnail"
DeleteThumbnail	DELETE	/aas/asset-information/thumbnail	DELETE	"/aas/asset-information/thumbnail"
GetAllSubmodelReferences	GET	/aas/submodel-refs	VIEW	"/aas/submodel-refs"
PostSubmodelReference	POST	/aas/submodel-refs	CREATE	"/aas/submodel-refs"
DeleteSubmodelReference	DELETE	/aas/submodel-refs/{submodelIdentifier}	DELETE	"/aas/submodel-refs/*"
GetSubmodelById_AAS	GET	/aas/submodels/{submodelIdentifier}	READ	"/aas/submodels/*"
PutSubmodelById_AAS	PUT	/aas/submodels/{submodelIdentifier}	CREATE or UPDATE	"/aas/submodels/*"
PatchSubmodelById_AAS	PATCH	/aas/submodels/{submodelIdentifier}	UPDATE	"/aas/submodels/*"
DeleteSubmodelById_AAS	DELETE	/aas/submodels/{submodelIdentifier}	DELETE	"/aas/submodels/*"
GetSubmodelById-Metadata_AAS	GET	/aas/submodels/{submodelIdentifier}/\$metadata	READ	"/aas/submodels/*"
PatchSubmodelById-Metadata_AAS	PATCH	/aas/submodels/{submodelIdentifier}/\$metadata	UPDATE	"/aas/submodels/*"
GetSubmodelById-ValueOnly_AAS	GET	/aas/submodels/{submodelIdentifier}/\$value	READ	"/aas/submodels/*"
PatchSubmodelById-ValueOnly_AAS	PATCH	/aas/submodels/{submodelIdentifier}/\$value	UPDATE	"/aas/submodels/*"
GetSubmodelById-Reference_AAS	GET	/aas/submodels/{submodelIdentifier}/\$reference	VIEW	"/aas/submodels/*"
GetSubmodelById-Path_AAS	GET	/aas/submodels/{submodelIdentifier}/\$path	VIEW	"/aas/submodels/*"
GetAllSubmodelElements_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements	READ	"/aas/submodels/*"
PostSubmodelElement_AAS	POST	/aas/submodels/{submodelIdentifier}/submodel-elements	CREATE	"/aas/submodels/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
GetAllSubmodelElements-Metadata_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements/\$metadata	READ	"/aas/submodels/*"
GetAllSubmodelElements-ValueOnly_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements/\$value	READ	"/aas/submodels/*"
GetAllSubmodelElementsReference_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements/\$reference	VIEW	"/aas/submodels/*"
GetAllSubmodelElementsPath_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements/\$path	VIEW	"/aas/submodels/*"
GetSubmodelElementByPath_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	READ	"/aas/submodels/*"
PostSubmodelElementByPath_AAS	POST	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	CREATE	"/aas/submodels/*"
PutSubmodelElementByPath_AAS	PUT	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	CREATE or UPDATE	"/aas/submodels/*"
PatchSubmodelElementValueByPath_AAS	PATCH	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	UPDATE	"/aas/submodels/*"
DeleteSubmodelElementByPath_AAS	DELETE	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	DELETE	"/aas/submodels/*"
GetSubmodelElementByPath-Metadata_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$metadata	READ	"/aas/submodels/*"
PatchSubmodelElementValueByPath-Metadata_AAS	PATCH	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$metadata	UPDATE	"/aas/submodels/*"
GetSubmodelElementByPath-ValueOnly_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$value	READ	"/aas/submodels/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
PatchSubmodelElementValueByPathValueOnly_AAS	PATCH	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$value	UPDATE	"/aas/submodels/*"
GetSubmodelElementByPath-Reference_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$reference	VIEW	"/aas/submodels/*"
GetSubmodelElementByPath-Path_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$path	VIEW	"/aas/submodels/*"
GetFileByPath_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/attachment	READ	"/aas/submodels/*"
PutFileByPath_AAS	PUT	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/attachment	UPDATE	"/aas/submodels/*"
DeleteFileByPath_AAS	DELETE	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/attachment	DELETE	"/aas/submodels/*"
InvokeOperationSync_AAS	POST	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/invoke	EXECUTE	"/aas/submodels/*"
InvokeOperationSync-ValueOnly_AAS	POST	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/invoke/\$value	EXECUTE	"/aas/submodels/*"
InvokeOperationAsync_AAS	POST	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/invoke-async	EXECUTE	"/aas/submodels/*"
InvokeOperationAsync-ValueOnly_AAS	POST	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/invoke-async/\$value	EXECUTE	"/aas/submodels/*"
GetOperationAsyncStatus_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/operation-status/{handleId}	EXECUTE	"/aas/submodels/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
GetOperationAsyncResult_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/operation-results/{handleId}	EXECUTE	"/aas/submodels/*"
GetOperationAsyncResult-ValueOnly_AAS	GET	/aas/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/operation-results/{handleId}/\$value	EXECUTE	"/aas/submodels/*"
GetSubmodel	GET	/submodel	READ	"/submodel"
PutSubmodel	PUT	/submodel	UPDATE	"/submodel"
PatchSubmodel	PATCH	/submodel	UPDATE	"/submodel"
GetSubmodel-Metadata	GET	/submodel/\$metadata	READ	"/submodel/\$metadata"
PatchSubmodel-Metadata	PATCH	/submodel/\$metadata	UPDATE	"/submodel/\$metadata"
GetSubmodel-ValueOnly	GET	/submodel/\$value	READ	"/submodel/\$value"
PatchSubmodel-ValueOnly	PATCH	/submodel/\$value	UPDATE	"/submodel/\$value"
GetSubmodel-Reference	GET	/submodel/\$reference	VIEW	"/submodel/\$reference"
GetSubmodel-Path	GET	/submodel/\$path	VIEW	"/submodel/\$path"
GetAllSubmodelElements	GET	/submodel/submodel-elements	READ	"/submodel/submodel-elements"
PostSubmodelElement	POST	/submodel/submodel-elements	CREATE	"/submodel/submodel-elements"
GetAllSubmodelElements-Metadata	GET	/submodel/submodel-elements/\$metadata	READ	"/submodel/submodel-elements/\$metadata"
GetAllSubmodelElements-ValueOnly	GET	/submodel/submodel-elements/\$value	READ	"/submodel/submodel-elements/\$value"
GetAllSubmodelElements-Reference	GET	/submodel/submodel-elements/\$reference	VIEW	"/submodel/submodel-elements/\$reference"
GetAllSubmodelElements-Path	GET	/submodel/submodel-elements/\$path	VIEW	"/submodel/submodel-elements/\$path"
GetSubmodelElementByPath	GET	/submodel/submodel-elements/{idShortPath}	READ	"/submodel/submodel-elements/*"
PostSubmodelElementByPath	POST	/submodel/submodel-elements/{idShortPath}	CREATE	"/submodel/submodel-elements/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
PutSubmodelElementByPath	PUT	/submodel/submodel-elements/{idShortPath}	CREATE or UPDATE	"/submodel/submodel-elements/*"
PatchSubmodelElementByPath	PATCH	/submodel/submodel-elements/{idShortPath}	UPDATE	"/submodel/submodel-elements/*"
DeleteSubmodelElementByPath	DELETE	/submodel/submodel-elements/{idShortPath}	DELETE	"/submodel/submodel-elements/*"
GetSubmodelElementByPath-Metadata	GET	/submodel/submodel-elements/{idShortPath}/\$metadata	READ	"/submodel/submodel-elements/*"
PatchSubmodelElementByPath-Metadata	PATCH	/submodel/submodel-elements/{idShortPath}/\$metadata	UPDATE	"/submodel/submodel-elements/*"
GetSubmodelElementByPath-ValueOnly	GET	/submodel/submodel-elements/{idShortPath}/\$value	READ	"/submodel/submodel-elements/*"
PatchSubmodelElementByPath-ValueOnly	PATCH	/submodel/submodel-elements/{idShortPath}/\$value	UPDATE	"/submodel/submodel-elements/*"
GetSubmodelElementByPath-Reference	GET	/submodel/submodel-elements/{idShortPath}/\$reference	VIEW	"/submodel/submodel-elements/*"
GetSubmodelElementByPath-Path	GET	/submodel/submodel-elements/{idShortPath}/\$path	VIEW	"/submodel/submodel-elements/*"
GetFileByPath	GET	/submodel/submodel-elements/{idShortPath}/attachment	READ	"/submodel/submodel-elements/*"
PutFileByPath	PUT	/submodel/submodel-elements/{idShortPath}/attachment	UPDATE	"/submodel/submodel-elements/*"
DeleteFileByPath	DELETE	/submodel/submodel-elements/{idShortPath}/attachment	DELETE	"/submodel/submodel-elements/*"
InvokeOperation	POST	/submodel/submodel-elements/{idShortPath}/invoke	EXECUTE	"/submodel/submodel-elements/*"
InvokeOperationSync-ValueOnly	POST	/submodel/submodel-elements/{idShortPath}/invoke/\$value	EXECUTE	"/submodel/submodel-elements/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
InvokeOperationAsync	POST	/submodel/submodel-elements/{idShortPath}/invoke-async	EXECUTE	"/submodel/submodel-elements/*"
InvokeOperationAsync-ValueOnly	POST	/submodel/submodel-elements/{idShortPath}/invoke-async/\$value	EXECUTE	"/submodel/submodel-elements/*"
GetOperationAsyncStatus	GET	/submodel/submodel-elements/{idShortPath}/operation-status/{handleId}	EXECUTE	"/submodel/submodel-elements/*"
GetOperationAsyncResult	GET	/submodel/submodel-elements/{idShortPath}/operation-results/{handleId}	EXECUTE	"/submodel/submodel-elements/*"
GetOperationAsyncResult-ValueOnly	GET	/submodel/submodel-elements/{idShortPath}/operation-results/{handleId}/\$value	EXECUTE	"/submodel/submodel-elements/*"
GetAllAssetAdministrationShells	GET	/shells	READ	"/shells"
PostAssetAdministrationShell	POST	/shells	CREATE	"/shells"
GetAllAssetAdministrationShellsRecentChanges	GET	/shells/\$recent-changes	READ	"/shells/\$recent-changes"
GetAllAssetAdministrationShells-Reference	GET	/shells/\$reference	VIEW	"/shells/\$reference"
GetAssetAdministrationShellById	GET	/shells/{aasIdentifier}	READ	"/shells/*"
PutAssetAdministrationShellById	PUT	/shells/{aasIdentifier}	CREATE or UPDATE	"/shells/*"
DeleteAssetAdministrationShellById	DELETE	/shells/{aasIdentifier}	DELETE	"/shells/*"
GetAssetAdministrationShellVersionByIdAndDate	GET	/shells/{aasIdentifier}/\$history	READ	"/shells/*"
GetAssetAdministrationShellById-Reference_AasRepository	GET	/shells/{aasIdentifier}/\$reference	VIEW	"/shells/*"
GetAssetAdministrationShellByIdSigned	GET	/shells/{aasIdentifier}/\$signed	READ	"/shells/*"
GetAssetInformation_AasRepository	GET	/shells/{aasIdentifier}/asset-information	READ	"/shells/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
PutAssetInformation_AasRepository	PUT	/shells/{aasIdentifier}/asset-information	UPDATE	"/shells/*"
GetThumbnail_AasRepository	GET	/shells/{aasIdentifier}/asset-information/thumbnail	READ	"/shells/*"
PutThumbnail_AasRepository	PUT	/shells/{aasIdentifier}/asset-information/thumbnail	UPDATE	"/shells/*"
DeleteThumbnail_AasRepository	DELETE	/shells/{aasIdentifier}/asset-information/thumbnail	DELETE	"/shells/*"
GetAllSubmodelReferences_AasRepository	GET	/shells/{aasIdentifier}/submodel-refs	VIEW	"/shells/*"
PostSubmodelReference_AasRepository	POST	/shells/{aasIdentifier}/submodel-refs	CREATE	"/shells/*"
DeleteSubmodelReference_AasRepository	DELETE	/shells/{aasIdentifier}/submodel-refs/{submodelIdentifier}	DELETE	"/shells/*"
GetSubmodelById_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}	READ	"/shells/*"
PutSubmodelById_AasRepository	PUT	/shells/{aasIdentifier}/submodels/{submodelIdentifier}	CREATE or UPDATE	"/shells/*"
PatchSubmodel_AasRepository	PATCH	/shells/{aasIdentifier}/submodels/{submodelIdentifier}	UPDATE	"/shells/*"
DeleteSubmodelById_AasRepository	DELETE	/shells/{aasIdentifier}/submodels/{submodelIdentifier}	DELETE	"/shells/*"
GetSubmodelById-Metadata_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/\$metadata	READ	"/shells/*"
PatchSubmodelById-Metadata_AasRepository	PATCH	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/\$metadata	UPDATE	"/shells/*"
GetSubmodelById-ValueOnly_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/\$value	READ	"/shells/*"
PatchSubmodelById-ValueOnly_AasRepository	PATCH	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/\$value	UPDATE	"/shells/*"
GetSubmodelById-Reference_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/\$reference	VIEW	"/shells/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
GetSubmodelById-Path_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/\$path	VIEW	"/shells/*"
GetAllSubmodelElements_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements	READ	"/shells/*"
PostSubmodelElement_AasRepository	POST	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements	CREATE	"/shells/*"
GetAllSubmodelElements-Metadata_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/\$metadata	READ	"/shells/*"
GetAllSubmodelElements-ValueOnly_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/\$value	READ	"/shells/*"
GetAllSubmodelElements-Reference_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/\$reference	VIEW	"/shells/*"
GetAllSubmodelElements-Path_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/\$path	VIEW	"/shells/*"
GetSubmodelElementByPath_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	READ	"/shells/*"
PostSubmodelElementByPath_AasRepository	POST	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	CREATE	"/shells/*"
PutSubmodelElementByPath_AasRepository	PUT	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	CREATE or UPDATE	"/shells/*"
PatchSubmodelElementValueByPath_AasRepository	PATCH	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	UPDATE	"/shells/*"
DeleteSubmodelElementByPath_AasRepository	DELETE	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	DELETE	"/shells/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
GetSubmodelElementByPath-Metadata_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$metadata	READ	"/shells/*"
PatchSubmodelElementValueByPath-Metadata	PATCH	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$metadata	UPDATE	"/shells/*"
GetSubmodelElementByPath-ValueOnly_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$value	READ	"/shells/*"
PatchSubmodelElementValueByPath-ValueOnly	PATCH	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$value	UPDATE	"/shells/*"
GetSubmodelElementByPath-Reference_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$reference	VIEW	"/shells/*"
GetSubmodelElementByPath-Path_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$path	VIEW	"/shells/*"
GetFileByPath_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/attachment	READ	"/shells/*"
PutFileByPath_AasRepository	PUT	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/attachment	UPDATE	"/shells/*"
DeleteFileByPath_AasRepository	DELETE	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/attachment	DELETE	"/shells/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
InvokeOperation_AasRepository	POST	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/invoke	EXECUTE	"/shells/*"
InvokeOperation-ValueOnly_AasRepository	POST	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/invoke/\$value	EXECUTE	"/shells/*"
InvokeOperationAsync_AasRepository	POST	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/invoke-async	EXECUTE	"/shells/*"
InvokeOperationAsync-ValueOnly_AasRepository	POST	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/invoke-async/\$value	EXECUTE	"/shells/*"
GetOperationAsyncStatus_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/operation-status/{handleId}	EXECUTE	"/shells/*"
GetOperationAsyncResult_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/operation-results/{handleId}	EXECUTE	"/shells/*"
GetOperationAsyncResult-ValueOnly_AasRepository	GET	/shells/{aasIdentifier}/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/operation-results/{handleId}/\$value	EXECUTE	"/shells/*"
QueryAssetAdministration Shells	POST	/query/shells	READ	"/query/shells"
GetAllSubmodels	GET	/submodels	READ	"/submodels"
PostSubmodel	POST	/submodels	CREATE	"/submodels"
GetAllSubmodelsRecentChanges	GET	/submodels/\$recent-changes	READ	"/submodels/\$recent-changes"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
GetAllSubmodels-Metadata	GET	/submodels/\$metadata	READ	"/submodels/\$metadata"
GetAllSubmodels-ValueOnly	GET	/submodels/\$value	READ	"/submodels/\$value"
GetAllSubmodels-Reference	GET	/submodels/\$reference	VIEW	"/submodels/\$reference"
GetAllSubmodels-Path	GET	/submodels/\$path	VIEW	"/submodels/\$path"
GetSubmodelById	GET	/submodels/{submodelIdentifier}	READ	"/submodels/*"
PutSubmodelById	PUT	/submodels/{submodelIdentifier}	CREATE or UPDATE	"/submodels/*"
PatchSubmodelById	PATCH	/submodels/{submodelIdentifier}	UPDATE	"/submodels/*"
DeleteSubmodelById	DELETE	/submodels/{submodelIdentifier}	DELETE	"/submodels/*"
GetSubmodelVersionById AndDate	GET	/submodels/{submodelIdentifier}/\$history	READ	"/submodels/*"
GetSubmodelById-Metadata	GET	/submodels/{submodelIdentifier}/\$metadata	READ	"/submodels/*"
PatchSubmodelById-Metadata	PATCH	/submodels/{submodelIdentifier}/\$metadata	UPDATE	"/submodels/*"
GetSubmodelById-ValueOnly	GET	/submodels/{submodelIdentifier}/\$value	READ	"/submodels/*"
PatchSubmodelById-ValueOnly	PATCH	/submodels/{submodelIdentifier}/\$value	UPDATE	"/submodels/*"
GetSubmodelById-Reference	GET	/submodels/{submodelIdentifier}/\$reference	VIEW	"/submodels/*"
GetSubmodelByIdSigned	GET	/submodels/{submodelIdentifier}/\$signed	READ	"/submodels/*"
GetSubmodelById-Path	GET	/submodels/{submodelIdentifier}/\$path	VIEW	"/submodels/*"
GetAllSubmodelElements_SubmodelRepository	GET	/submodels/{submodelIdentifier}/submodel-elements	READ	"/submodels/*"
PostSubmodelElement_SubmodelRepo	POST	/submodels/{submodelIdentifier}/submodel-elements	CREATE	"/submodels/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
GetAllSubmodelElements-Metadata_SubmodelRepo	GET	/submodels/{submodelIdentifier}/submodel-elements/\$metadata	READ	"/submodels/*"
GetAllSubmodelElements-ValueOnly_SubmodelRepo	GET	/submodels/{submodelIdentifier}/submodel-elements/\$value	READ	"/submodels/*"
GetAllSubmodelElements-Reference_SubmodelRepo	GET	/submodels/{submodelIdentifier}/submodel-elements/\$reference	VIEW	"/submodels/*"
GetAllSubmodelElements-Path_SubmodelRepo	GET	/submodels/{submodelIdentifier}/submodel-elements/\$path	VIEW	"/submodels/*"
GetSubmodelElementByPath_SubmodelRepo	GET	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	READ	"/submodels/*"
PostSubmodelElementByPath_SubmodelRepo	POST	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	CREATE	"/submodels/*"
PutSubmodelElementByPath_SubmodelRepo	PUT	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	CREATE or UPDATE	"/submodels/*"
PatchSubmodelElementByPath_SubmodelRepo	PATCH	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	UPDATE	"/submodels/*"
DeleteSubmodelElementByPath_SubmodelRepo	DELETE	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}	DELETE	"/submodels/*"
GetSubmodelElementByPath-Metadata_SubmodelRepo	GET	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$metadata	READ	"/submodels/*"
PatchSubmodelElementByPath-Metadata_SubmodelRepo	PATCH	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$metadata	UPDATE	"/submodels/*"
GetSubmodelElementByPath-ValueOnly_SubmodelRepo	GET	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$value	READ	"/submodels/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
PatchSubmodelElementByPath-ValueOnly_SubmodelRepo	PATCH	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$value	UPDATE	"/submodels/*"
GetSubmodelElementByPath-Reference_SubmodelRepo	GET	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$reference	VIEW	"/submodels/*"
GetSubmodelElementByPath-Path_SubmodelRepo	GET	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/\$path	VIEW	"/submodels/*"
GetFileByPath_SubmodelRepo	GET	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/attachment	READ	"/submodels/*"
PutFileByPath_SubmodelRepo	PUT	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/attachment	UPDATE	"/submodels/*"
DeleteFileByPath_SubmodelRepo	DELETE	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/attachment	DELETE	"/submodels/*"
InvokeOperation_SubmodelRepo	POST	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/invoke	EXECUTE	"/submodels/*"
InvokeOperation-ValueOnly_SubmodelRepository	POST	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/invoke/\$value	EXECUTE	"/submodels/*"
InvokeOperationAsync_SubmodelRepository	POST	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/invoke-async	EXECUTE	"/submodels/*"
InvokeOperationAsync-ValueOnly_SubmodelRepository	POST	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/invoke-async/\$value	EXECUTE	"/submodels/*"
GetOperationAsyncStatus_SubmodelRepository	GET	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/operation-status/{handleId}	EXECUTE	"/submodels/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
GetOperationAsyncResult_SubmodelRepository	GET	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/operation-results/{handleId}	EXECUTE	"/submodels/*"
GetOperationAsyncResult-ValueOnly_SubmodelRepository	GET	/submodels/{submodelIdentifier}/submodel-elements/{idShortPath}/operation-results/{handleId}/\$value	EXECUTE	"/submodels/*"
QuerySubmodels	POST	/query/submodels	READ	"/query/submodels"
GetAllConceptDescriptions	GET	/concept-descriptions	READ	"/concept-descriptions"
PostConceptDescription	POST	/concept-descriptions	CREATE	"/concept-descriptions"
GetAllConceptDescriptionsRecentChanges	GET	/concept-descriptions/\$recent-changes	READ	"/concept-descriptions/\$recent-changes"
GetConceptDescriptionById	GET	/concept-descriptions/{cdIdentifier}	READ	"/concept-descriptions/*"
PutConceptDescriptionById	PUT	/concept-descriptions/{cdIdentifier}	CREATE or UPDATE	"/concept-descriptions/*"
DeleteConceptDescriptionById	DELETE	/concept-descriptions/{cdIdentifier}	DELETE	"/concept-descriptions/*"
GetConceptDescriptionByIdSigned	GET	/concept-descriptions/{cdIdentifier}/\$signed	READ	"/concept-descriptions/*"
QueryConceptDescriptions	POST	/query/concept-descriptions	READ	"/query/concept-descriptions"
GetAllAssetAdministrationShellDescriptors	GET	/shell-descriptors	VIEW	"/shell-descriptors"
PostAssetAdministrationShellDescriptor	POST	/shell-descriptors	CREATE	"/shell-descriptors"
GetAssetAdministrationShellDescriptorById	GET	/shell-descriptors/{aasIdentifier}	VIEW	"/shell-descriptors/*"
PutAssetAdministrationShellDescriptorById	PUT	/shell-descriptors/{aasIdentifier}	CREATE or UPDATE	"/shell-descriptors/*"
DeleteAssetAdministrationShellDescriptorById	DELETE	/shell-descriptors/{aasIdentifier}	DELETE	"/shell-descriptors/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
GetAllSubmodelDescriptorsThroughSuperpath	GET	/shell-descriptors/{aasIdentifier}/submodel-descriptors	VIEW	"/shell-descriptors/*"
PostSubmodelDescriptorThroughSuperpath	POST	/shell-descriptors/{aasIdentifier}/submodel-descriptors	CREATE	"/shell-descriptors/*"
GetSubmodelDescriptorByIdThroughSuperpath	GET	/shell-descriptors/{aasIdentifier}/submodel-descriptors/{submodelIdentifier}	VIEW	"/shell-descriptors/*"
PutSubmodelDescriptorByIdThroughSuperpath	PUT	/shell-descriptors/{aasIdentifier}/submodel-descriptors/{submodelIdentifier}	CREATE or UPDATE	"/shell-descriptors/*"
DeleteSubmodelDescriptorByIdThroughSuperpath	DELETE	/shell-descriptors/{aasIdentifier}/submodel-descriptors/{submodelIdentifier}	DELETE	"/shell-descriptors/*"
QueryAssetAdministrationShellDescriptors	POST	/query/shell-descriptors	VIEW	"/query/shell-descriptors"
BulkPostAssetAdministrationShellDescriptor	POST	/bulk/shell-descriptors	CREATE	"/bulk/shell-descriptors"
BulkPutAssetAdministrationShellDescriptorById	PUT	/bulk/shell-descriptors	CREATE or UPDATE	"/bulk/shell-descriptors"
BulkDeleteAssetAdministrationShellDescriptorById	DELETE	/bulk/shell-descriptors	DELETE	"/bulk/shell-descriptors"
BulkGetAsyncStatus	GET	/bulk/status/{handleId}	originating RIGHT	"/bulk/status/*"
BulkGetAsyncResult	GET	/bulk/result/{handleId}	originating RIGHT	"/bulk/result/*"
GetAllSubmodelDescriptors	GET	/submodel-descriptors	VIEW	"/submodel-descriptors"
PostSubmodelDescriptor	POST	/submodel-descriptors	CREATE	"/submodel-descriptors"
GetSubmodelDescriptorById	GET	/submodel-descriptors/{submodelIdentifier}	VIEW	"/submodel-descriptors/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
PutSubmodelDescriptorById	PUT	/submodel-descriptors/{submodelIdentifier}	CREATE or UPDATE	"/submodel-descriptors/*"
DeleteSubmodelDescriptorById	DELETE	/submodel-descriptors/{submodelIdentifier}	DELETE	"/submodel-descriptors/*"
QuerySubmodelDescriptors	POST	/query/submodel-descriptors	VIEW	"/query/submodel-descriptors"
BulkPostSubmodel	POST	/bulk/submodel-descriptors	CREATE	"/bulk/submodel-descriptors"
BulkPutSubmodelDescriptorsById	PUT	/bulk/submodel-descriptors	CREATE or UPDATE	"/bulk/submodel-descriptors"
BulkDeleteSubmodelDescriptorsById	DELETE	/bulk/submodel-descriptors	DELETE	"/bulk/submodel-descriptors"
GetAllAssetAdministrationShellIdsByAssetLink	GET	/lookup/shells	VIEW	"/lookup/shells"
SearchAllAssetAdministrationShellIdsByAssetLink	POST	/lookup/shellsByAssetLink	VIEW	"/lookup/shellsByAssetLink"
GetAllAssetLinksById	GET	/lookup/shells/{aasIdentifier}	VIEW	"/lookup/shells/*"
PostAllAssetLinksById	POST	/lookup/shells/{aasIdentifier}	CREATE or UPDATE	"/lookup/shells/*"
DeleteAllAssetLinksById	DELETE	/lookup/shells/{aasIdentifier}	DELETE	"/lookup/shells/*"
GetAllAASXPackageIds	GET	/packages	VIEW	"/packages"
PostAASXPackage	POST	/packages	CREATE	"/packages"
GetAASXByPackageId	GET	/packages/{packageId}	READ	"/packages/*"
PutAASXByPackageId	PUT	/packages/{packageId}	CREATE or UPDATE	"/packages/*"
DeleteAASXByPackageId	DELETE	/packages/{packageId}	DELETE	"/packages/*"
PostAsyncAASXPackage	POST	/packages-async	CREATE	"/packages-async"
GetAasxAsyncStatus	GET	/packages-async/status/{handleId}	CREATE	"/packages-async/status/*"

Operation Name	HTTP	URL Template	RIGHT	ROUTE literal example
GetAasxAsyncResult	GET	/packages-async/result/{handleId}	CREATE	"/packages-async/result/*"
GenerateSerializationByIds	GET	/serialization	READ	"/serialization"
GetSelfDescription	GET	/description	READ	"/description"

For profiles that expose only a subset of these operations (see [Profiles](#)), only the rows that correspond to supported operations are applicable. Operations that are added in future minor versions **MUST** be appended to this normative table. The indicative mapping between operation verbs and rights in IDTA-01004 may guide that assignment, but does not replace an explicit mapping here.

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 20](#) 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 28](#). 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. [\[10\]](#)

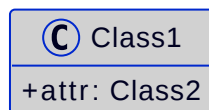


Figure 20. Class

[Figure 21](#) 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*.

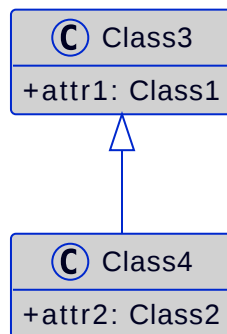


Figure 21. Inheritance/Generalization

[Figure 22](#) 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.

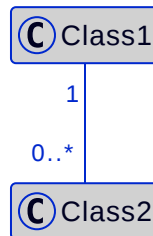


Figure 22. Multiplicity

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 23](#) 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.

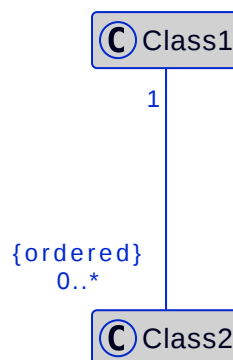


Figure 23. Ordered Multiplicity

[Figure 24](#) 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*.

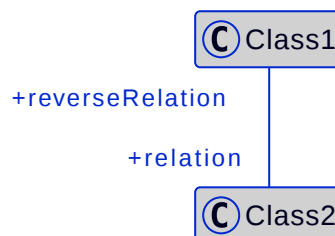


Figure 24. Association

[Figure 25](#) 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'.

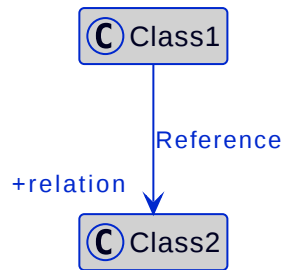


Figure 25. Association

A specialty in [Figure 25](#) 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 26](#) 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 26](#) shows the composition using an association relationship with a filled diamond as composition adornment.



Figure 26. Composition (Composite Aggregation)

[Figure 27](#) 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 27](#) shows the shared aggregation using an association relationship with a hallow diamond as aggregation adornment.



Figure 27. Aggregation

[Figure 28](#) 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.

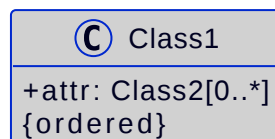


Figure 28. Navigable Attribute Notation for Associations

[Figure 29](#) 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.

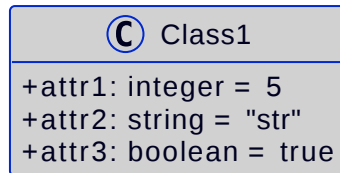


Figure 29. Default Value

[Figure 30](#) 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.

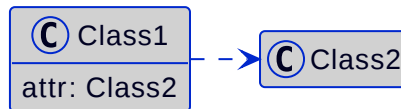


Figure 30. Dependency

[Figure 31](#) 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 31. Abstract Class

[Figure 32](#) shows a package named "Package2". A package is a namespace for its members. In this example, the member belonging to *Package2* is class *Class2*.

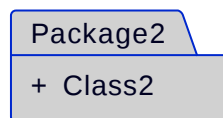


Figure 32. Package

[Figure 33](#) 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>>`.

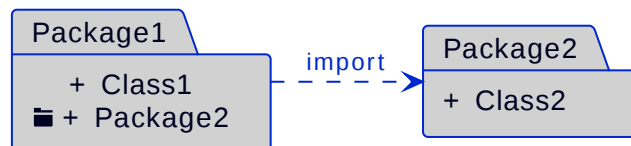


Figure 33. Imported Package

[Figure 34](#) 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.

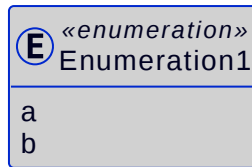


Figure 34. Enumeration

Figure 35 shows how a note can be attached to an element, in this example to class "Class1".



Figure 35. Note

Figure 36 shows how a constraint is attached to an element, in this example to class "Class1".

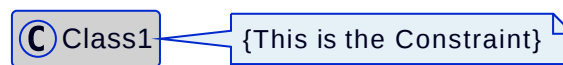


Figure 36. Constraint

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 37](#) 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.

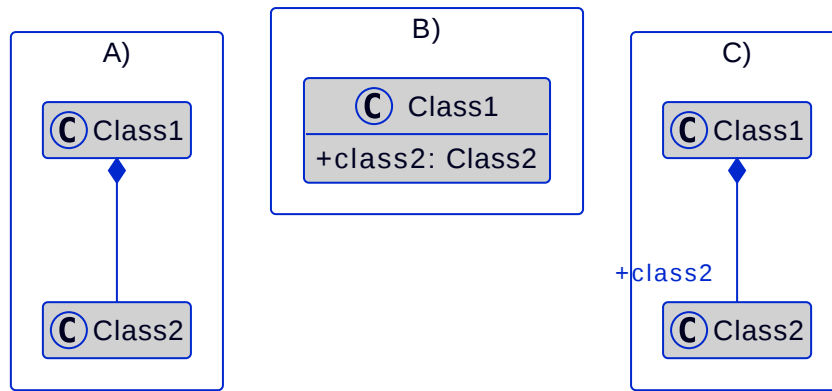


Figure 37. Graphical Representations of Composite Aggregation/Composition

Figure 38 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.

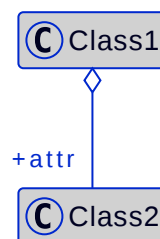


Figure 38. Graphical Representation of Shared Aggregation

Figure 39 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.

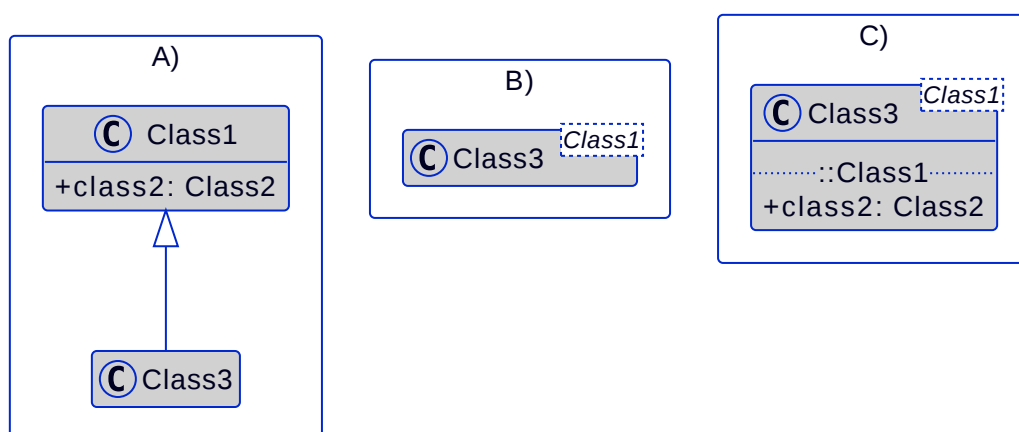


Figure 39. Graphical Representation of Generalization/Inheritance

Figure 40 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 40 that defines the same enumeration but without inheritance.

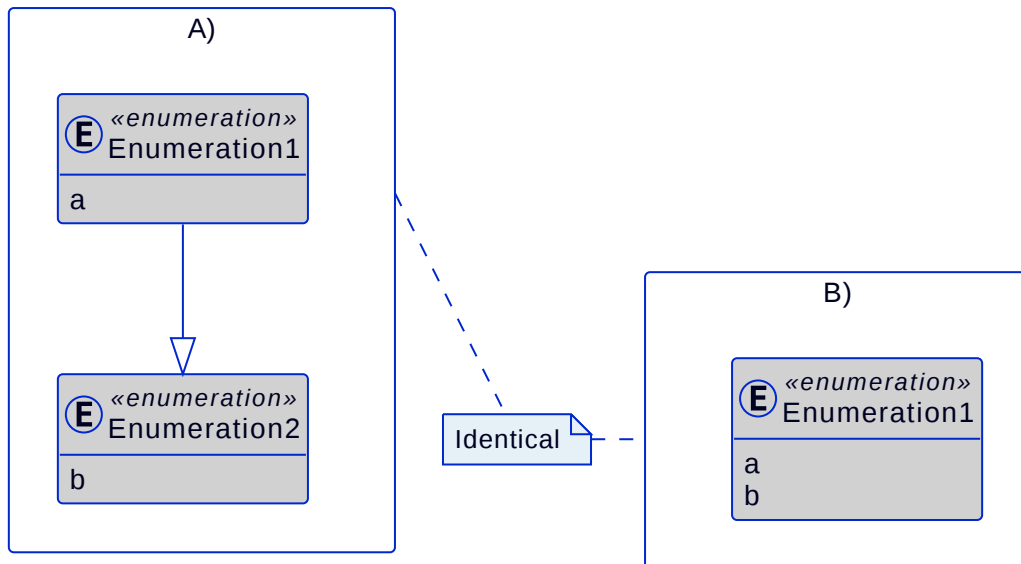


Figure 40. Graphical Representation for Enumeration with Inheritance

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

[Figure 41](#) shows an experimental class, marked by the stereotype "Experimental".

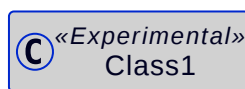


Figure 41. Graphical Representation for Experimental Classes

[Figure 42](#) depicts a deprecated class, which is marked by the stereotype "Deprecated".

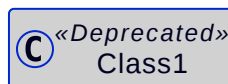


Figure 42. Graphical Representation for Deprecated Elements

[Figure 43](#) shows a class representing a template. It is marked by the stereotype "Template".



Figure 43. Graphical Representation of a Template Class

Conformance Test Corpus

This annex defines an informative but structured set of **acceptance tests** for conforming implementations of the AAS Query Language and the HTTP/REST API defined in this specification. The corpus is intentionally technology-neutral and worded as a set of test cases; an implementation is said to be conformant to a given area if it passes every required test case in that area.

The corpus is aligned with the conformance test corpus in IDTA-01004 Part 4: Security [\[3\]](#). Tests that exercise shared formula grammar productions (logical expressions, comparisons, FieldIdentifiers, value literals, type casts, \$match /\$and/\$or/\$not) are normative for both specifications.

Each test case has the shape:

ID: <spec-id>.<category>.<nnn>
 Purpose: <one-line intent>
 Input: <request / query / rule as JSON or BNF>
 Expected: <HTTP status, result predicate, or behavioural predicate>
 Requirement: MUST | SHOULD

Query Language

ID	Purpose	Expected
QL.parse.001	A syntactically valid query MUST be accepted.	HTTP 200 and a result set. No 400/422.
QL.parse.002	A query with an unknown top-level key MUST be rejected.	HTTP 400 (syntactic) with a machine-readable error pointer.
QL.parse.003	A query whose JSON parses but violates the JSON Schema MUST be rejected.	HTTP 400 with a JSON-pointer to the failing element.
QL.field.001	\$aas#id on an AAS Repository MUST resolve to AAS identifiers.	Result contains only AAS whose id matches.
QL.field.002	\$sm#semanticId.keys[0].value MUST match the first key of the submodel semanticId.	Result filtered on that key only.
QL.field.003	\$sme#value MUST search recursively over all matching SubmodelElements.	Matches occur in nested SubmodelElementCollections / Lists.
QL.field.004	\$aasdesc#... on a non-Registry profile MUST be treated as "not applicable", not as an error.	HTTP 200, empty-match semantics (see Evaluation Semantics).
QL.cast.001	num(str-attribute) with a non-numeric value MUST yield "no match" for that item (not a 500).	HTTP 200, item excluded from the result.
QL.cast.002	dateTime(<field>) with a field value that is not a valid xsd:dateTime MUST yield "no match".	HTTP 200, item excluded.
QL.time.001	A \$timeVal MUST accept hh:mm, hh:mm:ss, hh:mm:ss.fff, and timezone-qualified forms such as hh:mm:ssZ and hh:mm:ss+01:00.	All forms are accepted by the schema validator.
QL.time.002	A \$timeVal with invalid components (e.g. 25:00) MUST be rejected by the schema.	HTTP 400.
QL.dt.001	\$dayOfWeek , \$dayOfMonth , \$month , \$year MUST accept a Value operand (not only a literal).	A query using \$dayOfWeek(\$field: "\$sm#...") passes schema validation.
QL.suppl.001	\$sm#supplementalSemanticIds[].keys[0].value \$eq "..." MUST match on any listed supplementalSemanticId.	Items whose supplementalSemanticIds contain the value are returned.
QL.logic.001	\$and with fewer than two operands MUST be rejected by the schema.	HTTP 400.

ID	Purpose	Expected
QL.logic.002	\$or with fewer than two operands MUST be rejected by the schema.	HTTP 400.

HTTP/REST API

ID	Purpose	Expected
HTTP.auth.001	A request without credentials on a protected resource MUST receive 401.	Status 401 Unauthorized.
HTTP.auth.002	A request whose access rule evaluation yields "deny" MUST receive 403.	Status 403 Forbidden, no payload leakage.
HTTP.put.001	PUT on a client-addressable resource that does not yet exist MUST require the RIGHT CREATE.	Status 201 Created if rule allows, else 403.
HTTP.put.002	PUT on an existing client-addressable resource MUST require the RIGHT UPDATE.	Status 200 OK if rule allows, else 403.
HTTP.invoke.001	POST /.../operations/{idShortPath}/invoke MUST require the RIGHT EXECUTE.	Status 200 with result envelope or 403.
HTTP.file.001	GET .../attachment for a File SubmodelElement MUST require the RIGHT READ on the File element.	Status 200 with binary body or 403.
HTTP.query.001	POST /query MUST reject payloads that fail the Query JSON Schema with 400.	Status 400, JSON-pointer in body.
HTTP.discovery.001	Discovery endpoints MUST honour the operation-to-RIGHT mapping of this spec (Operation to RIGHT Mapping).	Right resolved according to mapping table, not ad-hoc.

Error handling and edge cases

ID	Purpose	Expected
ERR.parse.001	A request body that is not valid JSON MUST receive 400.	Status 400 Bad Request.
ERR.schema.001	A request body that is valid JSON but violates the schema MUST receive 400.	Status 400 Bad Request.
ERR.cast.001	A runtime cast error (num("abc")) MUST NOT yield 500; the item is excluded from the result.	Status 200, item missing from response.
ERR.field.001	An unknown <AttributeDeclaration> MUST be rejected at schema level (400), not at runtime.	Status 400.
ERR.profile.001	A rule/query referencing a FieldIdentifier that is not applicable to the current profile is treated as "not applicable" (no match, no error).	Status 200.

- [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. V3.1.3 to V3.2

Major Changes:

- Add new subchapter on the discovery of endpoints of AAS services based on ID Link. ([#535](#))
- Add new operations `GetAssetAdministrationShellVersionByIdAndDate` and `GetSubmodelVersionByIdAndDate` to retrieve a specific version of an AAS or Submodel from a given point in time. ([#555](#))
- Added a new profile for the AAS Repository Service Specification, the "Identifiable Profile", which only includes operations that require an identifier as input parameter. ([#496](#))
- New API operations and profiles for signing, including Asset Administration Shell Repository Service Specification - Signature Profile SSP-004, Submodel Repository Service Specification - Signature Profile SSP-003, and Concept Description Repository Service Specification - Signature Profile SSP-004 ([#518](#))
- Add the asynchronous AASX File Server Service Specification ([#347](#))
- Added: Query Filter and FragmentFieldIdentifiers ([#517](#))
- Add new section in the annex on the usage of the AAS Services for implementing a digital product passport (DPP) based on EN 18222 and the relevant API operations for DPP interactions. This section also includes an overview of how to expose the DPP interactions through a REST API that follows the specifications of the AAS Services.
- Added new experimental operations `GetAllAssetAdministrationShellsRecentChanges`, `GetAllSubmodelsRecentChanges`, `GetAllConceptDescriptionsRecentChanges` which rely on the `createdAt` and `updatedAt` attributes of `AdministrativeInformation` introduced in V3.2 of the metamodel. ([#551](#))
- Added new input parameter for `GetAllAssetAdministrationShells` and `GetAllSubmodels` and `GetAllConceptDescriptions`
- Added new input parameters to `GetAllAssetAdministrationShellDescriptors` and `GetAllSubmodelDescriptors` for filtering for recently changed AAS or Submodels, resp. ([#346](#))
- Switching from SwaggerHub to ([GitHub Pages](#)) for the maintenance of OpenAPI files. Versions until v3.1.3 are still available on SwaggerHub, but all new versions will be published on GitHub Pages.

Minor Changes:

- fix: Required the binary file part for AASX package uploads and removed the redundant `fileName` multipart field; an optional source filename is conveyed by the standard `filename` parameter of the file part.
- fix: Corrected `USEATTRIBUTES` in the access-rule JSON Schema to accept a list, so one ACL can reference multiple named attribute groups as defined by the BNF.
- fix: Marked the Query request body as required in all repository and registry query profiles, matching the mandatory query input in the interface definitions.
- fix: Marked bulk DELETE request bodies as required and corrected the reusable `handleId` parameter references in the asynchronous registry bulk profiles.
- fix: Corrected the AAS Service asynchronous invocation path from `/invoke-asnyc` to `/invoke-async` and removed undeclared `aasIdentifier` parameters from submodel operation routes.
- fix: Changed JSON Schema validation from `oneOf` to `anyOf` where BNF grammar allows mixed inline and named groups in concatenation scenarios.

- docs: Aligned prose tables in Query Language specification with BNF grammar and JSON Schema patterns for cast operators, field identifiers, and submodel element limitations.
- docs: Added normative FieldIdentifier applicability table per profile to clarify which Query and Access Rule prefixes apply to specific service specification families.
- docs: Added normative Query Evaluation Semantics clause to standardize query behavior across implementations, including error handling, field resolution, and list-valued matching.
- docs: Added normative Operation-to-RIGHT mapping table documenting the correspondence between HTTP/REST API operations and IDTA-01004 rightsEnum values.
- docs: Added cross-spec terminology matrix reference to align Metamodel classes, API path segments, and Query/Access-Rule prefixes across specifications.
- docs: Added cross-specification alignment matrix to document API compatibility with IDTA-01001 (Metamodel v3.2 instead of v3.1), DTA-01003-a, IDTA-01003-b, IDTA-01004 (Security v3.1), and IDTA-01005 (from v3.1 to v3.2).
- docs: Established IDTA-01002 as authoritative source for formula grammar and JSON Schema shared between API and Security specifications.
- fix: Aligned Query Language BNF and JSON Schema with IDTA-01004 Security access-rule schema, including wildcard array indexes, stricter object/reference identifiers, and artifact validation tests.
- fix: Aligned dateTimeLiteralPattern and timeLiteralPattern with the normative BNF, including HH:MM forms, optional seconds and fractions, timezone-qualified values, component ranges, and valid calendar dates.
- docs: Added informative Conformance Test Corpus annex providing technology-neutral test cases for Query Language, HTTP/REST API, and error handling.
- Added the ServerNotImplemented status code to the [Status Codes](#). Note that the HTTP API mentioned this status code already, therefore, no change in OpenAPI or other related files. ([#499](#))
- Removed the GetAllAssetAdministrationShellDescriptorsByAssetType operation from the OpenAPI files for the Asset Administration Shell Registry Service Specification, as it was not intended to be included in the first place and is not implemented by any known implementation. ([#515](#))
- Changing the serialization format for AASX packages from "application/asset-administration-shell-package+xml" to "application/aas+zip" in the OpenAPI files for the Asset Administration Shell Repository Service Specification and the Submodel Repository Service Specification. This change is necessary to align with the actual content type used by implementations and to avoid confusion, as the previous content type was not registered and could be misinterpreted as an XML-based format. ([#538](#))
- Editorial fixes in Table 2 of the Query Language specification ([#507](#))
- Removed the GetAllAssetAdministrationShellDescriptorsByAssetType operation from the OpenAPI files for the Asset Administration Shell Registry Service Specification, as it was not intended to be included in the first place and is not implemented by any known implementation. ([#515](#))
- Editorial: Remove links to SwaggerHub from the HTTP API section, as the links are maintained in the service specifications and profiles sections. ([#475](#))
- Added supplementalSemanticIds to the AAS Query Language specification ([#524](#))
- fix: Required the binary file part for thumbnail and attachment uploads and removed the redundant fileName multipart field; optional source filenames are conveyed by the standard filename parameter of each file part.
- Fixed examples for endpointProtocolVersion ([#599](#))
- Editorial changes w.r.t. structuring of sections
- Corrected examples for the Query Language specification and fixed the grammar and JSON Schema, which did not correctly reflect the intended semantics for optional Reference constraints.
- Fixing the Backus-Naur-Form for describing grammar for INTERFACE in endpoint. ([#472](#))
- fix: Extended query and access-rule identifier literals to the full AAS Identifier character range and added escaping for quotation marks and reverse solidus characters.

Interface Changes w.r.t. V3.1.3 to V3.2

BW C	Interface	Kind of Change	Comment
	AASX File Server Interface	Extended	new API operations for asynchronous creation: PostAsyncAASXPackage, GetAasxAsyncStatus, GetAasxAsyncResult (#347)
	Asset Administration Shell Repository Interface	Extended	new API operation for signing: GetAssetAdministrationShellByIdSigned (#518)
	Asset Administration Shell Repository Interface	Extended	new API operation for retrieving changed objects: GetAllAssetAdministrationShellsRecentChanges
	Asset Administration Shell Repository Interface	Extended	extended API operation GetAllAssetAdministrationShells with additional input parameters
	Asset Administration Shell Registry Interface	Extended	extended API operation for retrieving changed objects and objects based on specific Asset IDs
	Concept Description Repository Interface	Extended	new API operation for signing: GetConceptDescriptionByIdSigned (#518)
	Concept Description Repository Interface	Extended	new API operation for retrieving changed objects: GetAllConceptDescriptionsRecentChanges
	Concept Description Repository Interface	Extended	extended API operation GetAllConceptDescriptions with additional input parameters
	Submodel Repository Interface	Extended	new API operation for signing: GetSubmodelByIdSigned (#518)
	Submodel Repository Interface	Extended	new API operation for retrieving changed objects: GetAllSubmodelsRecentChanges
	Submodel Repository Interface	Extended	extended API operation GetAllSubmodels with additional input parameters
	Submodel Registry Interface	Extended	extended API operation for retrieving changed objects

Operation Changes w.r.t. V3.1.3 to V3.2

Operation	Kind of Change	Comment
GetAssetAdministrationShellVersionByIdAndDate	New	new API-Operation for AAS Repository interface to retrieve a specific version of an AAS from a given point in time. (#555)
GetSubmodelVersionByIdAndDate	New	new API-Operation for Submodel Repository interface to retrieve a specific version of a Submodel from a given point in time. (#555)
GenerateSerializationByIds	change	Added a note to clarify the intended behavior for transitively referenced concept descriptions. (#510)

Operation	Kind of Change	Comment
GetAasxAsyncResult	new	New API Operation for the asynchronous AASX File Server Service Specification.. (#347)
GetAasxAsyncStatus	new	New API Operation for the asynchronous AASX File Server Service Specification (#347)
GetAssetAdministrationShellByIdSigned	new	(#518)
GetConceptDescriptionByIdSigned	new	(#518)
GetSubmodelByIdSigned	new	(#518)
PostAsyncAASXPackage	new	New API Operation for the asynchronous AASX File Server Service Specification (#347)
PutAssetAdministrationShellById	change	Added the missing id input parameter and a note explaining the behavior in case the AAS identifier does not match the value of id. (#506)
GetAllAssetAdministrationShellDescriptors	change	added new input parameters createdFrom and updatedFrom and as well as key and keyIdentifier for specific Asset IDs.
GetAllSubmodelDescriptors	change	added new input parameters createdFrom and updatedFrom for recent changes
GetAllAssetAdministrationShells	change	added new input parameters createdFrom and updatedFrom for recent changes as well as key and keyIdentifier for specific Asset IDs (was already contained in openAPI).
GetAllSubmodels	change	added new input parameters createdFrom and updatedFrom for recent changes
GetAllConceptDescriptions	change	added new input parameters createdFrom and updatedFrom for recent changes
GetAllAssetAdministrationShellsRecentChanges	New	new experimental API-Operation for AAS Repository interface to retrieve recently changed or created AAS with an optimized return object
GetAllSubmodelsRecentChanges	New	new experimental API-Operation for Submodel Repository interface to retrieve recently changed or created Submodels with an optimized return object
GetAllConceptDescriptionsRecentChanges	New	new experimental API-Operation for Concept Description Repository interface to retrieve recently changed or created Concept Descriptions with an optimized return object

Changes w.r.t. V3.1.2 to V3.1.3

Major Changes:

- Fixed the response payload type of PutSubmodelById in the Asset Administration Shell Repository OpenAPI files: status code 201 now returns a Submodel instead of a Reference. ([#601](#))

Minor Changes:

- fix: Changed JSON Schema validation from `oneOf` to `anyOf` where BNF grammar allows mixed inline and named groups in concatenation scenarios.
- fix: Harmonized `SecurityQueryFilter` between API and Security specifications and added missing `FILTERLIST` construct to the BNF grammar and JSON Schema.
- fix: Aligned `idShort` pattern in BNF grammar and JSON Schema with Metamodel Constraint AASd-002 to correctly enforce minimum two characters and prevent trailing hyphens.
- fix: Aligned date/time extraction functions (`$dayOfWeek`, `$dayOfMonth`, `$month`, `$year`) in JSON Schema to accept all `dateTimeOperand` types per BNF grammar, and widened `timeLiteralPattern` to support fractional seconds.
- Updated the response contract of `PostAllAssetLinksById` in Discovery OpenAPI files by removing status code 404 for `POST /lookup/shells/{aasIdentifier}` to align with create-or-replace semantics. ([#595](#))
- Updated the response contract of `PostAllAssetLinksById` in Discovery OpenAPI files by removing status code 409 for `POST /lookup/shells/{aasIdentifier}` to align with replace/update behavior on repeated payloads. ([#596](#))
- Updated the response contract of `GetAllSubmodelDescriptors` in Submodel Registry OpenAPI files by removing status code 404 for `GET /submodel-descriptors`, so an empty collection is consistently represented by 200 with an empty result. ([#597](#))
- Submodel Service Specification – Read Profile had the wrong identifier <https://admin-shell.io/aas/API/3/1/AssetAdministrationShellServiceSpecification/SSP-002> instead of <https://admin-shell.io/aas/API/3/1/SubmodelServiceSpecification/SSP-002>
- fix: Replaced OpenAPI references to `ReferenceValue` with `Reference` to include the `referredSemanticId` field and align with the specification text that mandates Normal serialization format for References in valueOnly JSON. ([#521](#))

Changes w.r.t. V3.1.1 to V3.1.2

Major Changes:

- none

Minor Changes:

- fix: Wrong `ServiceSpecificationProfileEnum` values for v3.0 profiles. ([#526](#))
- removed: Remove `TREE`. ([#537](#))
- fix: fixed `FILTER` object in json schema & fixed inconsistencies in BNF ([#547](#))
- fix: naming inconsistencies inside BNF and json schema ([#547](#))
- fix: aligned BNF and json schema to api spec ([#547](#))
- fix: BNF did not compile ([#62](#))
- fix: BNF used incorrectly plural for `USEACL` & `USEFORMULA` ([#61](#))
- fix: Regex `FieldIdentifier` was incorrect ([#60](#))
- fix: Incorrect `FieldIdentifier` AAS Submodel References ([#59](#))
- Remove the requirement to have `"clientTimeoutDuration"` for asynchronous operation invocations in the OpenAPI files. This is a change in the OpenAPI files only, as the `"clientTimeoutDuration"` parameter was already optional in the definition of the `OperationRequestAsync` class in the OpenAPI schema. ([#563](#))
- Changed `SupplementalSemanticId` to `SupplementalSemanticIds` in the OpenAPI files to align with the definition in the OpenAPI schema. ([#553](#))
- Adjusted the return type for the operations `OperationRequestValueOnly` and `OperationResultValueOnly` from the incorrect `SubmodelElementValue` to `ValueOnly`. No changes in the already correct OpenAPI files. ([Part 1#663](#))

Changes w.r.t. V3.1 to V3.1.1

Major Changes:

- Extending the ServiceSpecificationProfileEnum to include all v3.1 profiles and correcting the example ([#473](#))
- Make the AssetLink properties "name" and "value" mandatory in the OpenAPI files. This is a bugfix, as the AssetLink class already defines these properties as mandatory. ([#471](#))
- changed: Push all semantic IDs to V3.1 ([#490](#))

Minor Changes:

- changed: Removed second regex pattern from Blob/contentType, File/contentType, File/value, Referable/idShort, Resource/path, and Resource/contentType attributes in the Part1-MetaModel-Schemas OpenAPI file. The first regex pattern is sufficient to cover all valid content types. While this is not directly fixing a bug, many downstream tools report issues, e.g. during code generation, when two regex patterns are defined for the same attribute. ([#481](#))
- changed: Introduced 204 response code for DeleteThumbnail operation in the OpenAPI files. The 200 response code is deprecated and should not be used anymore. ([#469](#))
- changed: Upgraded semanticIds of ProtocolInformation, SecurityAttributeObject, SecurityTypeEnum, and the items of the SecurityTypeEnum to V3.1 ([#470](#))
- new: Add 404 'Not Found' to the PutSubmodelDescriptorByIdThroughSuperpath operation in the OpenAPI file for the AssetAdministrationShellRegistryServiceSpecification/SSP-001 profile.

Operation Changes w.r.t. V3.1 to V3.1.1

Operation	Kind of Change	Comment
PostAllAssetLinksById	change	Corrected description of PostAllAssetLinksById operation to clarify that it creates or replaces all asset links associated to the Asset Administration Shell to edit discoverable content.

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

B W C	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)
	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

B W C	Interface	Kind of Change	Comment
	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
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

Operation	Kind of Change	Comment
PutBulkSubmodelDescriptorsByld	new	new API-Operation for Submodel Registry Interface
DeleteBulkSubmodelDescriptorsByld	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 27. 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)

Nc	V3.1 Change w.r.t. V3.0	Comment
	ProtocolInformation/securityAttributes	Changed securityAttributes from mandatory to optional. OpenAPI remains unchanged. (#384)

Template 28. 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 resolvment 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

BW C	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

BW C	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/3/2/RepositoryServiceSpecification/SSP-001 and https://admin-shell.io/aas/API/3/2/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"
	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

B W C	Interface Change	Kind of Change	Comment
	Submodel	New	PatchSubmodel and PatchSubmodelElementByPath (PUT to completely replace and PATCH to update content)
	Asset Administration Shell, Submodel, AASX File Server, AAS Repository, Submodel Repository, CD Repository, AAS Registry, Submodel Registry, AAS Basic Discovery	Changed	Add Pagination: GetAllAssetAdministrationShells GetAllAssetAdministrationShellsByAssetId GetAllAssetAdministrationShellsByIdShort GetAllSubmodelReferences GetAllSubmodels GetAllSubmodelsBySemanticId GetAllSubmodelsByIdShort GetAllSubmodelElements GetSubmodelElementByPath GetAllConceptDescriptions GetAllConceptDescriptionsByIdShort GetAllConceptDescriptionsByIsCaseOf GetAllConceptDescriptionsByDataSpecificationReference GetAllAssetAdministrationShellDescriptors GetAllSubmodelDescriptors GetAllAssetAdministrationShellIdsByAssetLink GetAllAASXPackageIds
	Submodel	Changed	SerializationModifier Content as path: \$metadata, \$value, \$reference, \$path
	Asset Administration Shell	New	GetThumbnail, PutThumbnail
	Submodel Repository	New	PatchSubmodelForId was missing

BWC	Interface Change	Kind of Change	Comment
	Registry	New	Add extensions to descriptor
	AssetAdministrationShellDescriptor	New	Add the attributes assetKind and assetType
	SubmodelDescriptor	New	Add supplementalSemanticId
	*	Changed	Rename GetDescriptor to GetDescription
	*	Changed	API versioning with major + minor
	*	New	Profiles
	*	Changed	Clarify service specifications and APIs
	CD Registry	Changed	Renaming parameter 'cdIdentifier' in GetConceptDescriptionById to 'id'. Parameter has not been changed in the HTTP API.

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

BWC	Interface Change	Kind of Change	Comment
	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

BW C	Interface Change	Kind of Change	Comment
*	Submodel	Changed	Removed: GetAllSubmodelElementsByParentPathAndSemanticId, GetAllSubmodelElementsBySemanticId New: PutSubmodel, PostSubmodelElement, PostSubmodelElementByPath
*	Asset Administration Shell Serialization	Changed	Renamed: GetSerializationByIds to GenerateSerializationByIds Removed: GetAASX
	AASX File Server	New	New interface
(*)	Asset Administration Shell Registry	Changed	Renamed: PutAssetAdministrationShellDescriptor to PutAssetAdministrationShellDescriptorById New: PostAssetAdministrationShellDescriptor
(*)	Submodel Registry	Changed	Renamed: PutSubmodelDescriptor to PutSubmodelDescriptorById New: PostSubmodelDescriptor
(*)	Asset Administration Shell Repository	Changed	Renamed: GetAllAssetAdministrationShellsById to GetAssetAdministrationShellById, PutAssetAdministrationShell to PutAssetAdministrationShellById New: PostAssetAdministrationShell
(*)	Submodel Repository	Changed	Renamed: PutSubmodel to PutSubmodelById New: PostSubmodel

BW C	Interface Change	Kind of Change	Comment
(*)	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	

Operation Change Old	Operation Change New	Kind of Change	Comment
GetAllSubmodelElementsByParentPathAndSemanticId		Removed	
GetAllSubmodelElementsBySemanticId		Removed	
GetAASX		Removed	
GetSerializationByIds	GenerateSerializationByIds	Renamed	
	GetAllAASXPackageIds	New	
	GetAASXByPackageId	New	
	PostAASXPackage	New	
	PutAASXByPackageId	New	
	DeleteAASXByPackageId	New	
PutAssetAdministrationShellDescriptor	PutAssetAdministrationShellDescriptorById	Changed	Naming pattern byId
	PostAssetAdministrationDescriptor	New	
PutSubmodelDescriptor	PutSubmodelDescriptorById	Changed	Naming pattern byId
	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

Operation Change Old	Operation Change New	Kind of Change	Comment
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. Specification of the Asset Administration Shell. Part 1: Metamodel. Industrial Digital Twin Association (IDTA). V3.2. Online. Available: <https://industrialdigitaltwin.org/en/content-hub/aasspecifications>
- [2] IDTA-01003-a. Specification of the Asset Administration Shell. Part 3a: Data Specification – IEC 61360. Industrial Digital Twin Association (IDTA). V3.1. Online. Available: <https://industrialdigitaltwin.org/en/content-hub/aasspecifications>
- [3] IDTA-01004. Specification of the Asset Administration Shell. Part 4: Security. Industrial Digital Twin Association (IDTA). V3.1. Online. Available: <https://industrialdigitaltwin.org/en/content-hub/aasspecifications>
- [4] IDTA-01005. Specification of the Asset Administration Shell. Part 5: Package File Format. Industrial Digital Twin Association (IDTA). V3.1. 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/>
- [15] IDTA-01003-b. Specification of the Asset Administration Shell. Part 3b: Measurement Units. V3.0. Industrial Digital Twin Association (IDTA). V3.0. Online. Available: <https://industrialdigitaltwin.org/en/content-hub/aasspecifications>