Specification of the Asset Administration Shell

Part 4: Security

SPECIFICATION

IDTA Number: 01004
Version 3.0.1

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

Version

This is version 3.0.1 of the specification IDTA-01004.

Previous version: 3.0.

Metamodel Versions

This document uses the following parts and versions of the “Specification of the Asset Administration Shell” series:

- IDTA-01001 Part 1: Metamodel in version 3.1 [1]
- IDTA-01002 Part 2: Application Programming Interfaces in version 3.1 [2]
- IDTA-01003-a Part 3a: Data Specification – IEC 61360 in version 3.1 [3]

If there are bugfixes of the parts, these shall be used.

Notice

DOI: https://doi.org/10.62628/IDTA.01004-3-0-1

IDTA Number: IDTA-01004

Version: 3.0.1

July 2025

How to Get in Contact

Contact: IDTA

Sources: GitHub

Feedback:

- new issues, bugs
- Questions and Answers

Editorial Notes

History

This document (release candidate for review) was produced from May 2023 to January 2025 by the Task Force “Security” of the working group "Open Technology" in the Industrial Digital Twin Association (IDTA).

Earlier content of this document was developed in the Plattform Industrie 4.0 working group “Reference Architectures, Standards and Norms“, especially Clause 7 “Security” of the Part 1 specification Version 3.0RC02 and the discussion paper “Secure Download Service” [6].

Versioning

This specification is versioned using Semantic Versioning 2.0.0 (semver) and follows the semver specification [5].

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 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.

Please note missing terms and definitions in the review feedback form.

If available, definitions were taken from IEC 63278-1:2023 and IEC 63278-3.

Access Control

The decision to permit or deny a subject access to system objects (network, data, application, service, etc.)

[SOURCE: NIST SP 800-162]

Usage Control

Enforcement of data usage restrictions on the consumer side after access to data has been granted.

> Note 1: Usage Control is concerned with requirements that pertain to data processing (obligations) rather than data access (provisions).

[SOURCE: IDS-RAM 4 - Usage Control]

accountability

property of a system (including all of its system resources) that ensures the actions of a system entity may be traced uniquely to that entity, which can be held responsible for its actions

[SOURCE: IEC TS 62443-1-1:2009, 3.2.3]

authenticate

verify the identity of a user, user device, or other entity, or the integrity of data stored, transmitted, or otherwise exposed to unauthorized modification in an information system, or to establish the validity of a transmission

[SOURCE: IEC TS 62443-1-1:2009, 3.2.12]

authentication

security measure designed to establish the validity of a transmission, message, or originator, or a means of verifying an individual’s authorization to receive specific categories of information

[SOURCE: IEC TS 62443-1-1:2009, 3.2.13]

authorization

right or permission that is granted to a system entity to access a system resource

[SOURCE: IEC TS 62443-1-1:2009, 3.2.14]

data integrity

property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner

> Note 1 to entry: This term deals with constancy of and confidence in data values, not with the information that the values represent or the trustworthiness of the source of the values.

[SOURCE: IEC TS 62443-1-1:2009, 3.2.38]

Abbreviations

Abbreviation	Description
AAS	Asset Administration Shell
AASX	Package file format for the AAS
ABAC	Attribute Based Access Control
ACL	Access Control List
API	Application Programming Interface
BLOB	Binary Large Object
BNF	Backus Naur Form
DKE	Deutsche Kommission für Elektrotechnik
HTTP	Hypertext Transfer Protocol
ID	Identifier
IDTA	Industrial Digital Twin Association
IDP	Identity Provider
IEC	International Electrotechnical Commission
IRDI	International Registration Data Identifier
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
MIME	Multipurpose Internet Mail Extensions
OAUTH	Open Authorization
ODRL	Open Digital Rights Language
OIDC	OpenID Connect
OPC	Open Packaging Conventions (ECMA-376, ISO/IEC 29500-2)
RDF	Resource Description Framework
REST	Representational State Transfer
RFC	Request for Comment
ROA	Resource Oriented Architecture
SOA	Service Oriented Architecture

Abbreviation	Description
UML	Unified Modeling Language
URI, URL, URN	Uniform Resource Identifier, Locator, Name
VDE	Verband der Elektrotechnik Elektronik Informationstechnik e. V.
VDI	Verein Deutscher Ingenieure e.V.
VDMA	Verband Deutscher Maschinen- und Anlagenbau e.V.
W3C	World Wide Web Consortium
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language
X509	Standard format for public key certificates
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

Metamodel Versions

This document uses the following parts of the “Specification of the Asset Administration Shell” series:

- IDTA 01001 Part 1: Metamodel in version 3.1 [1]
- IDTA 01002 Part 2: REST API in version 3.1 [2]
- IDTA-01003-a Part 3a: Data Specification – IEC 61360 in version 3.1 [3]
- IDTA 01005 Part 5: Package File Format (AASX) in version 3.1 [4]

Scope of this Document

This document specifies the security for the Asset Administration Shell and its submodels, i.e. how to use Access Tokens and how to define Access Rules for Authorization. The signing of submodel data will be specified in a next version of this document.

This document includes the grammar of a technology neutral model, which is used both for HTTP API 3.1 Query Language and for the Access Rules. In addition, a corresponding JSON schema is defined.

Structure of the Document

Clause *Terms and Definitions* lists Terms, Definitions and Abbreviations

Clause *Introduction* gives a detailed introduction to the security topic

Clause *Access Rule Model (normative)* defines the Access Rule Model (normative)

Clause *Summary and Outlook* provides a summary and outlook

Annex Backus-Naur-Form defines the grammar language used in the specification

Annex Examples of Access Rules in text serialization contains Examples of Access Rules in text serialization

Annex Examples of Access Rules in JSON serialization contains Examples of Access Rules in JSON serialization

Introduction

This document explains the security of the Asset Administration Shell.

This specification defines a model for access rules. Such model targets 2 purposes.

Purpose 1 is to provide guidelines for implementations.

Purpose 2 is to address the definition and exchange of access rules. Such exchange of access rules is very useful for complex assets like robot or machine. The supplier of an asset may already supply a set of access rules, which the operator of the asset may simply adjust then. In case of switching between AAS implementations, an exchange of access rules is useful.

AAS implementations may decide, to which extent the access rule model is used. The access rule model can be used as a guideline which shows, what a client application will expect. If an AAS implementation allows the exchange of access rules, it should follow the specification of the access rule model.

AAS implementations may decide, which subset of the access rules is implemented. If an AAS implementation e.g. only provides a READ profile, only such access rules for READ shall be allowed. Further subsets can be e.g. AAS repository, submodel respository, concept description repository, AAS registry or submodel registry.

AAS implementations (of non test systems) should consider relevant security standards and regulations, e.g. IEC 62443, ISO 27001, EU CRA or EU NIS2. This includes rules for the development process and for the written code. AAS implementations may provide millions of AAS. In such case, access rules are typically translated into the related AAS implementation to reach the needed perfomance and fulfill memory constraints.

Access Tokens to be used with the Asset Administration Shell are defined. Such Access Token is the result of an authentication flow which is not defined by this document. Important authentication flows are explained in this clause in Authentication Flows.

An Access Rule Model is defined, which uses Attribute Based Access Control (ABAC) as underlying concept. Claims of an Access Token are used as attributes for the ABAC authorization. Further attributes as DATETIME or machine state can also be included in the authorization.

Access rules can be defined for routes in the AAS API [2] (part 2), for Identifiables and Referables by reference or semanticId, or for certain assets.

Protection Goals

Confidentiality protection

Data leakage protection: Access to information considered sensible by an AAS responsible shall be controlled by well-defined rules. Only authorized parties shall be able to access sensitive information managed by the AAS. This applies to both,

- actual values (Properties) as well as the
- structure and meta-data represented by the AAS (topology described by Submodels, SubmodelElements, SubmodelElementCollections, …)

Integrity protection

Consistent data exposure: AAS shall ensure that data is exposed in a consistent way. Data exposure shall only depend on well-defined attributes. Attributes to be considered are, subject attributes (who is requesting the data), which action is to be performed (e.g., read, write, execute), object attributes (which data is accessed), and environmental attributes (contextual information like time, system state). Attribute Based Access Control (ABAC) is explained below.

Integrity and accountability of data: AAS users expect the data provided by the AAS to be correct and reliable. AAS shall consider the following:

- Protection against the entry of data by unauthorized parties. Entry/modification of data shall only be possible for authorized parties.
- AAS should allow to detect unintentional change of data and allow to trace the originator of data.
- Protection against manipulation of stored data. AAS should allow to protect asset data against unauthorized manipulation including the attribution of origin/authorship. This is especially important in case the AAS is operated by a 3rd party.
- Protection against the manipulation of the AAS structure. It is possible that moving Properties or SubmodelElements between different SubmodelElementCollections results in incorrect interpretation of the provided data.
- Protection against the manipulation of AAS management data. Management data especially includes the data relevant for the control of data exposure (e.g., access control).

Integrity of relation between data and asset: AAS shall protect against pretense of incorrect relations between an asset and its related data stored in the AAS. It is important to know that AAS data actually belongs to the asset as identified by the asset identifier recorded in the AAS.

Integrity of relation between services and asset: AAS shall protect against pretense of incorrect relations between an asset and services provided by the AAS. It is essential that services announced by the AAS affect the intended asset and that the behavior of the service is described correctly.

Protecting the Availability of Data

Availability of asset data: AAS shall protect against malicious hiding of current data. It shall be possible to determine whether data is sufficiently recent (e.g. lost updates) or data has been intentionally removed (e.g. malicious rollbacks).

IEC 63278

Figure 1 shows a detailed overview of AAS as defined in IEC 62378-1. The yellow comments have been added and describe the scope of this document in relation to the defined entities in the standard. Signatures will be defined in a later version of this document.

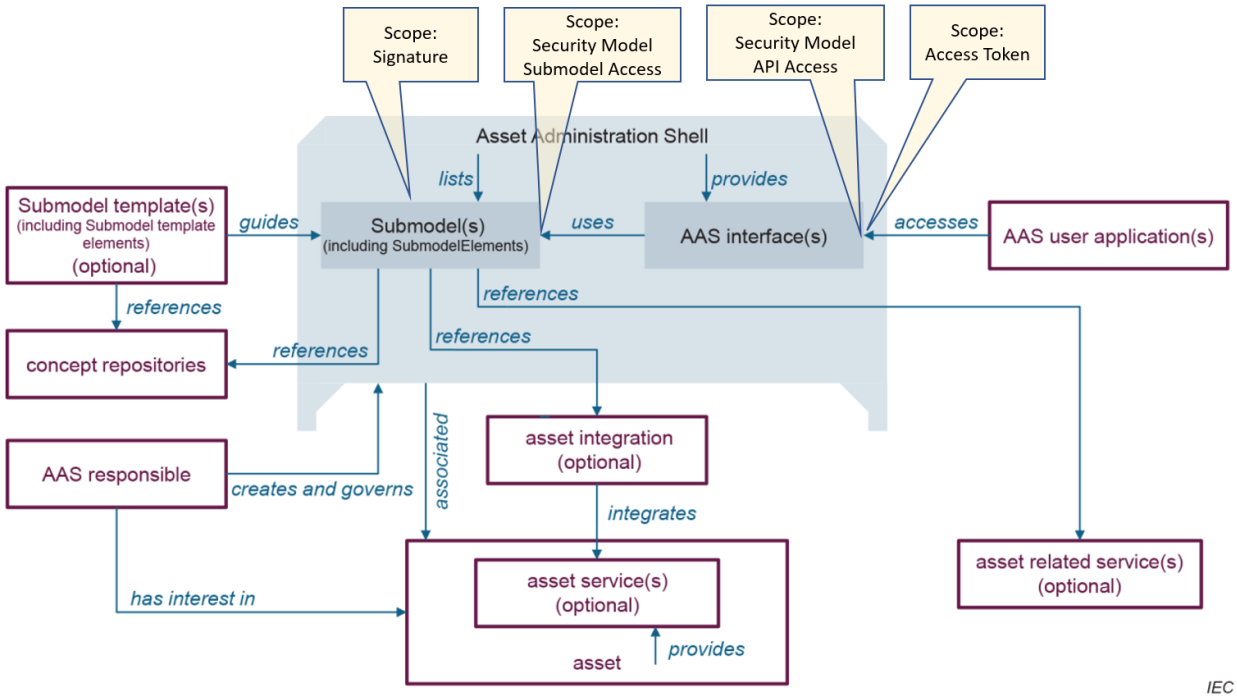

Figure 1. Detailed overview of Asset Administration Shell and related entities. Based on Figure 4 of IEC 63278-1 (Reproduced by permission of DKE, German Member of IEC (www.dke.de). For the valid edition see the latest publication of IEC or DIN EN IEC and www.dke.de and www.vde-verlag.de)

Types of AAS

As introduced in AAS Part 2 API, Figure 2 shows 3 different types of information via the Asset Administration Shell.

- Type 1 is the exchange as an AASX file [3] (part 5). Such AASX file may include a Security Model to be passed to a business partner. In the included Submodels in the AASX also the Signing of AAS may be used.
- Type 2 is the access to AAS by the API [2] (part 2) to a server. Such server may use and enforce the ABAC rules of the Security Model. Depending on the access rules, such server may also allow to access the submodel with the Security Model by the API. Submodels on the server may also include signed elements.
- Type 3 is the so called “Industrie 4.0 Language” with active AAS, which is similar to agents. Basically Type 3 is out of scope for this document, but the definitions of this document may also be used in the context of Type 3.

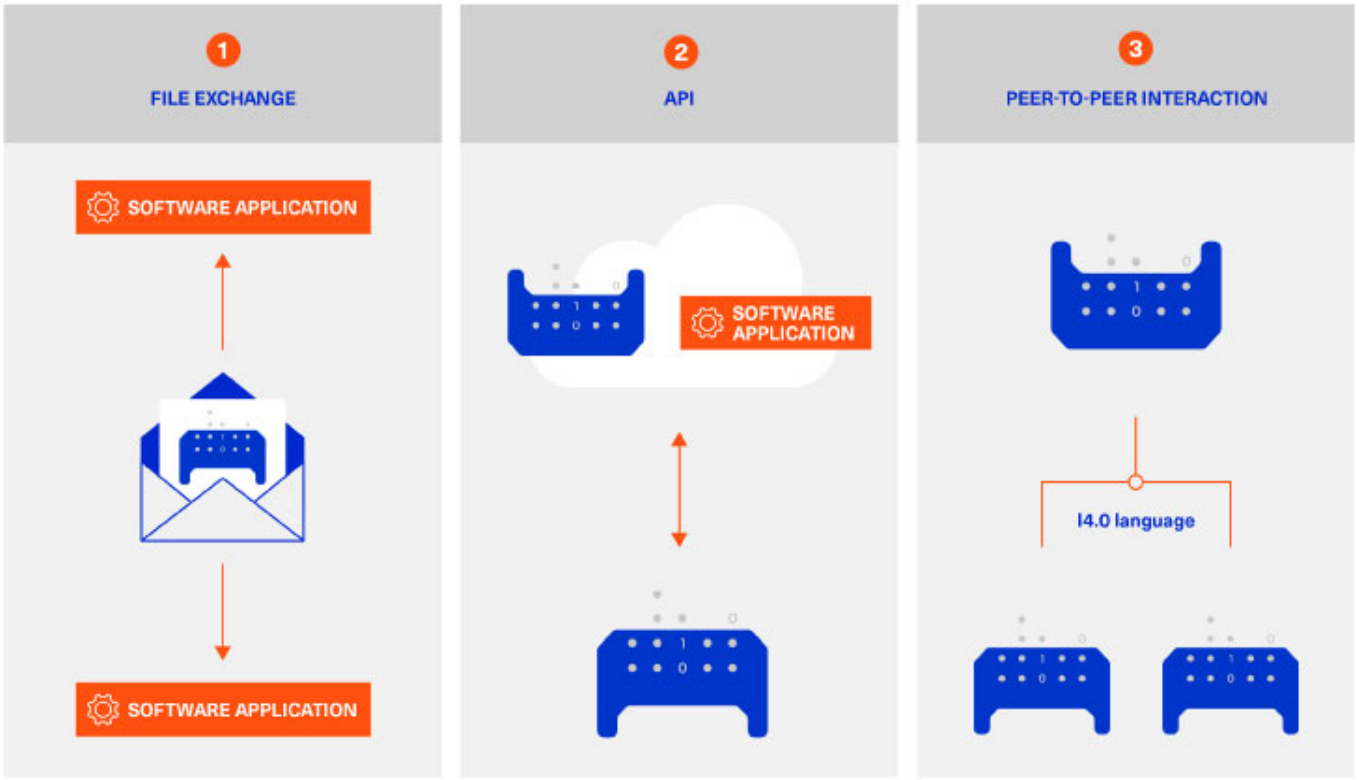

Figure 2. Types of Information Exchange via Asset Administration Shells

Services, Interfaces and Interface Operations

AAS Part 2 API also introduces the Industrie 4.0 Service Model illustrated in Figure 3 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 the operations (procedures) that may be called through an API. It is derived from the interface operation description on the technology-neutral level. When selecting technologies, one interface operation may be mapped to several API-operations; several interface operations may also be mapped to the same API-operation.

The implementation level comprises the following concepts:

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

The runtime level comprises the following concepts:

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

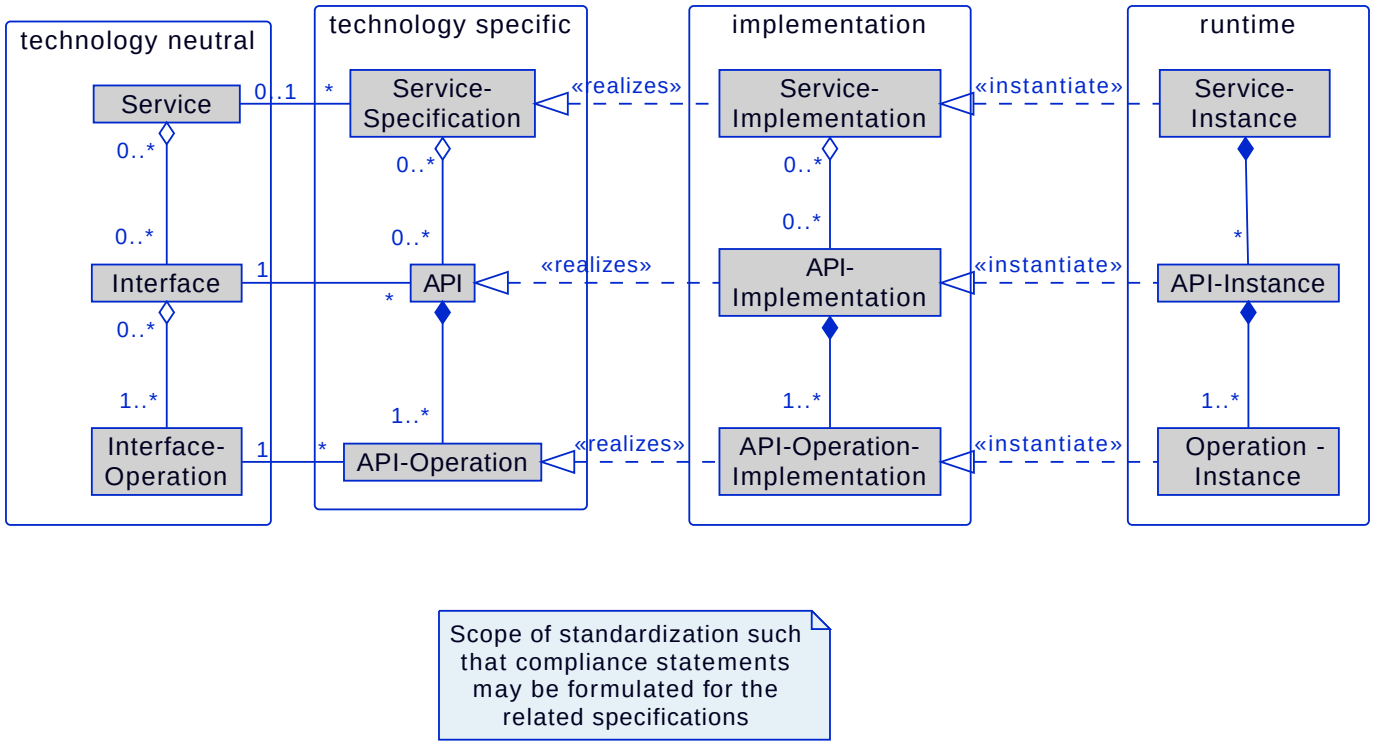

Figure 3. Services, Interfaces and Interface Operations

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.

Figure 4 shows AAS Services/Interfaces and an example sequence how they are called from a client application:

- At first a client application provides an asset ID (asset link) to the AAS Discovery Interface to retrieve the corresponding AAS ID or AAS IDs.
- By the AAS ID the related AAS Descriptor can be retrieved through the AAS Registry Interface.

An AAS descriptor includes the endpoint of the AAS and of the related Submodels.

- AAS or Submodels may be hosted standalone or as part of a larger AAS or Submodel Repository. In Figure 4 the first submodel is accessed by the Submodel Interface and the second submodel is accessed by the Submodel Repository Interface.

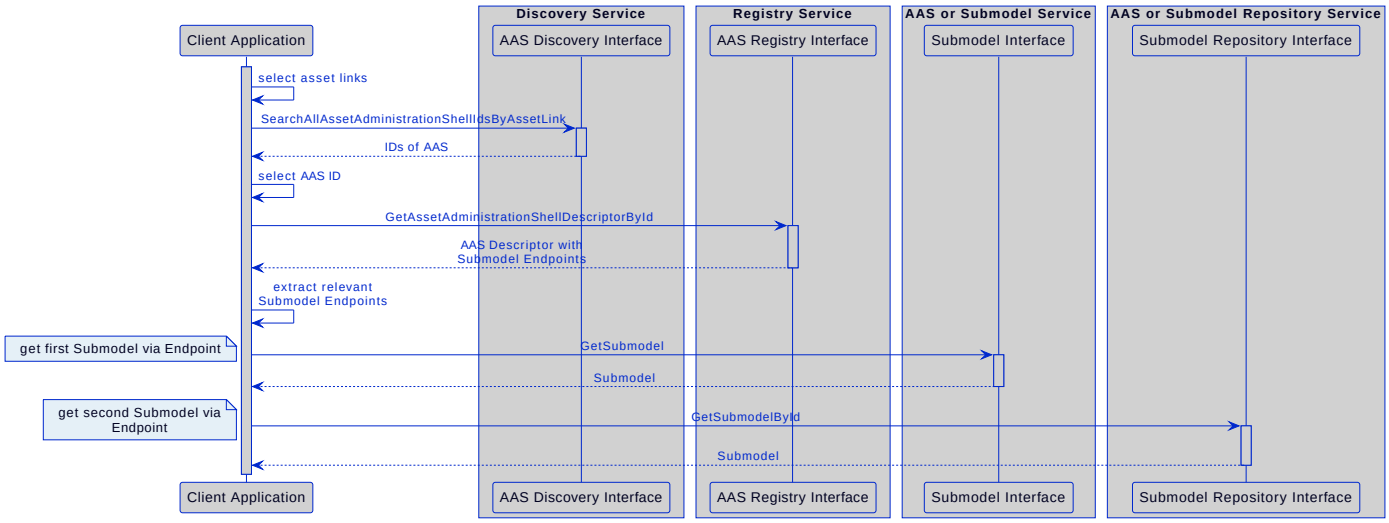

Figure 4. Sequence Diagram of AAS Services

Use Case File Exchange

Figure 5 shows an example use case of File Exchange between business partners as introduced by AAS Part 5 Package File.

In the example a supplier sends AAS as AASX by email to an integrator. The integrator may send these AAS and additional own AAS to his customers. The integrator may also only select certain submodels from the supplier's AAS.

This use case is an example for the necessity of Signing, since it must be possible to check the integrity of the AAS originally provided by the supplier. Signing will be defined in a later version of this document.

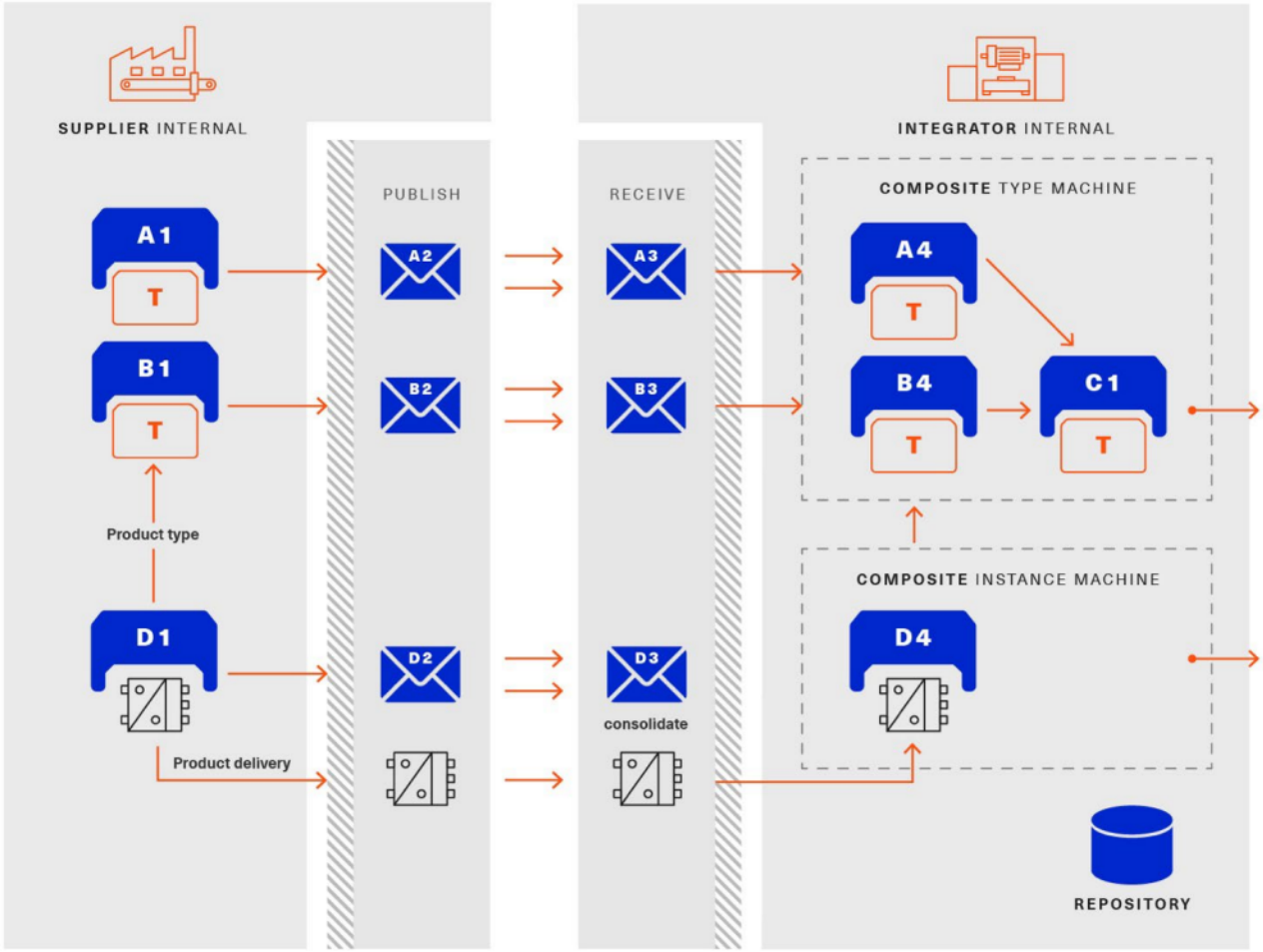

Figure 5. Use Case File Exchange between Value Chain Partners

Figure 5 also explains the need to pass access permissions to business partners.

As the AAS is a central point for data access, there is the need to support fine-grained access control that supports multiple roles as well as separate access control policies for individual elements or submodels in the AAS. Access Control is based on Identity Management and can only be successfully implemented in a secure environment. For this document, the focus lies on the supported access control model.

When having a look at the Use Case File Exchange in Figure 5 also security aspects have to be considered when transferring information from one value chain partner to the next.

When AAS content is passed from one partner to another, this is typically related to a change in the access control domain of the partners involved (supplier, integrator, operator), i.e. the scope of the validity of access control policies.

Therefore, for the example that the supplier passes on data to the integrator, the following typical steps are carried out:

- Step A1-A2: The supplier makes a choice which data is to be passed on, and thus determines which APIs are accessible to whom and/or the content of the AASX package.
- Step A2-A3: the AASX package is transferred to the integrator. With API this step is only needed, if the supplier pushes (POST, PUT) the AAS data to the integrator.
- Step A3-A4: The integrator receives the AAS by the API or receives the package and imports the content into his security domain. During this step, the integrator has to establish access rights according to the requirements in his own security domain.

ABAC is a very flexible approach, that also encompasses role-based access as a role can be considered as one attribute in this context. Other attributes might be the time-of-day, the location of the asset, the originating address and others.

In addition to the AAS content itself, also the defined access permissions have to be transferred between the partners due to the following two reasons:

1. Access permissions to information elements of an AAS must be established in each access control domain.
2. One partner must be able to pass a suggestion which access permissions should be established for the asset that is described in the AAS.

An example for the requirement (2):

A robot manufacturer suggests that for the robot the following roles shall be defined: machine setter, operator and a maintenance role. Note that the roles have to be expressed by means of attributes of the AAS representing the robot. He also suggests permissions for these roles, e.g. an integrator does have write-access to the program of the robot, but an operator does not.

The above example motivates that the access permission rules need to be passed from one access control domain to the other.

Use Case AAS Servers with API

Figure 6 shows an example of a User Application accessing 3 AAS Servers and a Registry. This example will be further detailed in Architecture Examples below.

The User Application will access the registry by the Registry API to retrieve the endpoints of AAS and Submodels.

It will then access the data on the different AAS servers by the related APIs.

This use case is an example for the necessity of Authentication, Authorization and Signing.

The AAS Servers may provide data available to the public, i.e. anonymous users. But an AAS Server may also restrict the access of certain data to specific users. In such case the User Application must first authenticate (not in the figure) and get an Access Token to supply its identity and further attributes to an AAS Server and also to the Registry. Access rules will be validated and enforced using the attributes in the Access Token.

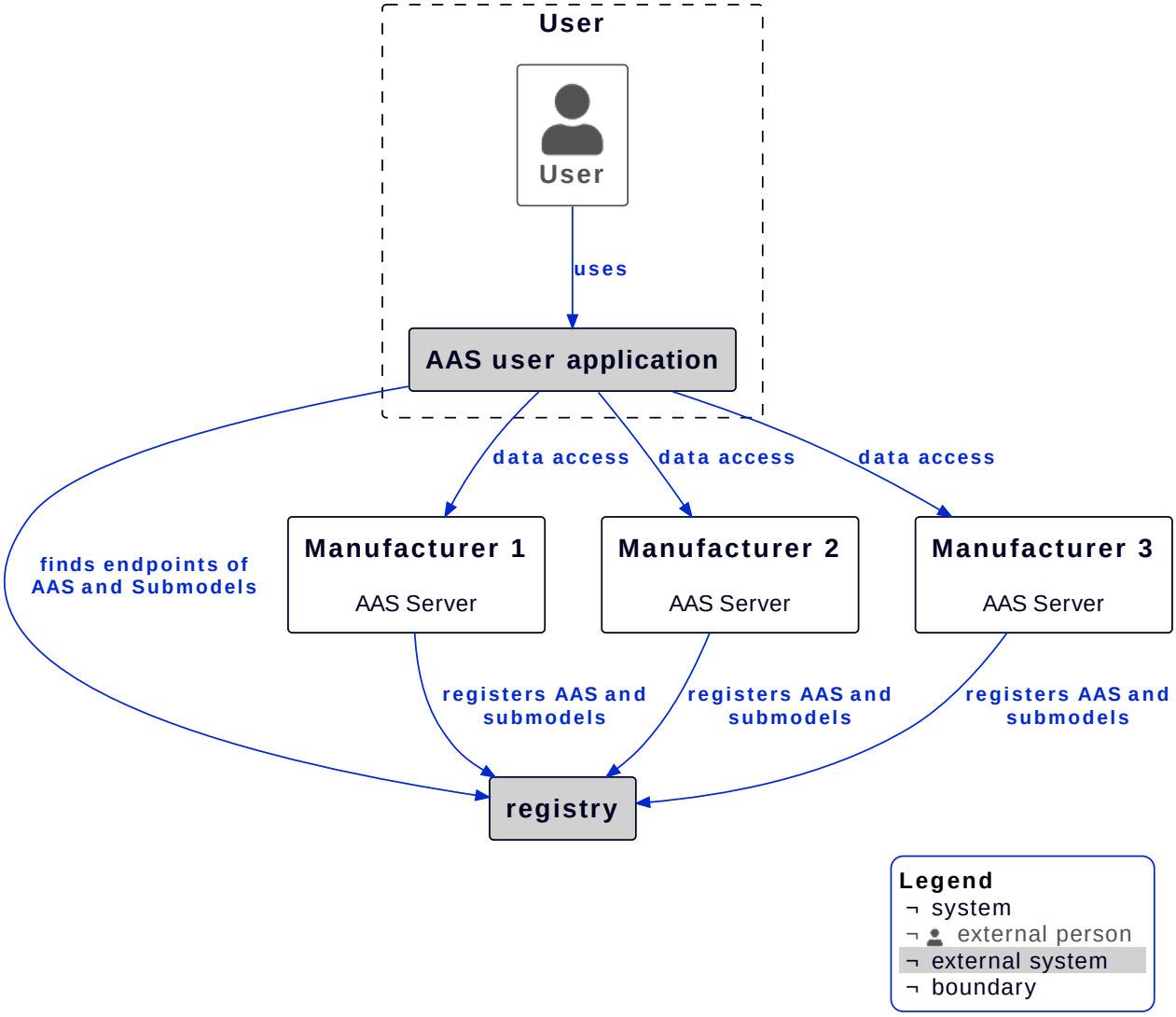

Figure 6. Example of a User Application accessing 3 AAS Servers and a Registry

AAS Server means either an AAS Repository with multiple AAS or just an AAS endpoint with a single AAS.

Further use cases are available at the Platform Industrie 4.0, e.g. Collaborative Condition Monitoring or Exchange of Engineering Data, and in IEC 63278-4 Use Cases.

Repository and registry

AAS Security applies to all AAS APIs, especially to both repositories and registries.

In case of repositories the Access Rules define the access to the data in the repository itself.

A registry only includes AAS descriptors or submodel descriptors. In that case the Access Rules define the access to the descriptors, i.e. by rules with semanticId, assetId or ID for identifiable.

This gives the possibility to copy the related rules made for a repository also to a registry, either manually or even automatically.

Architecture Examples

Protecting an AAS Server

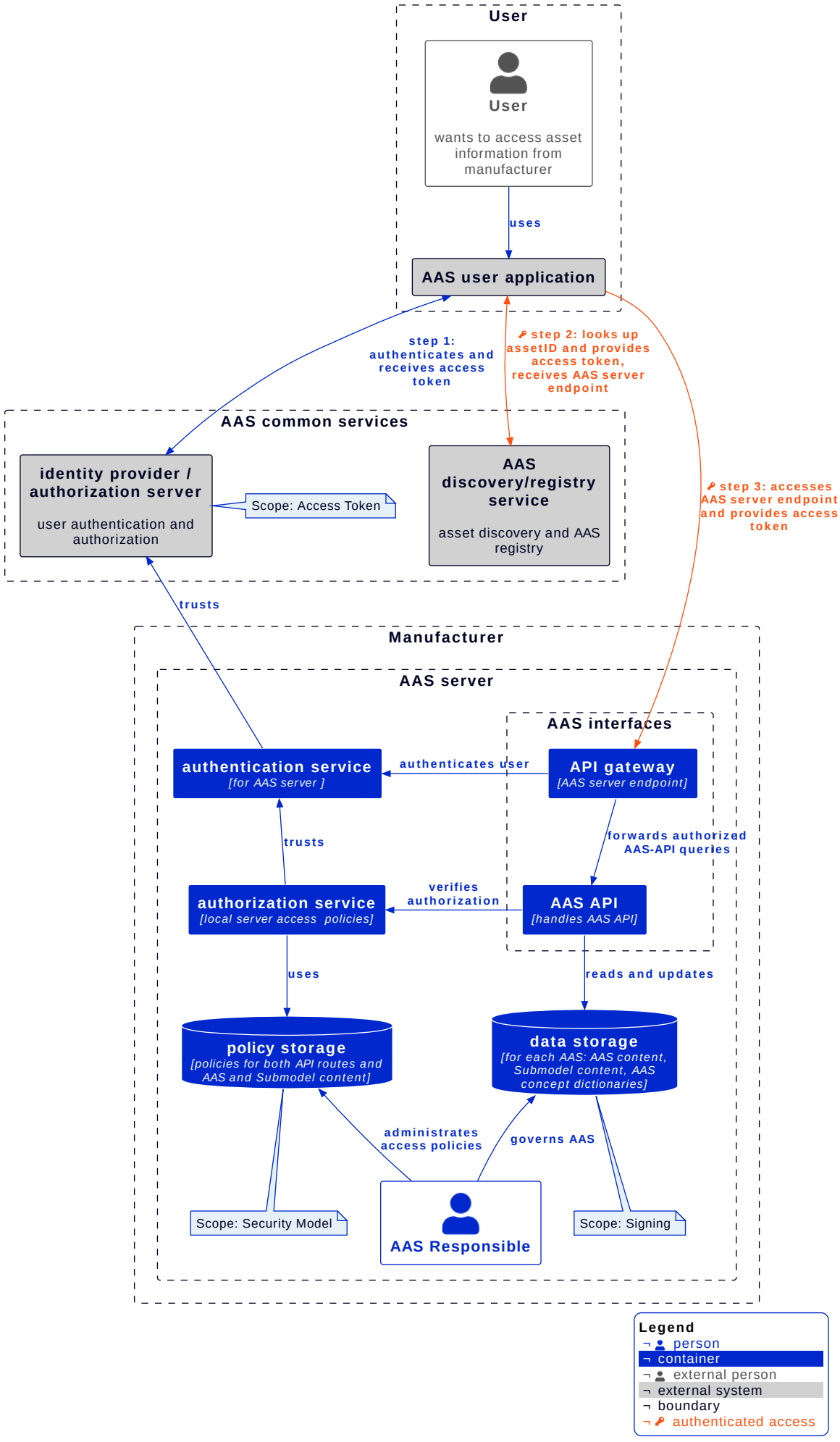

Figure 7. Example of a possible AAS server implementation

Figure 7 shows a possible AAS server implementation with security:

- A **User** wants to access AAS information from a **Manufacturer** using an **AAS User Application**.
- The **Manufacturer** hosts an **AAS Server** which is internally administrated by an **AAS responsible**.
- In step 1 the **User Application** authenticates at and receives an access token from the **Identity Provider / Authorization Server**.
- In step 2 the **User Application** looks up e.g. the aasId at the **AAS Discovery/Registry Service** and receives the related AAS endpoint for the **AAS Server**.
- In step 3 the **User Application** accesses such endpoints in the **AAS Server**.
- First such access will be handled by the **API Gateway** in the **AAS Server**. Such **API Gateway** will verify the signature of the access token by the **Authentication Service**, which uses the public key of the **Identity Provider / Authorization Server**.
- Second the **AAS API** verifies by the **Authorization Service**, that the requested API route is authorized. This may be defined by related ABAC access rules for that API route.
- Third the **AAS API** verifies by the **Authorization Service**, that the requested AAS element is authorized. This may be defined by related ABAC access rules with IDs of Identifiables, Referables, Assets or semanticIds.
- Fourth the **AAS API** will access the **Data Storage**, but only in case of all the positive authorizations above.
- The **AAS Responsible** may receive proposed ABAC access rules from his business partners, e.g. a machine supplier proposes such rules to a machine operator. Such access rules may be stored in a proprietary way by the **AAS Server**, but they might also be stored in the Data Storage in AAS format as received.
- As described for the **AAS Server**, the AAS **User Application** will also provide the access token to the **Discovery/Registry Service** which can make the verification and authorization accordingly.

Protecting an AAS Registry

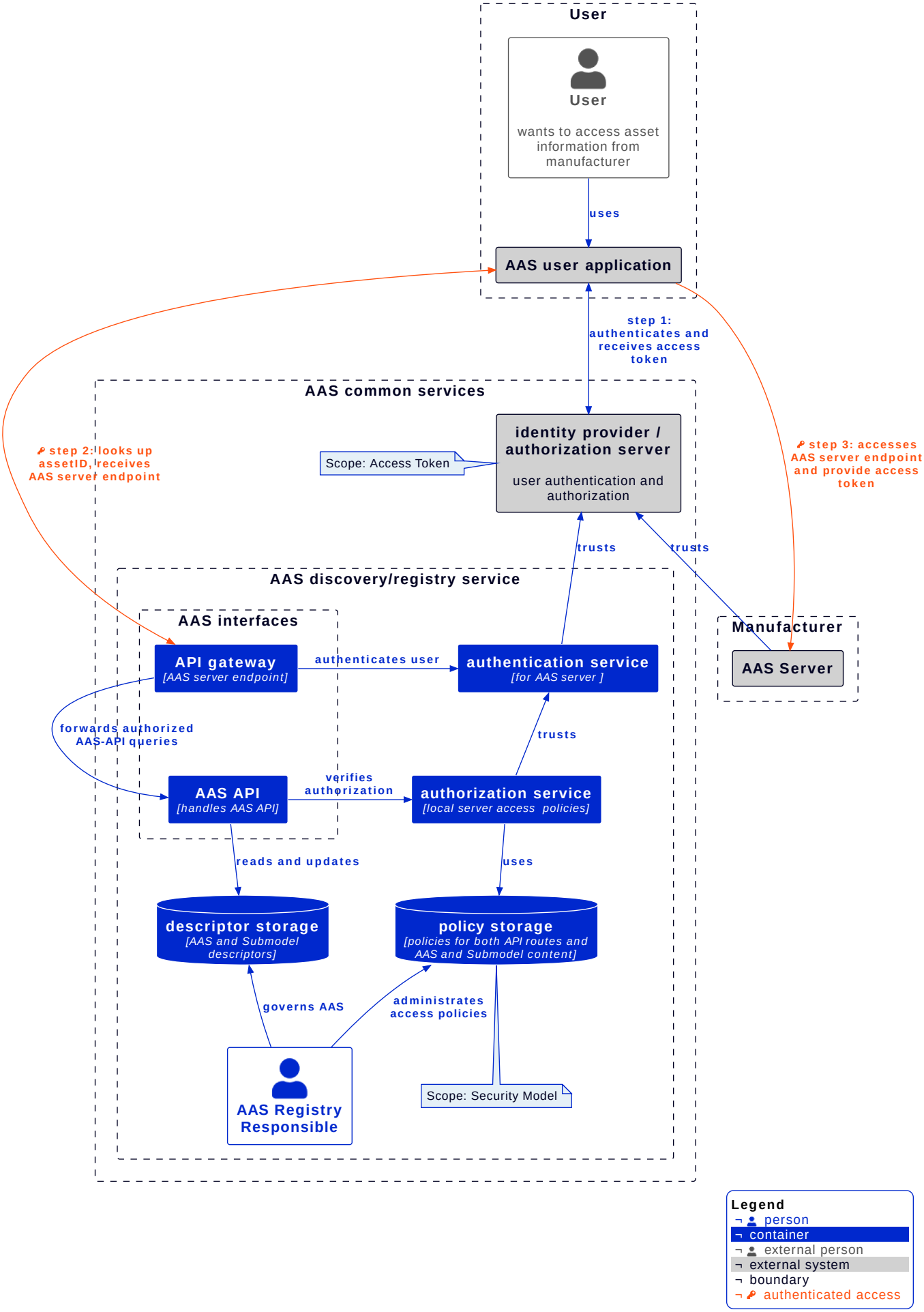

Figure 8. Example of a possible AAS registry implementation

Figure 8 shows a possible AAS registry implementation with security:

- The same example as before is shown, but this time the **Discovery/Registry Service** is shown in more detail.
- Instead of AAS Data only AAS and Submodel descriptors are stored in the descriptor storage.

Attribute Based Access Control (ABAC)

The objective of access control is to protect system resources (here: AAS content) against unauthorized access. The protection measures are specified in access control policies whose scope of validity is defined by security domains dedicated to access control.

The underlying concept applied for access control is the concept of attribute-based access control (ABAC). In general, the ABAC request flow is described in [8]. Originally, ABAC relies upon the data-flow model and language model of the OASIS eXtensible Access Control Markup Language (XACML) specifications [9].

OASIS XACML includes concepts such as:

- Policy administration point (PAP): The system entity that creates a policy set.
- Policy decision point (PDP): The system entity that evaluates an applicable policy and renders an authorization decision.
- Policy enforcement point (PEP): The system entity that performs access control, by making decision requests and enforcing authorization decisions.
- Policy information point (PIP): The system entity that acts as a source of attribute values.

The general request flow is depicted in Figure 9:

- A subject is requesting access to an object (1). In the context of an AAS, an object is for example a submodel or a property or any other submodel element.
- The implemented access control mechanism of the AAS evaluates the access permission rules (2a) that include constraints that need to be fulfilled w.r.t. the subject attributes (2b), the object attributes (2c) and the environment conditions (2d).
- After the evaluation a decision is taken and enforced upon the object (3), i.e. the access to the object is permitted or declined.

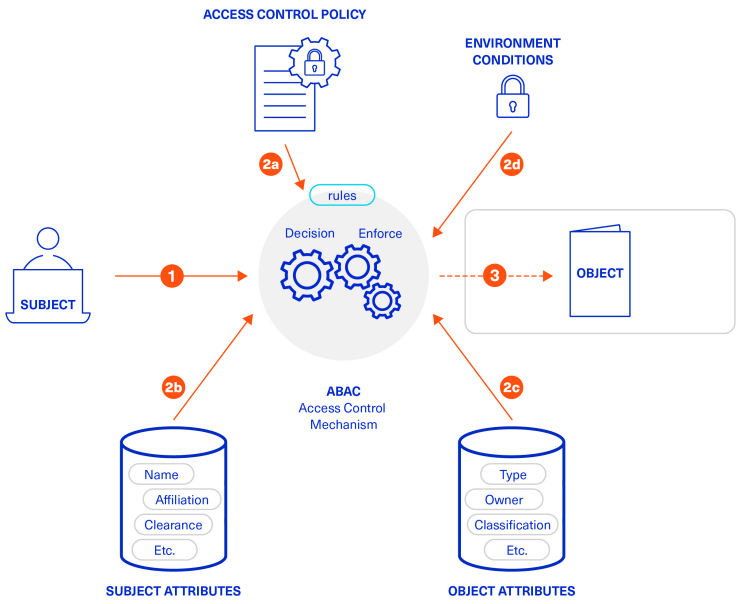

Figure 9. Attribute Based Access Control

In this document the concept of attributes is specialized for the application of ABAC in the context of AAS

authorization.

Attributes may be claims from an access token (e.g. subject attributes), global attributes (e.g. environmental conditions) or attributes from AAS and submodel data (e.g. object attributes).

Dataspaces

A Dataspace facilitates the exchange of data between participants via a set of technical services. Each participant is capable of implementing and operating a set of services/agents that allows them to participate in data exchange. A Dataspace Authority manages the specifications and conventions for these interactions. Identity Providers issue identity tokes to the participants who can in turn decide which issuers to trust.

AAS Part 2 defines interoperability on the level of standardized interfaces serving business objects leveraging the semantics of AAS Part 1. Neither specification makes normative statements about common mechanisms to establish authenticity, identification or authorization between consumers. This is however necessary to define the terms and conditions for secure access to the standardized AAS-interfaces.

Interoperability across all layers of a Dataspace requires as answer a layered set of specifications with well-defined integrations among them.

In a dataspace members can exchange data securely and interoperably.

Such data exchange is always done in the same way, so that scalability and efficiency can be achieved.

Millions of business partners shall be able to easily exchange data with each other in a dataspace.

The data exchange can be with and without user interaction.

To achieve interoperability, members of a dataspace must agree on:

- Common models
- Common APIs
- Common security (e.g. credentials, authentication, authorization)

For a dataspace with AAS there are already specifications available:

- AAS metamodel
- AAS API, including AAS Discovery and AAS Registry
- Many AAS Submodel Templates (e.g. Nameplate) conformant to the AAS metamodel
- AAS Security (i.e. this document)

In addition, a dataspace with AAS needs to define:

- Further dataspace specific Submodel Templates
- Allowed concept repositories to be referenced by semanticIds
- Technology for identities, e.g. X509 or Verifiable Credentials
- Technology for identity providers, e.g. OAUTH, OIDC, Dataspace Protocol/Decentralized Claims Protocol
- Using central or decentral identity providers
- Claims used and allowed in JSON Web Tokens issued by identity providers

Remark: Until now no common AAS specifications for the following concepts are published and need also to be defined in a dataspace with AAS:

- How to find AAS registries of the dataspace members? Solutions may be a registry with registry endpoints or a naming convention like registry.company-domain.com. Also, an indirect registration via dataspace connectors that

also support additional data exchange patterns besides AAS, e.g. simple bilateral file exchange, is possible.

- How to find concept repositories? A solution can be a registry of concept repository endpoints or a concept repository discovery service for semantic IDs. Additional discovery services if needed, for example for the company-domain etc.

Integration Patterns

With AAS many use cases between business partners are supported. In this chapter different integration patters, see Figure 10, are described, which are used in such use cases and how standardized AAS Security supports this.

An integration pattern describes

- if an own AAS or an AAS of a business partner is used.
- if an AAS is hosted on a server internally or externally.
- if an AAS is copied.
- the requirements on Identity Providers.
- the requirements on access tokens and claims (as part of a JSON Web Token).
- the requirements on access rules.

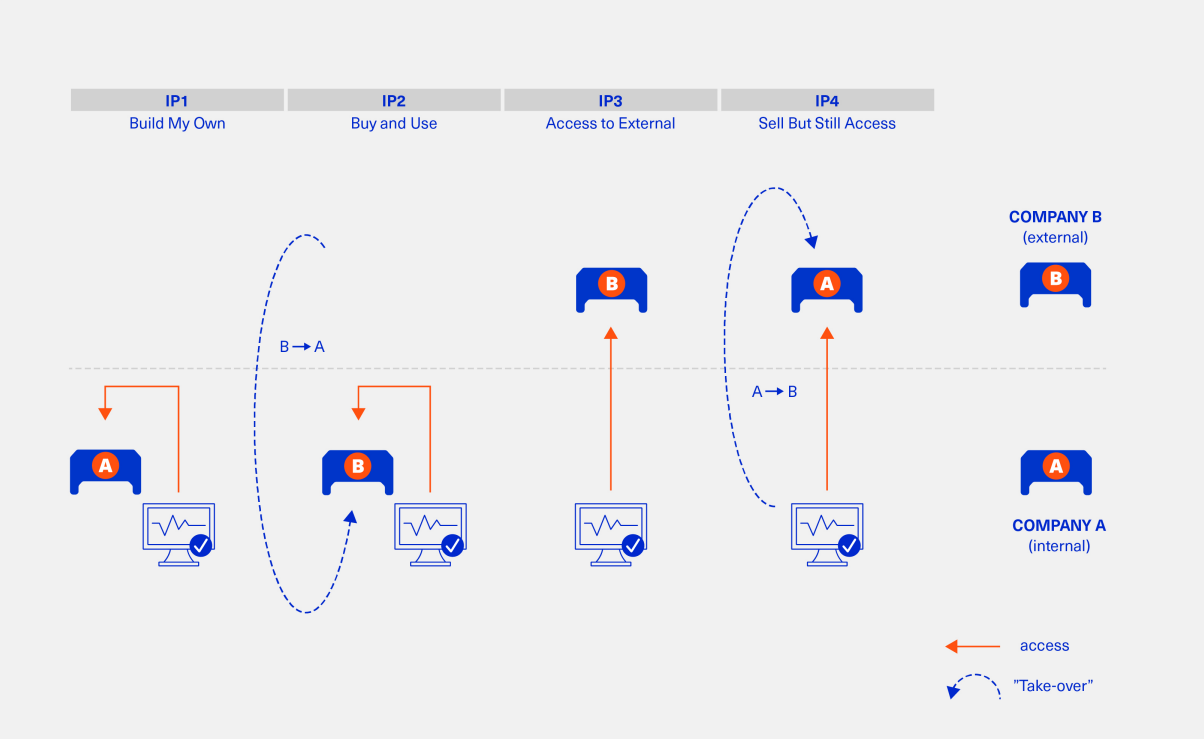

Figure 10. Integration patterns

Integration Pattern “Build My Own” (IP1)

- Company A creates, manages and uses its own AAS. The AAS is not accessible from outside the Company.
- Everything can be decided internally by the company (Company A).
- Company hosts AAS on a server internally. The server software may be open source, a commercial product or a software as a service.
- Company will choose an Identity Provider, that is aligned with company’s IT infrastructure. Chosen server software must be able to support this.
- Chosen server software may have predefined claims or may be fully configurable for claims.
- Chosen server software may have certain limited access rules or may be fully configurable for access rules according standardized AAS security.

- For many internal Use Cases the security may be complete company specific not using the standardized AAS security. Since the company may decide later to make their AAS available to other companies (pattern “Access to External”) or the company may use existing software to host their AAS, company may still benefit from using standardized AAS security.

Integration Pattern “Buy and Use” (IP2)

- Company A buys a product from Company B and uses and manages it internally.
- Company B has created the AAS and Company A copies the AAS.
- The AAS is hosted on a server of company A.
- Company A defines Identity Provider, access token and claims.
- Company B may propose standardized AAS Access Rules to Company A, which Company A may simply copy and paste into his AAS Server.
- With respect to security this case is very similar to the “Build My Own” access pattern.

Integration Pattern “Access to External” (IP3)

- Company A accesses an AAS on an external server of Company B.
- Company A may access AAS from one business partner (bilateral) or may access AAS from many different business partners (multilateral).
- The external business partner Company B hosts his AAS on his server.
- Company A accesses AAS by the API and uses it in its user applications. If the usage policy from Company B allows it, Company A may copy the AAS and store it in its own systems.
- In case of bilateral access, Identity Provider, access token and claims are defined by Company B.
- In case of multilateral access, a common Identity Provider of a dataspace is used. The members of the dataspace have to define access token and claims.
- Access rules need to be defined by Company B to allow access by Company A.

Integration Pattern “Sell But Still Access” (IP4)

- Company A sells a product to Company B.
- Company A has created the AAS and Company B copies the AAS.
- The AAS is hosted on a server of company B.
- Company B defines Identity Provider, access token and claims.
- Company A may propose standardized AAS Access Rules to Company B, which Company B may simply copy and paste into his AAS Server.
- With respect to security this case is similar to the bilateral “Access to External” access pattern.

External and Internal Identity Providers

A Secured System in AAS Security can be Discovery Service, Registry Service, AAS/Submodel Service or AAS/Submodel/ConceptDescription Repository Service.

Figure 11 shows different alternatives of Clients, Identity Providers (IDP) and Secured Systems interacting.

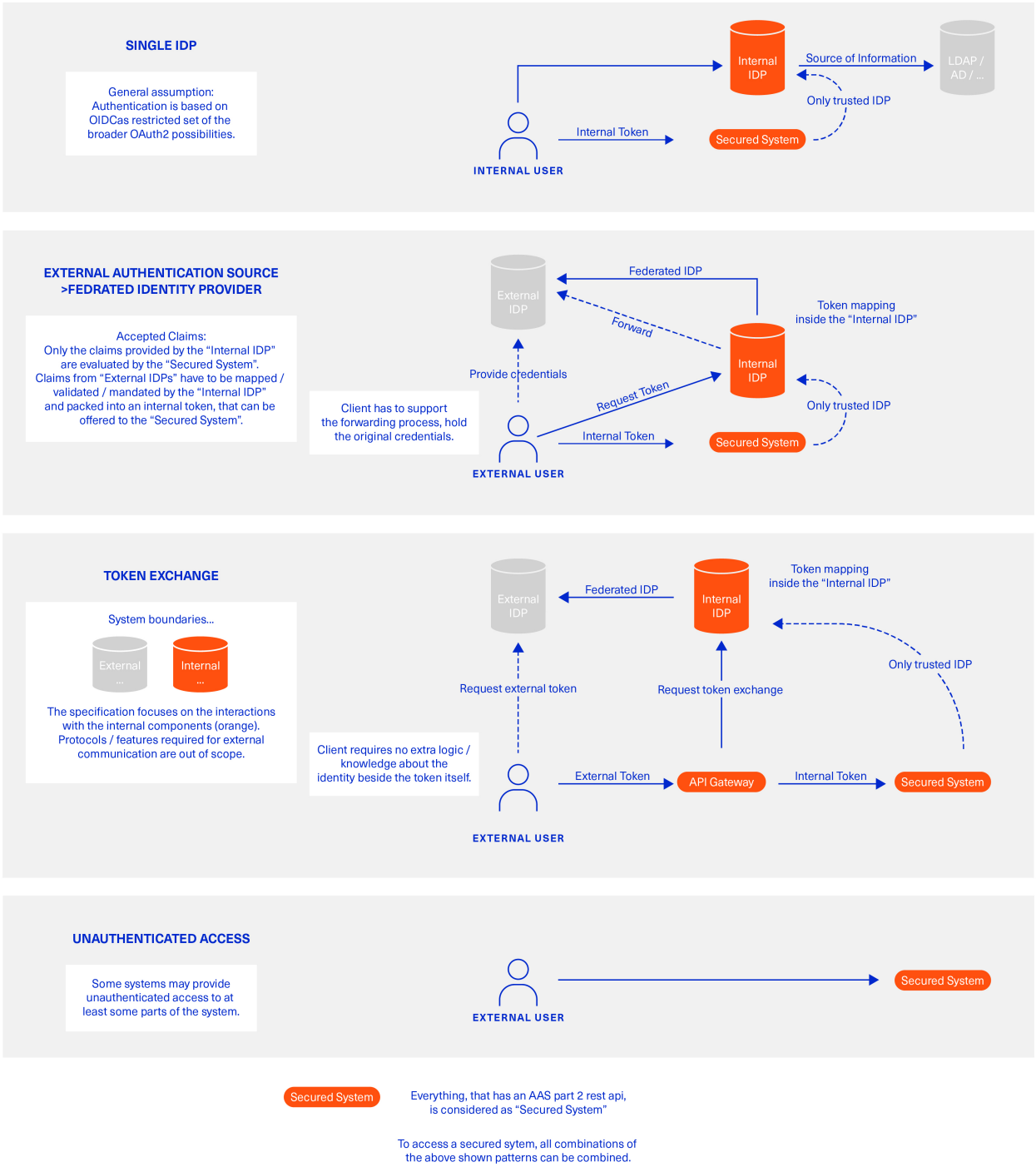

Figure 11. External and internal Identity Providers

Unauthenticated Access

AAS Security supports unauthenticated access to a secured system.
A lot of information is publicly available to human users by web browsers. If such information is provided machine-readable as AAS, it is typically also public without authentication. Examples are the submodels Nameplate or TechnicalData.

Single Identity Provider

AAS Security supports company specific Identity Providers.

In case an internal system is set up, an internal Identity Provider will be used, which can be used by internal user applications.

This might even be used for external user applications, in case a company IT might not allow to support the federated identity providers below. In that case still AAS with AAS Security can be provided, but using a company specific Identity Provider.

Federated Identity Provider

AAS Security supports Identity Providers of dataspaces or of other multilateral company use cases.

In that case, an External User authenticates to an External Identity Provider and receives an external token.

This external token is provided to the Identity Provider of a company and the External User receives an access token. Such federation is possible, when an Identity Provider of a company can check the validity of the external token and trusts the External Identity Provider.

Token Exchange

In case of “Federated Identity Providers”, the external user has to be aware of the “Internal IDP” and has to process (store and forward) the token to the secured system. If the external user requires access to other resources, that are not secured by the “Internal IDP”, he has to deal with multiple access tokens.

To overcome this complexity, RFC 8693 (https://www.rfc-editor.org/rfc/rfc8693) specifies a mechanism, where the external user is able to use its “natural / external” access token to communicate with a secured system under control of the “Internal IDP”.

In cases where a shared AAS/SM Registry references identifiables from different providers, the token exchange protocol enables transparent access to resources of different secured systems under control of different Identity Providers, as long as the Identity Providers trust each other and are able to map the tokens as described in RFC 8693.

From administration point of view, the Federated Identity Provider and the Token Exchange approach enable a system owner to fully control access to their secured systems, because the Identity Provider is managed by the system owner.

A distributed identity layer may also be achieved via self-sovereign identity wallets.

Authentication Flows

AAS Security supports any authentication flow which provides signed JSON Web Tokens. Even other formats for Access Tokens might be used.

An authentication flow can be with or without user interaction, depending on the dataspace or use case.

Authentication flows like OAUTH 2.0 or OIDC can be used.

Authentication flows as defined by a specific dataspace or by a specific dataspace protocol can be used.

In any case a signed JSON Web Token must be provided by the related Identity Provider.

The following sections describe example flows that can be used.The following terms are used in the description:

Resource owner: A logical entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as an end-user [RFC 6749]. In the context of AAS, the Resource owner is the AAS Responsible, or a user or organization that has been entitled by the AAS Responsible. In the OAuth 2.0 Client Credential Flow, the client acts on behalf of the resource owner and therefore receives permissions according to the rights of the resource owner.

Resource server: The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens. [RFC 6749]

Authorization Server: The server issuing access tokens to the client after successfully authenticating the resource

owner and obtaining authorization. [RFC 6749]

Identity Provider: An entity (usually an organization) that is responsible for establishing, maintaining, securing, and vouching for the identities associated with individuals. [RFC 6973]

Example: OAuth 2.0 Client Credential Grant

The specification of the OAuth 2.0 Framework (RFC 6749) defines Client Credentials flow as:

Client credentials are used as an authorization grant typically when the client is acting on its own behalf (the client is also the resource owner) or is requesting access to protected resources based on an authorization previously arranged with the authorization server.

- The Client Credentials Flow involves an application providing its application credentials.
- The application credentials are client id and client secret.
- This flow is best suited for Machine-to-Machine (M2M) applications, such as CLIs, daemons, or backend services.

The system must authenticate and authorize the application instead of a user.

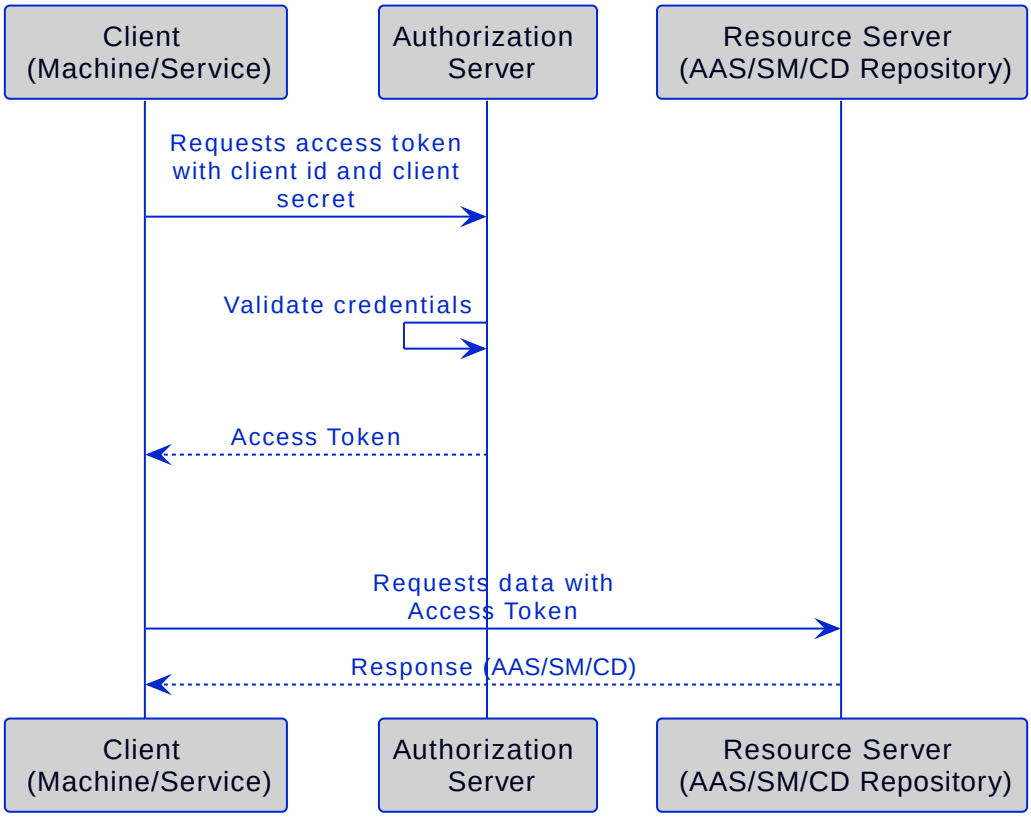

Figure 12. Authentication Flow OAuth 2.0

Figure 12 represents an OAuth 2.0 Client Credentials Grant flow that consists of the following steps:

1. **Client Requests Access Token**:
   - The client (Machine/Service A) sends a request to the Authorization Server (Identity Provider).
   - This request includes the client's credentials, specifically the client ID and client secret.
2. **Authorization Server Validates Credentials**:
   - The Authorization Server receives the request and validates the client's credentials (client ID and client secret).
   - Client secrets are usually long, random strings designed to be difficult to guess. It is commonly an alphanumeric string or Base64 encoded string.

- If the credentials are valid, the Authorization Server generates an access token. If not, an error is returned.

3. **Authorization Server Issues Access Token:**
   - After successful validation, the Authorization Server sends an access token back to the client.
4. **Client Requests Data with Access Token**:
   - The client uses the obtained access token to make a request to the Resource Server (AAS/SM/CD Repository).
   - The access token is included in the request header for authentication and authorization.
5. **Resource Server Responds**:
   - The Resource Server validates the access token.
   - If the token is valid and has the necessary permissions, the Resource Server processes the request and sends the requested data back to the client.
   - The response contains the requested data (AAS/SM/CD).

In the OAuth 2.0 Client Credentials Grant flow the client id and client secret can simply be any string. As per the official specification [RFC 6749] the client id and client secret is any visible (printable) ASCII character (*VSCHAR). The Resource server evaluates and checks the claims in the access token and validates it with the Access Rules, as specified later in this specification.

Example: OpenID Connect

ISO/IEC 26133:2024 defines OpenID connect as:

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It enables Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

Client Credentials Flow in OIDC: While the Client Credentials flow is typically used for authorization (OAuth 2.0), OIDC can also be used in a similar manner but is more often used for scenarios involving (human) user authentication. However, the main distinction comes when OIDC is used in other flows, such as the Authorization Code flow, where it provides ID tokens to assert the identity of the end-user.

ID Token: The ID token is a JSON Web Token (JWT) that contains user profile information (claims) and is signed by the identity provider.

Instead of using user context, the client credential flow uses application context, and an ID token is not issued in this scenario. Only access tokens can be obtained by applications.

The following sequence diagram explains the OpenID Connect in OAuth2 Authorization code grant flow:

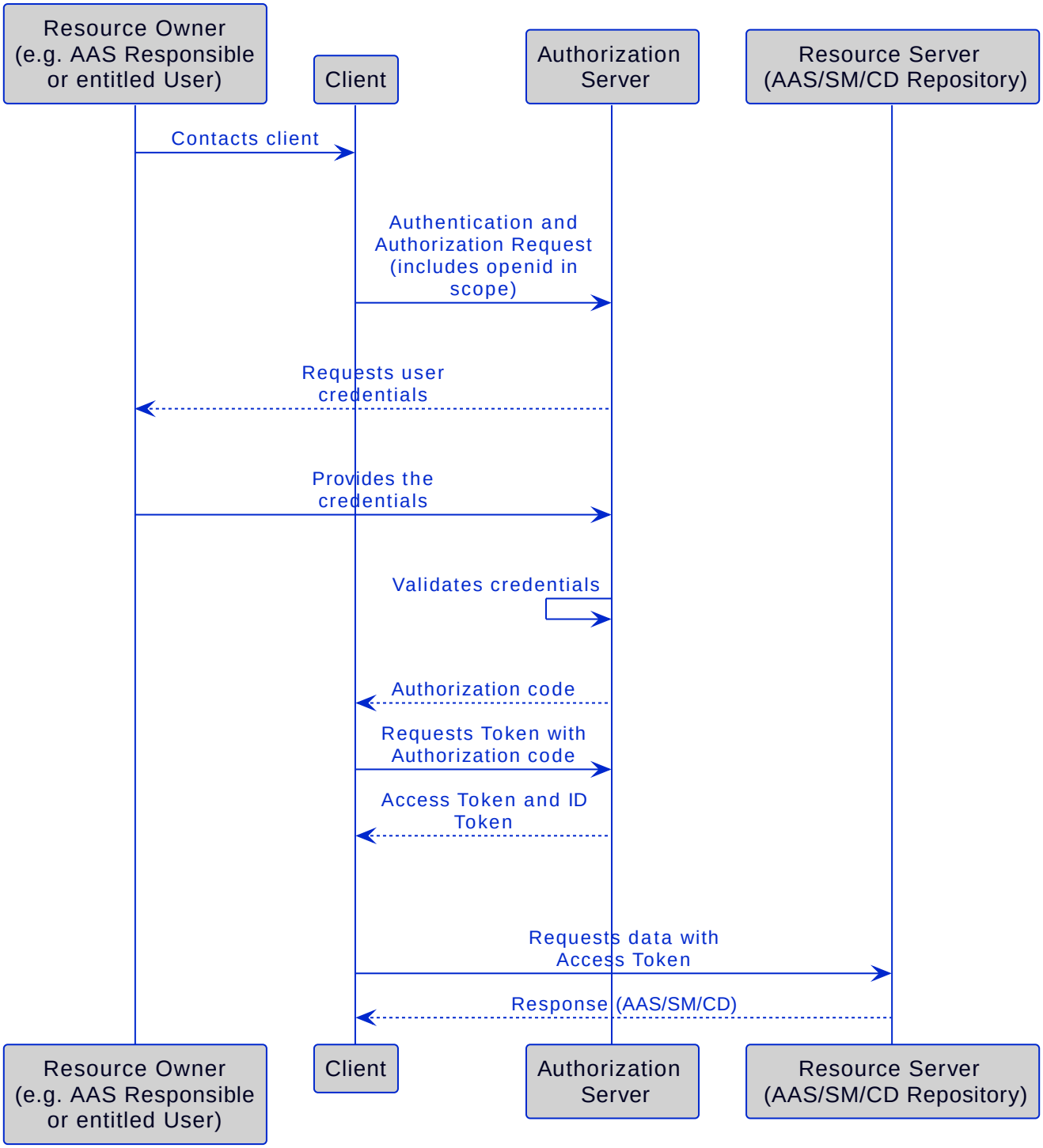

Figure 13. Authentication Flow OpenId Connect

Here's a step-by-step explanation of the flow depicted in Figure 13:

1. **Authentication and Authorization Request:** The client application initiates an authentication and authorization request to the Authorization Server (Identity Provider). This request includes the openid scope, indicating that the client is requesting authentication using OpenID Connect.
2. **User Provides Credentials**: The user is prompted to provide his credentials (e.g., username and password) to the Authorization Server.
3. **Authorization Code**: Upon successful authentication, the Authorization Server issues an authorization code and redirects the user back to the client application with this code.
4. **Request Token with Authorization Code**: The client application sends a request to the Authorization Server to exchange the authorization code for tokens. This request includes the authorization code received in the previous step.

5. **Access Token and ID Token**: The Authorization Server validates the authorization code and, if valid, issues an access token and an ID token to the client application.
6. **Request Data with Access Token**: The client application uses the access token to request data from the Resource Server (AAS/SM/CD Repository). The access token is a credential that allows the client to access protected resources on behalf of the user if authorized.
7. **Response**: The Resource Server validates the access token and, if valid, responds with the requested data (AAS/SM/CD).

Tokens:

- **Authorization Code:** A temporary code issued by the Authorization Server after the user successfully authenticates. It is used to obtain the access token and ID token.
- **Access Token:** A token that allows the authorized client to access protected resources on behalf of the user.
- **ID Token:** A token that contains information about the user (such as user ID, email) and is used to verify the user's identity.

AAS are used both in contexts with and without User, i.e. often in Machine-to-Machine (M2M) applications. For M2M typically only an Access Token and no ID Token is used. In both contexts the Authorization Server creates an Access Token with claims. The AAS server will evaluate and check these claims by the Access Rules as specified later in this specification.

Example: DataSpace Protocol

According to the Dataspace Protocol (DSP) [7], a Dataspace is "A set of technical services that facilitate interoperable Dataset sharing between entities." The specification goes on to define interactions between a Data Provider and Data Consumer to expose metadata, agree on conditions for exchange and execute the transfer of Datasets. Asset Administration Shell resources can be such Datasets.

A Dataspace may decide for a subset of all AAS Service Specifications of AAS Part 2, e.g. using both AssetAdministrationShellRegistryServiceSpecification and SubmodelRepositoryServiceSpecification together with AssetAdministrationShellDescriptors and included SubmodelDescriptors. Specific Dataspaces define conventions on the standardized exposure of these resources for Data Consumers to discover and retrieve.

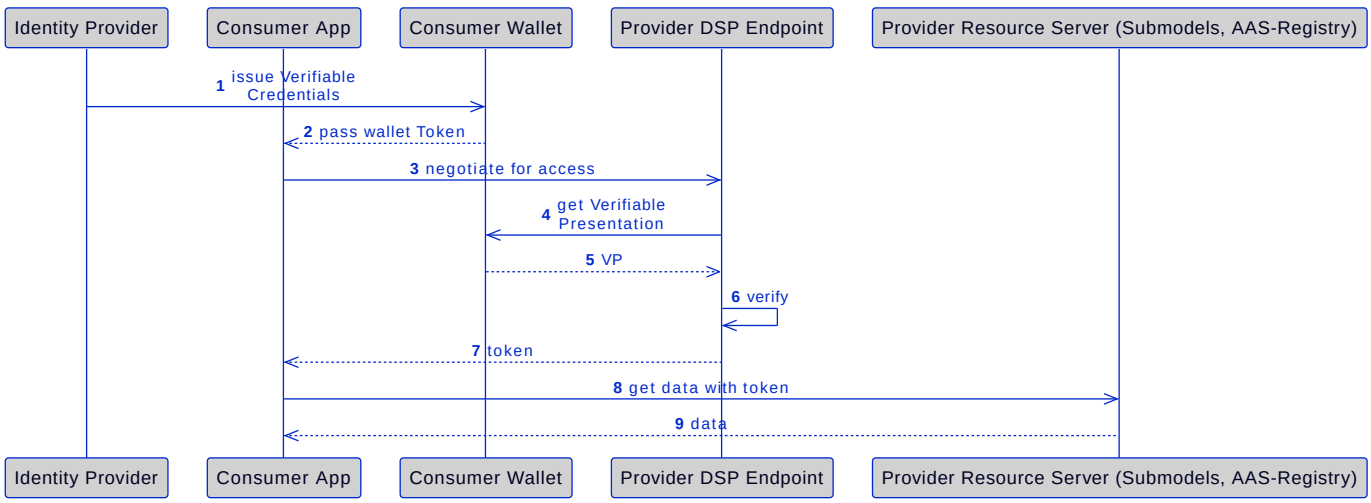

Figure 14. Dataspace Protocol negotiation with integrated SSI-based authentication flow for credential presentation

A Provider will define access conditions encoded as ODRL-Policies that a potential Consumer must comply with. To authorize against these Policies, a Consumer may be required to provide access to long-living claims signed by a commonly trusted party (Identity Provider, step 1). These claims will usually follow the Verifiable Credential Data Model embedded in a jwt (see Example Verifiable Credential for authentication and authorization against Dataspace Protocol endpoints):

Example Verifiable Credential for authentication and authorization against Dataspace Protocol endpoints

```
{
  "iss": "did:web:dataspace-identity-issuer.com",
  "vc": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://w3id.org/dataspace-specific/credentials/context.json"
    ],
    "id": "1f36af58-0fc0-4b24-9b1c-e37d59668089",
    "type": [
      "VerifiableCredential",
      "SpecificCredentialType"
    ],
    "issuer": "did:web:dataspace-identity-issuer.com",
    "issuanceDate": "2021-06-16T18:56:59Z",
    "expirationDate": "2022-06-16T18:56:59Z",
    "credentialSubject": {
      "id": "did:web:consumer.com",
      "holderIdentifier": "SomeUniqueIdentifier"
    }
  },
  "jti": "ba1300d8-8a78-40b4-8749-60d84fc3ab97",
  "sub": "did:web:consumer.com",
  "iat": 1737276930,
  "nbf": 1737363330,
  "exp": 1768985730
}
```

A Data Provider will verify those claims by validating the VC's signature using the Identity Provider's public key (step 5). This step ensures the authenticity of the claims. If the claims match the Data Provider's expectation for a particular Policy Constraint, the Data Provider will return information about accessing the exposed AAS resources. This will usually include the URL and a short-living access token as well as a refresh token (step 6). The Tokens' format is usually irrelevant for interoperability as it is issued and validated by the same Participant and remains opaque to the Consumer. Only scenarios where the Policy Decision Point and Policy Enforcement Point are implemented in separate runtimes, possibly from different vendors, require standardization of such tokens. Access concludes by presenting said short-lived token to the server hosting the AAS-resource (step 7).

To strike a balance between granularity and public meta-data exposure, AAS resources may be exposed in a granular manner masking implicit information like the number of Submodels or AAS-Descriptors. The AAS-Registry API completes the Data Consumer's discovery-sequence by linking the Submodels from distributed data sources while at the same time protecting the AAS-Descriptor objects from the public.

The AAS data server and the AAS registry will evaluate and check the claims in the token by the Access Rules as specified later in this specification.

Access token

An AAS Data Server trusts one or several Identity Providers. An Identity Provider may also give the possibility to federate identities to/from other Identity Providers.

Trust means, that for such Identity Provider(s) an AAS Data Server can verify the public key of the Identity Provider. Such public key may be available directly as a key, by a public certificate or by a verifiable credential.

AAS Security uses signed JWT (JSON Web Token) bearer tokens as Access Tokens, as they are defined in RFC

7519.

The JWT is digitally signed according to RFC 7519 with the private key of the Identity Provider. The public key needs to be accessible by the AAS server, such that the JWT can be verified. The key length should follow best practices. Other properties like the lifetime of the token or Proof-of-Possession (RFC 7800) properties can be determined based on the use case.

AAS Security only requires the standard claims of the RFC 9068 Chapter 2.2 Data Structure for the JWT, i.e “iss” (Issuer), “exp” (Expiration time), “aud” (Audience), “sub” (Subject), “client_id” (Client identifier), “iat” (Issued at), “jti” (JWT ID).

Dataspaces should define their specific requirements on “iss”, “sub” and “aud”.

All claims in an Access Token can be used in ABAC access rules as attributes and as described in the Access Rule Model below.

Access rules can also be defined for anonymous access, i.e. no Access Token exists.
This is especially important for the Digital Product Passport to be able to scan the QRCODE according to IEC 61406 with a normal mobile phone camera.

Further definitions can be made by dataspaces. A specific dataspace may define the technology how to get an access token and may also define the attributes which can be used in the ABAC access rules.

This both assures interoperability and makes it possible to integrate new dataspaces in the future.

Access Rule Model (normative)

General

The introduction in Chapter *Introduction* has explained in detail the background for AAS Security. The use of Identity Providers and Authentication Flows have been shown, which provide an access token to the client user application. Such access token contains claims as subject attributes which can be used in the Access Rule Model as defined below.

In addition, the Access Rule Model contains further ABAC attributes like global attributes (also called environmental attributes) or object attributes.

Figure 15 gives an overview of the Access Rule Model. Section BNF grammar of Access Rules defines the text serialization of the Access Rule Model. Section JSON Serialization of Access Rule Model defines the JSON schema of the Access Rule Model.

The Access Rule Model allows to define Access Rules in a modular way, so that parts can be reused.

- Attribute Groups can be defined to reuse combinations of attributes.
- Object Groups can be defined to reuse combinations of objects.
- Access Control Lists (ACLs) can be defined to reuse combinations of attributes and access rights.

Each Access Rule can define an own ACL or reuse an existing one. Each Access Rule can provide access to single objects or object groups. Each Access Rule can define an own Formula or reuse an existing one.

The formula decides, if an Access Rule becomes active or not. Only if the overall result of the Formula is valid and true, the Access Rule becomes active. A formula can include nested boolean expressions, comparisons and string operations. A special operation match allows to work with tuples in lists, e.g. specificAssetId.

Access shall only be granted if there is (at least) one access rule which allows the requested action on the designated AAS object (Route/Identifiable/Referable/Fragment/Descriptor). If there is no access rule, the default behavior shall be to deny access.

This grammar handles both respositories and registries. The symbol FieldIdentifier allows to use attributes of AAS, Submodel, SubmodelElement, ConceptDescription, AAS-Descriptor and Submodel-Descriptor. Depending on the type of repository or registry such attributes are allowed or not.

The AAS Query Language and the AAS Access Rules share the same BNF grammar for formula expressions. In addition to the text below, the further details of the formula expressions are explained in IDTA-01002, Query Language .

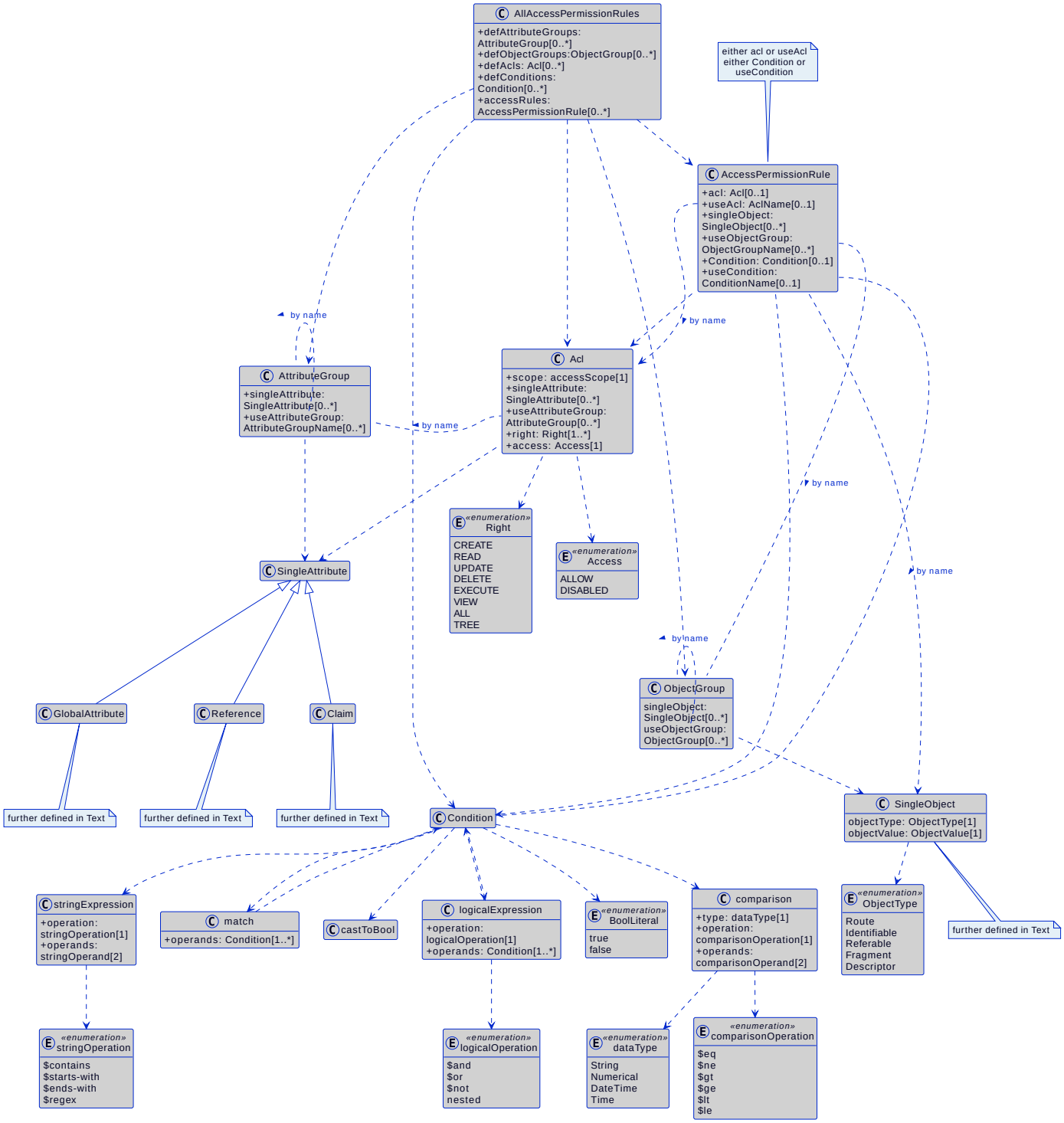

Figure 15. Access Rule Model

Grammar for Access Rule Model

BNF grammar of Access Rules

The following BNF (Backus-Naur-Form) grammar defines the AAS Access Rule Model in a technology neutral form. The grammar includes whitespaces, so that the examples in Examples of Access Rules in text serialization can be validated.

The Security Model is explained step by step in the following Chapter Explanation of the Access Rule Model BNF.

Examples can be found in Examples of Access Rules in text serialization.

The grammar has been tested with BNF playground (https://bnfplayground.pauliankline.com/). BNF playground allows to check and test the grammar itself and if expressions are correct according to the grammar.*

The tested BNF is:

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

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

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

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

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

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

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

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

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

<ReferenceAttribute> ::=
    "REFERENCE" <ws> "(" <ws> <ReferenceLiteral> <ws> ")"

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

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

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

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

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

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

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

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

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

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

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

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


<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>

<DateTimeLiteral> ::= <datetime> <ws>
<TimeLiteral> ::= <time> <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]
<StringLiteral> ::= "\"" ( [A-Z] | [a-z] | [0-9] | "/" | "*" | "[" | "]" | "(" | ")" | " "
| "_" | "@" | "#" | "\\" | "+" | "-" | "." | "," | ":" | "$" | "^" )+ "\""
<ClaimLiteral> ::= <StringLiteral>
<ReferenceLiteral> ::= <StringLiteral>
<RouteLiteral> ::= <StringLiteral>
<IdentifiableLiteral> ::= <StringLiteral>
<ReferableLiteral> ::= <StringLiteral>
<FragmentLiteral> ::= <StringLiteral>
<DescriptorLiteral> ::= <StringLiteral>
<NumericalLiteral> ::= ( "+" | "-" )? ( [0-9]+ ( "." [0-9]* )? | "." [0-9]+ ) ( ( "e" | "E"
)? [0-9]+ )
<HexLiteral> ::= "16#" ( [0-9] | [A-F] )+
<BoolLiteral> ::= "true" | "false"

<FieldIdentifier> ::= <FieldIdentifierAAS> | <FieldIdentifierSM> | <FieldIdentifierSME> |
```

```
<FieldIdentifierCD> | <FieldIdentifierAasDescriptor> | <FieldIdentifierSmDescriptor>
<FieldIdentifierAAS> ::= "$aas#" ( "idShort" | "id" | "assetInformation.assetKind" |
"assetInformation.assetType" | "assetInformation.globalAssetId" | "assetInformation."
<SpecificAssetIdsClause> | "submodels." <ReferenceClause> )
<FieldIdentifierSM> ::= "$sm#" ( <SemanticIdClause> | "idShort" | "id" )
<FieldIdentifierSME> ::= "$sme" ( "." <idShortPath> )? "#" ( <SemanticIdClause> | "idShort"
| "value" | "valueType" | "language" )
<FieldIdentifierCD> ::= "$cd#" ( "idShort" | "id" ) <ws>
<FieldIdentifierAasDescriptor> ::= "$aasdesc#" ( "idShort" | "id" | "assetKind" |
"assetType" | "globalAssetId" | <SpecificAssetIdsClause>  | "endpoints" ( "[" ( [0-9]* )
"]" ) "." <EndpointClause> | "submodelDescriptors" ( "[" ( [0-9]* ) "]" ) "."
<SmDescriptorClause> )
<FieldIdentifierSmDescriptor> ::= "$smdesc#" <SmDescriptorClause>
<SmDescriptorClause> ::= ( <SemanticIdClause> | "idShort" | "id" | "endpoints" ( "[" ( [0-
9]* ) "]" ) "." <EndpointClause> )
<EndpointClause> ::= "interface" | "protocolinformation.href"

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

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

Explanation of the Access Rule Model BNF

General

The AAS Access Rule Model can be used to describe access rules. Whether and how access rules are enforced is beyond the specification of the model for access control. The parties involved are supposed to agree on governance and policies.

The AAS Access Rule Model uses Attribute Based Access Control (ABAC), i.e. Attributes are used in access rules. By ABAC also Role Based Access Control (RBAC) can be implemented by defining role attributes. Subject Attributes and Roles may be provided as claims in Access Tokens.

Attributes in access rules are either claims from an Access Token provided by an Identity Provider, global attributes like actual DATETIME or from a Submodel like a property for a machine state.

Objects to be protected are either API Routes, Identifiables (e.g. AAS or Submodel), Referables (e.g. SubmodelElements), Descriptors or Fragments of all those (e.g. AssetId, SemanticId, SpecificAssetId).

Reuse

The AAS Access Rule Model allows to define modular parts which can be reused in different access rules.

- The first concept of reuse is groups. Both attributes (DEFATTRIBUTES) and objects (DEFOBJECTS) can be combined into related groups, which may also be used in other groups.
- The second concept of reuse is Access Control Lists (DEFACLS).
  An ACL defines which access rights are given for a certain combination of attributes.
- The third concept of reuse are FORMULAs (DEFFORMULAS), which define a Boolean result when an Access

Rule is enabled/disabled. FORMULAs allow to express only expressions with Boolean results, e.g. comparisons. Arithmetic in FORMULAS is currently not supported.

Access Rule Model

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

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

An Access Rule Model defines a list of Access Rules.

For reuse in multiple Access Rules it also contains lists of Attribute Groups (DEFATTRIBUTES), Object Groups (DEFOBJECTS), ACLs (DEFACLS) and Formulas (DEFFORMULAS). Such elements defined for reuse get a name.

One Access Rule must either directly define an ACL or reuse an existing ACL definition.

One access rule may directly list Single Objects or may reuse defined Object Groups.

One access rule may directly define a Formula or may reuse a Formula definition.

An access rule may optionally include a FILTER, which can be used to further restrict the returned objects. A FILTER is an additional FORMULA, which enables by its boolean expression, which part(s) of the given object(s) can be accessed, i.e. without a FILTER the complete objects are accessed. The FILTER contains a FragmentObject, which defines which part of the accessed object has to be filtered. The related FragmentLiteral defines the Prefix of a FieldIdentifier to be filtered, e.g. "$aasdesc#assetInformation.specificAssetIds[]" defines that the specificAssetIds part shall be filtered.

ACL

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

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

<Right> ::=
```

```
    "CREATE" | "READ" | "UPDATE" | "DELETE" | "EXECUTE" | "VIEW" | "ALL" | "TREE"

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

An ACL (Access Control List) defines which access rights are given for a certain combination of attributes.

Attributes can be provided as a list of single attributes and/or as a list of names of other attribute groups.

The rights in ACLs essentially use the CRUDX pattern, i.e. rights for CREATE, READ, UPDATE, DELETE and EXECUTE can be defined. Without an ALLOW rule any access is forbidden by default. For testing and alternative configuration purposes, an access rule may be DISABLED. To avoid complex conflicting situations, deny rules are not supported.

In addition, VIEW allows to see the existence of an element as Id or idShort, but not to read its values and its further attributes.

In addition, TREE defines, that further access rules exist within its child elements, which must be processed.

ALL is an abbreviation to define all rights.

Attributes

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

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

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

<ReferenceAttribute> ::=
    "REFERENCE" <ws> "(" <ws> <ReferenceLiteral> <ws> ")"

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

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

Single Attributes are either claims from an Access Token provided by an Identity Provider, global attributes like actual DATETIME or references to a SubmodelElement e.g. to a property for a machine state.

An Attribute Group defines a list of single attributes and/or a list of names of other attribute groups.

Global Attributes are:

- **LOCALNOW** - Date and time of server according ISO 8601 in local time zone
- **UTCNOW** - Date and time of server according ISO 8601 as UTC time
- **CLIENTNOW** - Date and time of client according ISO 8601, provided as claim in access token

- **ANONYMOUS** - Tag for anonymous and non-authenticated user, i.e. no access token

References in ReferenceAttributes are defined in Section Text Serialization of Values of Type Reference.

Objects

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

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

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

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

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

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

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

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

Objects to be protected are either API Routes, Identifiables (e.g. AAS or Submodel), Referables (e.g. SubmodelElements), Descriptors or Fragments of those (e.g. AssetId, SemanticId, SpecificAssetId).

Routes may use * or end with a *, which means that all routes with a given prefix are valid.

References to IdentifiableObjects, ReferableObjects, FragmentObjects and DescriptorObjects are defined in Section Text Serialization of Values of Type Reference.

An Object Group defines a list of single objects and/or a list of names of other object groups.

Formulas

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

<Condition> ::= <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> )

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

FORMULAs define a logical expression with a Boolean result when an Access Rule is enabled/disabled. FORMULAs allow to express only expressions with Boolean results. Arithmetic in FORMULAS is currently not supported.

Nested logical expressions create combinations of several logical expressions, where any, all or none of the expressions needs to be enabled: **<logicalNestedExpression>, <logicalOrExpression>, <logicalAndExpression>, <logicalNotExpression>**.

String Comparison Operations compare or match the first given argument (left argument) with the second given argument (right argument). **$eq, $ne, $gt, $lt, $ge, $le** make an alphabetic string comparison. $**starts-with, $ends-with, $contains and $regex** check, if the first given argument is part of the second argument or if the first argument matches with the given REGEX.

Numerical Comparison Operations compare the first given argument (left argument) with the second given argument (right argument). Since AAS also supports XS Datatypes Hex, Bool, DateTime and Time, related comparisons are available accordingly.

For specific comparisons datatypes can be casted to the other datatypes.

Specific operations exist to extract parts from DateTime, i.e. $dayOfWeek, $dayOfMonth, $month, $year. This enables access rules related to week days or specific times in the year.

An important special operation is $match, which can be used with any element containing a list of elements, e.g. semanticId[], specificAssetId[], SubmodelElementList or SubmodelElementCollection. The list element is written with [] to express, that $match shall check if a certain expression is true for at least one element in the list.

Text Serialization of Values of Type Reference

References are used in ReferenceAttributes, IdentifiableObjects, ReferableObjects, FragmentObjects and DescriptorObjects for the corresponding ReferenceLiterals, IdentifiableLiterals, ReferableLiterals, FragmentLiterals and DescriptorLiterals.

> Note: V3.1 will include a more detailed grammar for defining <ReferenceAttribute>, <IdentifiableObject>, <ReferableObject>, <FragmentObject> and <DescriptorObject>.

JSON Serialization of Access Rule Model

The AAS Access Rule model can also be defined as a JSON schema. Since the related JSON schema shall also allow automatic code generation, specific constraints must be fulfilled by such JSON schema. The use of "oneof" is limited and type information for objects must be available.

It shall also be possible to check a JSON with the JSON schema.

Examples can be found in Examples of Access Rules in JSON serialization.

The following schema meets such constraints:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Common JSON Schema for AAS Queries and Access Rules",
  "description": "This schema contains the AAS Access Rule Language.",
  "definitions": {
    "standardString": {
      "type": "string",
      "pattern": "^(?!\\$).*"
    },
    "modelStringPattern": {
      "type": "string",
      "pattern":
"^(?:\\$aas#(?:idShort|id|assetInformation\\.assetKind|assetInformation\\.assetType|assetIn
formation\\.globalAssetId|assetInformation\\.(?:specificAssetIds\\[[0-
9]*\\](?:\\.(?:name|value|externalSubjectId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(?:type|value)
)?)?)?)|submodels\\.(?:type|keys\\[\\d*\\](?:\\.(?:type|value))?))|submodels\\.(type|keys\\
[\\d*\\](?:\\.(type|value))?))|(?:\\$sm#(?:semanticId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(typ
e|value))?)?|idShort|id))|(?:\\$sme(?:\\.[a-zA-Z][a-zA-Z0-9_]*\\[[0-9]*\\]?(?:\\.[a-zA-
Z][a-zA-Z0-9_]*\\[[0-
9]*\\]?)*)?#(?:semanticId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(type|value))?)?|idShort|value|v
alueType|language))|(?:\\$cd#(?:idShort|id)))|(?:\\$aasdesc#(?:idShort|id|assetKind|assetTy
pe|globalAssetId|specificAssetIds\\[[0-
9]*\\]?(?:\\.(name|value|externalSubjectId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(type|value))?)
?)?)|endpoints\\[[0-
9]*\\]\\.(interface|protocolinformation\\.href)|submodelDescriptors\\[[0-
9]*\\]\\.(semanticId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(type|value))?)?|idShort|id|endpoints
\\[[0-
9]*\\]\\.(interface|protocolinformation\\.href))))|(?:\\$smdesc#(?:semanticId(?:\\.type|\\.
```

```
keys\\[\\d*\\](?:\\.(type|value))?)?|idShort|id|endpoints\\[[0-
9]*\\]\\.(interface|protocolinformation\\.href)))$"
    },
    "hexLiteralPattern": {
      "type": "string",
      "pattern": "^16#[0-9A-F]+$"
    },
    "dateTimeLiteralPattern": {
      "type": "string",
      "format": "date-time"
    },
    "timeLiteralPattern": {
      "type": "string",
      "pattern": "^[0-9][0-9]:[0-9][0-9](:[0-9][0-9])?$"
    },
    "Value": {
      "type": "object",
      "properties": {
        "$field": {
          "$ref": "#/definitions/modelStringPattern"
        },
        "$strVal": {
          "$ref": "#/definitions/standardString"
        },
        "$attribute": {
          "$ref": "#/definitions/attributeItem"
        },
        "$numVal": {
          "type": "number"
        },
        "$hexVal": {
          "$ref": "#/definitions/hexLiteralPattern"
        },
        "$dateTimeVal": {
          "$ref": "#/definitions/dateTimeLiteralPattern"
        },
        "$timeVal": {
          "$ref": "#/definitions/timeLiteralPattern"
        },
        "$boolean": {
          "type": "boolean"
        },
        "$strCast": {
          "$ref": "#/definitions/Value"
        },
        "$numCast": {
          "$ref": "#/definitions/Value"
        },
        "$hexCast": {
          "$ref": "#/definitions/Value"
        },
        "$boolCast": {
```

```
        "$ref": "#/definitions/Value"
      },
      "$dateTimeCast": {
        "$ref": "#/definitions/Value"
      },
      "$timeCast": {
        "$ref": "#/definitions/Value"
      },
      "$dayOfWeek": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      },
      "$dayOfMonth": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      },
      "$month": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      },
      "$year": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      }
    },
    "oneOf": [
      {
        "required": [
          "$field"
        ]
      },
      {
        "required": [
          "$strVal"
        ]
      },
      {
        "required": [
          "$attribute"
        ]
      },
      {
        "required": [
          "$numVal"
        ]
      },
      {
        "required": [
          "$hexVal"
        ]
      },
      {
        "required": [
          "$dateTimeVal"
        ]
      },
```

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

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

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

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

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

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

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

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

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

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

```
        "items": {
          "$ref": "#/definitions/objectItem"
        },
        "additionalProperties": false
      },
      "USEOBJECTS": {
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "FORMULA": {
        "$ref": "#/definitions/logicalExpression",
        "additionalProperties": false
      },
      "USEFORMULA": {
        "type": "string"
      },
      "FILTER": {
        "type": "object",
            "properties": {
              "FRAGMENT": { "type": "string" },
            "CONDITION": {
              "$ref": "#/definitions/logicalExpression",
                    "additionalProperties": false
                      },
                    "USEFORMULA": { "type": "string" }
                  },
                  "oneOf": [
                    {
                      "required": ["FRAGMENT", "CONDITION"],
                      "not": { "required": ["USEFORMULA"] }
                    },
                    {
                      "required": ["FRAGMENT", "USEFORMULA"],
                      "not": { "required": ["CONDITION"] }
                  }
                  ],
        "additionalProperties": false
      }
    },
    "oneOf": [
      {
        "required": [
          "ACL"
        ]
      },
      {
        "required": [
          "USEACL"
        ]
      }
```

```
      ],
      "oneOf": [
        {
          "required": [
            "OBJECTS"
          ]
        },
        {
          "required": [
            "USEOBJECTS"
          ]
        }
      ],
      "oneOf": [
        {
          "required": [
            "FORMULA"
          ]
        },
        {
          "required": [
            "USEFORMULA"
          ]
        }
      ],
      "additionalProperties": false
    }
  },
  "type": "object",
  "properties": {
    "AllAccessPermissionRules": {
      "type": "object",
      "properties": {
        "DEFATTRIBUTES": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "name": {
                "type": "string"
              },
              "attributes": {
                "type": "array",
                "items": {
                  "$ref": "#/definitions/attributeItem"
                }
              }
            },
            "required": [
              "name",
              "attributes"
            ],
```

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

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

Explanation of the Access Rule Model JSON schema

General

The AAS Access Rule Model can be used to describe access rules. Whether and how access rules are enforced is beyond the specification of the model for access control. The parties involved are supposed to agree on governance and policies.

Access Rule Model

The Access Rules BNF grammar is expressed in JSON schema syntax. Please refer to the detailed step by step explanation of the grammar.

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

```
    "objectItem": {
      "oneOf": [
        {
          "required": [
            "ROUTE"
          ]
        },
        {
          "required": [
            "IDENTIFIABLE"
          ]
        },
        {
          "required": [
            "REFERABLE"
          ]
        },
        {
          "required": [
            "FRAGMENT"
          ]
        },
        {
          "required": [
            "DESCRIPTOR"
          ]
        }
      ],
      "properties": {
        "ROUTE": {
          "type": "string"
        },
        "IDENTIFIABLE": {
          "type": "string"
        },
        "REFERABLE": {
          "type": "string"
        },
        "FRAGMENT": {
          "type": "string"
        },
        "DESCRIPTOR": {
          "type": "string"
        }
      },
      "additionalProperties": false
    },
    "rightsEnum": {
      "type": "string",
      "enum": [
        "CREATE",
        "READ",
```

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

```
    },
    "AccessPermissionRule": {
      "type": "object",
      "properties": {
        "ACL": {
          "$ref": "#/definitions/ACL"
        },
        "USEACL": {
          "type": "string"
        },
        "OBJECTS": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/objectItem"
          },
          "additionalProperties": false
        },
        "USEOBJECTS": {
          "type": "array",
          "items": {
            "type": "string"
          }
        },
        "FORMULA": {
          "$ref": "#/definitions/logicalExpression",
          "additionalProperties": false
        },
        "USEFORMULA": {
          "type": "string"
        },
        "FILTER": {
          "type": "object",
              "properties": {
                "FRAGMENT": { "type": "string" },
              "CONDITION": {
                "$ref": "#/definitions/logicalExpression",
                      "additionalProperties": false
                        },
                      "USEFORMULA": { "type": "string" }
                    },
                    "oneOf": [
                      {
                        "required": ["FRAGMENT", "CONDITION"],
                        "not": { "required": ["USEFORMULA"] }
                      },
                      {
                        "required": ["FRAGMENT", "USEFORMULA"],
                        "not": { "required": ["CONDITION"] }
                    }
                    ],
          "additionalProperties": false
        }
```

```
      },
      "oneOf": [
        {
          "required": [
            "ACL"
          ]
        },
        {
          "required": [
            "USEACL"
          ]
        }
      ],
      "oneOf": [
        {
          "required": [
            "OBJECTS"
          ]
        },
        {
          "required": [
            "USEOBJECTS"
          ]
        }
      ],
      "oneOf": [
        {
          "required": [
            "FORMULA"
          ]
        },
        {
          "required": [
            "USEFORMULA"
          ]
        }
      ],
      "additionalProperties": false
    }
  },
  "type": "object",
  "properties": {
    "AllAccessPermissionRules": {
      "type": "object",
      "properties": {
        "DEFATTRIBUTES": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "name": {
                "type": "string"
```

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

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

```
  },
  "required": [
    "AllAccessPermissionRules"
  ],
  "additionalProperties": false
}
```

Formulas and logical expressions

A detailed step by step explanation follows below the JSON schema code.

```
  "definitions": {
    "standardString": {
      "type": "string",
      "pattern": "^(?!\\$).*"
    },
    "modelStringPattern": {
      "type": "string",
      "pattern":
"^(?:\\$aas#(?:idShort|id|assetInformation\\.assetKind|assetInformation\\.assetType|assetIn
formation\\.globalAssetId|assetInformation\\.(?:specificAssetIds\\[[0-
9]*\\](?:\\.(?:name|value|externalSubjectId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(?:type|value)
)?)?)?)|submodels\\.(?:type|keys\\[\\d*\\](?:\\.(?:type|value))?))|submodels\\.(type|keys\\
[\\d*\\](?:\\.(type|value))?))|(?:\\$sm#(?:semanticId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(typ
e|value))?)?|idShort|id))|(?:\\$sme(?:\\.[a-zA-Z][a-zA-Z0-9_]*\\[[0-9]*\\]?(?:\\.[a-zA-
Z][a-zA-Z0-9_]*\\[[0-
9]*\\]?)*)?#(?:semanticId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(type|value))?)?|idShort|value|v
alueType|language))|(?:\\$cd#(?:idShort|id)))|(?:\\$aasdesc#(?:idShort|id|assetKind|assetTy
pe|globalAssetId|specificAssetIds\\[[0-
9]*\\]?(?:\\.(name|value|externalSubjectId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(type|value))?)
?)?)|endpoints\\[[0-
9]*\\]\\.(interface|protocolinformation\\.href)|submodelDescriptors\\[[0-
9]*\\]\\.(semanticId(?:\\.type|\\.keys\\[\\d*\\](?:\\.(type|value))?)?|idShort|id|endpoints
\\[[0-
9]*\\]\\.(interface|protocolinformation\\.href))))|(?:\\$smdesc#(?:semanticId(?:\\.type|\\.
keys\\[\\d*\\](?:\\.(type|value))?)?|idShort|id|endpoints\\[[0-
9]*\\]\\.(interface|protocolinformation\\.href)))$"
    },
    "hexLiteralPattern": {
      "type": "string",
      "pattern": "^16#[0-9A-F]+$"
    },
    "dateTimeLiteralPattern": {
      "type": "string",
      "format": "date-time"
    },
    "timeLiteralPattern": {
      "type": "string",
      "pattern": "^[0-9][0-9]:[0-9][0-9](:[0-9][0-9])?$"
    },
    "Value": {
```

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

```
      },
      "$year": {
        "$ref": "#/definitions/dateTimeLiteralPattern"
      }
    },
    "oneOf": [
      {
        "required": [
          "$field"
        ]
      },
      {
        "required": [
          "$strVal"
        ]
      },
      {
        "required": [
          "$attribute"
        ]
      },
      {
        "required": [
          "$numVal"
        ]
      },
      {
        "required": [
          "$hexVal"
        ]
      },
      {
        "required": [
          "$dateTimeVal"
        ]
      },
      {
        "required": [
          "$timeVal"
        ]
      },
      {
        "required": [
          "$boolean"
        ]
      },
      {
        "required": [
          "$strCast"
        ]
      },
      {
```

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

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

```
    "type": "object",
    "properties": {
      "$match": {
        "type": "array",
        "minItems": 1,
        "items": {
          "$ref": "#/definitions/matchExpression"
        }
      },
      "$eq": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$ne": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$gt": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$ge": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$lt": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$le": {
        "$ref": "#/definitions/comparisonItems"
      },
      "$contains": {
        "$ref": "#/definitions/stringItems"
      },
      "$starts-with": {
        "$ref": "#/definitions/stringItems"
      },
      "$ends-with": {
        "$ref": "#/definitions/stringItems"
      },
      "$regex": {
        "$ref": "#/definitions/stringItems"
      },
      "$boolean": {
        "type": "boolean"
      }
    },
    "oneOf": [
      {
        "required": [
          "$eq"
        ]
      },
      {
        "required": [
          "$ne"
```

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

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

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

```
        },
        {
          "required": [
            "$le"
          ]
        },
        {
          "required": [
            "$contains"
          ]
        },
        {
          "required": [
            "$starts-with"
          ]
        },
        {
          "required": [
            "$ends-with"
          ]
        },
        {
          "required": [
            "$regex"
          ]
        },
        {
          "required": [
            "$boolean"
          ]
        },
        {
          "required": [
            "$match"
          ]
        }
      ],
      "additionalProperties": false
    }
  }
```

The provided JSON schema includes a *modelStringPattern* starting with a $, so that the syntax of model elements can be checked by the schema, e.g. $sm.idShort.

String, number and boolean are supported by JSON itself. Specific patterns with regular expressions are provided for datatypes Hex, DateTime and Time: *hexLiteralPattern*, *dateTimeLiteralPattern* and *timeLiteralPattern*.

Value is used for comparisons and *StringValue* is used for the specific string operations. For type safety and automatic code generation, each possible type is available as an own object:

- String values can be given by *$field*, $*strVal*, $*attribute* oder $*strCast*.
- Numerical values can be given by $*numVal* or $*numCast*.

- Hex values can be given by $*hexVal* or $*hexCast*.
- Bool values can be given by *$boolean* or $*boolCast*.
- DateTime values can be given by $*dateTimeVal* or $*dateTimeCast*.
- Time values can be given by $*timeVal* or $*timeCast*.

Casts allow any of the values as input and convert this to the given type if possible.

Parts of DateTime values can be extracted as numbers:

- *$dayOfWeek* extracts the day of week as number, starting with 0 for Sunday.
- *$dayOfMonth*, *$month*, *$year* extract the related part as number.

If a conversion or any other operation is invalid, the complete expression has to be treated as invalid. An error message shall be generated and the result of the complete expression becomes FALSE, so that the access is not allowed and/or the filter is empty.

To allow the use of different types in operations, arrays are used for parameters. *comparisonItems* is used for comparisons and *stringItems* is used for specific string operations.

logicalExpressions and *match* can be recursively nested.

Exchange of Access Rules

Business Partners may be interested to exchange Access Rules.

As explained as an example above, a robot manufacturer suggests that for the robot the following roles shall be defined: machine setter, operator and a maintenance role. He also suggests permissions for these roles, e.g. an installer (integrator) does have write-access to the program of the robot, but an operator does not.

Providing such suggested access rules, makes the integration of the operator much easier. The operator can copy and paste the access rules into his AAS implementation and only needs to make adjustments.

The enforcement of Access Rules is implementation specific.

This specification defines the text serialization of access rules and the JSON schema serialization of access rules. The support of both is optional and not mandatory.

The JSON schema serialization is recommended.

Such serialization provides the possibility to exchange access rules. An implementation may even store such serialized access rules in submodel elements BLOB or FILE, so that these can be accessed by the API and can be protected by access rules itself.

API Queries and Access Rules

Version 3.1 of the AAS API defines queries and uses the same grammar and same JSON schema to define queries. The JSON schema serialization is used in the related API operations.

In addition, using the same concepts will be needed for large amounts of AAS data. Querying such large amounts of data and protecting such data with Access Rules requires optimized access of data. Database queries may combine the filter expression provided by the client query with the FOMULAs of the Access Rules on the server.

Summary and Outlook

This document specifies the Access Rule Model for the Asset Administration Shell APIs, including the APIs for repositories and registries.

The general concept of access tokens allows to combine the Access Rule Model with any available security infrastructure in companies, dataspaces or for legal requirements.

The grammar of the Access Rule Model allows to adopt the concepts to any further technology besides AAS HTTP API. For AAS HTTP API a JSON schema is already defined.

The grammar may also be used to create mappings to other access rule languages, e.g. XACML or ODRL.

This document is the base for the upcoming IEC 63278-3 “Security of the Asset Administration Shell”. IEC 63278-3 uses the explained concepts of access token together with the grammar for access rules.

A next version of this document shall define signatures (and possibly encryption) of AAS data. So far this is only possible together with AASX packages, but signatures are also needed when using APIs to exchange data. Some business partners like to copy AAS data to their servers, so that the signature of the originator of the AAS data must be able to be proven by a final receiver.

A next version of this document may also include an API to manage access rules. Since the grammar and the JSON schema are already used for the Query Language in AAS HTTP API 3.1, the needed elements for such additional API are already available.

Annex

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>],
- items existing 0 or more times are enclosed in round brackets and 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 also 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-Address> ::= <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
```

Examples of Access Rules in text serialization

Allow READ access for Anonymous to complete API

```
ACCESSRULE:
  ATTRIBUTES:
    GLOBAL(ANONYMOUS)
  RIGHTS: READ
  ACCESS: ALLOW
  OBJECTS:
    ROUTE "*"
  FORMULA:
    true
```

Allow READ access for Anonymous to list of semanticIDs for submodels

```
ACCESSRULE:
  ATTRIBUTES:
    GLOBAL(ANONYMOUS)
  RIGHTS: READ
  ACCESS: ALLOW
  OBJECTS:
    ROUTE  "*"
  FORMULA:
    $or(
      $sm#semanticId $eq "SemanticID-Nameplate",
      $sm#semanticId $eq "SemanticID-TechnicalData"
    )
```

Allow READ and UPDATE for specific authenticated users

```
ACCESSRULE:
  ATTRIBUTES:
    CLAIM("email")
  RIGHTS: READ UPDATE
  ACCESS: ALLOW
  OBJECTS:
    IDENTIFIABLE "(Submodel)*"
  FORMULA:
```

```
    $and(
      $or(
        $sm#semanticId $eq "SemanticID-Nameplate",
        $sm#semanticId $eq "SemanticID-TechnicalData"
      ),
      $or(
        CLAIM("email") $eq "user1@company1.com",
        CLAIM("email") $eq "user2@company2.com"
      )
    )
```

Allow READ and UPDATE for specific submodel "submodel1"

```
ACCESSRULE:
  ATTRIBUTES:
    CLAIM("email")
  RIGHTS: READ UPDATE
  ACCESS: ALLOW
  OBJECTS:
    IDENTIFIABLE "(Submodel)https://submodel1.company1.com"
  FORMULA:
    CLAIM("email") $eq "user1@company1.com"
```

Reuse of ACL, OBJECT and FORMULA

```
DEFACLS "acl1"
  ATTRIBUTES:
    CLAIM("email")
    GLOBAL(UTCNOW)
  RIGHTS: READ UPDATE
  ACCESS: ALLOW

DEFOBJECTS "Properties"
  REFERABLE "(Submodel)https://s1.com, (Property)p1"
  REFERABLE "(Submodel)https://s1.com, (Property)p2"

DEFFORMULAS "allowSubjectGroup1"
  $and(
    GLOBAL(UTCNOW) $gt "15:00",
    $or(
      CLAIM("email") $eq "user1@company1.com",
      CLAIM("email") $eq "user2@company2.com"
    )
  )

ACCESSRULE:
  USEACLS "acl1"
  OBJECTS:
    USEOBJECTS "Properties"
```

```
  FORMULA:
    USEFORMULAS "allowSubjectGroup1"
```

Example with BusinessPartnerNumber

```
ACCESSRULE:
  ATTRIBUTES:
   CLAIM("BusinessPartnerNumber")
  RIGHTS: READ
  ACCESS: ALLOW
  OBJECTS:
    ROUTE "*"
  FORMULA:
    CLAIM("BusinessPartnerNumber") $eq "BPN1234"
```

Allow READ for all authenticated users of a company for submodels Nameplate and TechnicalData

```
ACCESSRULE:
  ATTRIBUTES:
  CLAIM("email")
  RIGHTS: READ
  ACCESS: ALLOW
  OBJECTS:
    IDENTIFIABLE "(Submodel)*"
  FORMULA:
    $and(
      $or(
        $sm#semanticId $eq "SemanticID-Nameplate",
        $sm#semanticId $eq "SemanticID-TechnicalData"
      ),
      $regex(CLAIM("email"), "[\w\.]+@company\.com")
    )
```

Allow READ to all Submodels with ID pattern for all authenticated users of a company for submodels with Nameplate and TechnicalData from 9:00-17:00

```
ACCESSRULE:
  ATTRIBUTES:
  CLAIM("companyName")
  RIGHTS: READ
  ACCESS: ALLOW
  OBJECTS:
    IDENTIFIABLE "(Submodel)*"
  FORMULA:
    $and(
      $or(
```

```
        $sm#semanticId $eq "SemanticID-Nameplate",
        $sm#semanticId $eq "SemanticID-TechnicalData"
      ),
      CLAIM("companyName") $eq "company1-name",
      $regex($sm#id, "^https://company1.com/.*$"),
      GLOBAL(UTCNOW) $ge "09:00",
      GLOBAL(UTCNOW) $le "17:00"
    )
```

Example with FILTER statement

```
ACCESSRULE:
  ATTRIBUTES:
    CLAIM("BusinessPartnerNumber")
  RIGHTS: READ
  ACCESS: ALLOW
  OBJECTS:
    DESCRIPTOR "(aasDesc)*"
  FORMULA:
    $and(
      CLAIM("BusinessPartnerNumber") $eq "BPNL00000000000A",
      $match(
        $aasdesc#specificAssetIds[].name  $eq "manufacturerPartId",
        $aasdesc#specificAssetIds[].value $eq "99991",
        $aasdesc#specificAssetIds[].externalSubjectId $eq "PUBLIC_READABLE"
      ),
      $match(
        $aasdesc#specificAssetIds[].name $eq "customerPartId",
        $aasdesc#specificAssetIds[].value $eq "ACME001"
      )
    )
  FILTER:
    FRAGMENT "$aasdesc#assetInformation.specificAssetIds[]"
    $or(
      $match(
        $aasdesc#specificAssetIds[].name  $eq "manufacturerPartId",
        $aasdesc#specificAssetIds[].value $eq "99991"
      ),
      $match(
        $aasdesc#specificAssetIds[].name $eq "customerPartId",
        $aasdesc#specificAssetIds[].value $eq "ACME001"
      ),
      $aasdesc#specificAssetIds[].name $eq "partInstanceId",
      $aasdesc#specificAssetIds[].externalSubjectId $eq CLAIM("BusinessPartnerNumber"),
      $aasdesc#specificAssetIds[].externalSubjectId $eq "PUBLIC_READABLE"
    )
```

Allow READ access for Anonymous to complete API

```
{
  "AllAccessPermissionRules": {
    "rules": [
      {
        "ACL": {
          "ATTRIBUTES": [
            {
              "GLOBAL": "ANONYMOUS"
            }
          ],
          "RIGHTS": [
            "READ"
          ],
          "ACCESS": "ALLOW"
        },
        "OBJECTS": [
          {
            "ROUTE": "*"
          }
        ],
        "FORMULA": {
          "$boolean": true
        }
      }
    ]
  }
}
```

Allow READ access for Anonymous to list of semanticIDs for submodels

```
{
  "AllAccessPermissionRules": {
    "rules": [
      {
        "ACL": {
          "ATTRIBUTES": [
            {
              "GLOBAL": "ANONYMOUS"
            }
          ],
          "RIGHTS": [
            "READ"
          ],
          "ACCESS": "ALLOW"
        },
        "OBJECTS": [
```

```
          {
            "ROUTE": "*"
          }
        ],
        "FORMULA": {
          "$or": [
            {
              "$eq": [
                {
                  "$field": "$sm#semanticId"
                },
                {
                  "$strVal": "SemanticID-Nameplate"
                }
              ]
            },
            {
              "$eq": [
                {
                  "$field": "$sm#semanticId"
                },
                {
                  "$strVal": "SemanticID-TechnicalData"
                }
              ]
            }
          ]
        }
      }
    ]
  }
}
```

Allow READ and UPDATE for specific authenticated users

```
{
  "AllAccessPermissionRules": {
    "rules": [
      {
        "ACL": {
          "ATTRIBUTES": [
            {
              "CLAIM": "email"
            }
          ],
          "RIGHTS": [
            "READ",
            "UPDATE"
          ],
          "ACCESS": "ALLOW"
```

```
    },
    "OBJECTS": [
      {
        "IDENTIFIABLE": "(Submodel)*"
      }
    ],
    "FORMULA": {
      "$and": [
        {
          "$or": [
            {
              "$eq": [
                {
                  "$field": "$sm#semanticId"
                },
                {
                  "$strVal": "SemanticID-Nameplate"
                }
              ]
            },
            {
              "$eq": [
                {
                  "$field": "$sm#semanticId"
                },
                {
                  "$strVal": "SemanticID-TechnicalData"
                }
              ]
            }
          ]
        },
        {
          "$or": [
            {
              "$eq": [
                {
                  "$attribute": {
                    "CLAIM": "email"
                  }
                },
                {
                  "$strVal": "user1@company1.com"
                }
              ]
            },
            {
              "$eq": [
                {
                  "$attribute": {
                    "CLAIM": "email"
                  }
```

```
                    },
                    {
                      "$strVal": "user2@company2.com"
                    }
                  ]
                }
              ]
            }
          ]
        }
      }
    ]
  }
}
```

Allow READ and UPDATE for specific submodel "submodel1"

```
{
  "AllAccessPermissionRules": {
    "rules": [
      {
        "ACL": {
          "ATTRIBUTES": [
            {
              "CLAIM": "email"
            }
          ],
          "RIGHTS": [
            "READ",
            "UPDATE"
          ],
          "ACCESS": "ALLOW"
        },
        "OBJECTS": [
          {
            "IDENTIFIABLE": "(Submodel)https://submodel1.company1.com"
          }
        ],
        "FORMULA": {
          "$eq": [
            {
              "$attribute": {
                "CLAIM": "email"
              }
            },
            {
              "$strVal": "user1@company1.com"
            }
          ]
        }
```

```
      }
    ]
  }
}
```

Reuse of ACL, OBJECT and FORMULA

```
{
  "AllAccessPermissionRules": {
    "DEFACLS": [
      {
        "name": "acl1",
        "acl": {
          "ATTRIBUTES": [
            {
              "CLAIM": "email"
            }
          ],
          "RIGHTS": [
            "READ",
            "UPDATE"
          ],
          "ACCESS": "ALLOW"
        }
      }
    ],
    "DEFOBJECTS": [
      {
        "name": "Properties",
        "objects": [
          {
            "REFERABLE": "(Submodel)https://s1.com, (Property)p1"
          },
          {
            "REFERABLE": "(Submodel)https://s1.com, (Property)p2"
          }
        ]
      }
    ],
    "DEFFORMULAS": [
      {
        "name": "allowSubjectGroup1",
        "formula": {
          "$and": [
            {
              "$eq": [
                {
                  "$attribute": {
                    "GLOBAL": "UTCNOW"
                  }
```

```
                },
                {
                  "$timeVal": "15:00"
                }
              ]
            },
            {
              "$or": [
                {
                  "$eq": [
                    {
                      "$attribute": {
                        "CLAIM": "email"
                      }
                    },
                    {
                      "$strVal": "user1@company1.com"
                    }
                  ]
                },
                {
                  "$eq": [
                    {
                      "$attribute": {
                        "CLAIM": "email"
                      }
                    },
                    {
                      "$strVal": "user2@company2.com"
                    }
                  ]
                }
              ]
            }
          ]
        }
      }
    ],
    "rules": [
      {
        "USEACL": "acl1",
        "USEOBJECTS": [ "Properties" ],
        "USEFORMULA": "allowSubjectGroup1"
      }
    ]
  }
}
```

Example with BusinessPartnerNumber

```
{
  "AllAccessPermissionRules": {
    "rules": [
      {
        "ACL": {
          "ATTRIBUTES": [
            {
              "CLAIM": "BusinessPartnerNumber"
            }
          ],
          "RIGHTS": [
            "READ"
          ],
          "ACCESS": "ALLOW"
        },
        "OBJECTS": [
          {
            "ROUTE": "*"
          }
        ],
        "FORMULA": {
          "$eq": [
            {
              "$attribute": {
                "CLAIM": "BusinessPartnerNumber"
              }
            },
            {
              "$strVal": "BPN1234"
            }
          ]
        }
      }
    ]
  }
}
```

Allow READ for all authenticated users of a company for submodels Nameplate and TechnicalData

```
{
  "AllAccessPermissionRules": {
    "rules": [
      {
        "ACL": {
          "ATTRIBUTES": [
            {
              "CLAIM": "email"
```

```
        }
      ],
      "RIGHTS": [
        "READ"
      ],
      "ACCESS": "ALLOW"
    },
    "OBJECTS": [
      {
        "IDENTIFIABLE": "(Submodel)*"
      }
    ],
    "FORMULA": {
      "$and": [
        {
          "$or": [
            {
              "$eq": [
                {
                  "$field": "$sm#semanticId"
                },
                {
                  "$strVal": "SemanticID-Nameplate"
                }
              ]
            },
            {
              "$eq": [
                {
                  "$field": "$sm#semanticId"
                },
                {
                  "$strVal": "SemanticID-TechnicalData"
                }
              ]
            }
          ]
        },
        {
          "$regex": [
            {
              "$attribute": {
                "CLAIM": "email"
              }
            },
            {
              "$strVal": "[\\w\\.]+@company\\.com"
            }
          ]
        }
      ]
    }
```

```
      }
    ]
  }
}
```

Allow READ to all Submodels with ID pattern for all authenticated users of a company for submodels with Nameplate and TechnicalData from 9:00-17:00

```
{
  "AllAccessPermissionRules": {
    "rules": [
      {
        "ACL": {
          "ATTRIBUTES": [
            {
              "CLAIM": "companyName"
            }
          ],
          "RIGHTS": [
            "READ"
          ],
          "ACCESS": "ALLOW"
        },
        "OBJECTS": [
          {
            "ROUTE": "*"
          }
        ],
        "FORMULA": {
          "$and": [
            {
              "$or": [
                {
                  "$eq": [
                    {
                      "$field": "$sm#semanticId"
                    },
                    {
                      "$strVal": "SemanticID-Nameplate"
                    }
                  ]
                },
                {
                  "$eq": [
                    {
                      "$field": "$sm#semanticId"
                    },
                    {
                      "$strVal": "SemanticID-TechnicalData"
                    }
```

```
        ]
      }
    ]
  },
  {
    "$eq": [
      {
        "$attribute": {
          "CLAIM": "companyName"
        }
      },
      {
        "$strVal": "company1-name"
      }
    ]
  },
  {
    "$regex": [
      {
        "$attribute": {
          "REFERENCE": "(Submodel)*#Id"
        }
      },
      {
        "$strVal": "^https://company1.com/.*$"
      }
    ]
  },
  {
    "$ge": [
      {
        "$attribute": {
          "GLOBAL": "UTCNOW"
        }
      },
      {
        "$timeVal": "09:00"
      }
    ]
  },
  {
    "$le": [
      {
        "$attribute": {
          "GLOBAL": "UTCNOW"
        }
      },
      {
        "$timeVal": "17:00"
      }
    ]
  }
```

```
          ]
        }
      }
    ]
  }
}
```

Example with FILTER statement

```
{
  "AllAccessPermissionRules": {
    "rules": [
      {
        "ACL": {
          "ATTRIBUTES": [
            {
              "CLAIM": "BusinessPartnerNumber"
            }
          ],
          "RIGHTS": [
            "READ"
          ],
          "ACCESS": "ALLOW"
        },
        "OBJECTS": [
          {
            "DESCRIPTOR": "(aasdesc)*"
          }
        ],
        "FORMULA": {
          "$and": [
            {
              "$eq": [
                {
                  "$attribute": {
                    "CLAIM": "BusinessPartnerNumber"
                  }
                },
                {
                  "$strVal": "BPNL00000000000A"
                }
              ]
            },
            {
              "$match": [
                {
                  "$eq": [
                    {
                      "$field": "$aasdesc#specificAssetIds[].name"
                    },
```

```
                {
                  "$strVal": "manufacturerPartId"
                }
              ]
            },
            {
              "$eq": [
                {
                  "$field": "$aasdesc#specificAssetIds[].value"
                },
                {
                  "$strVal": "99991"
                }
              ]
            },
            {
              "$eq": [
                {
                  "$field": "$aasdesc#specificAssetIds[].externalSubjectId"
                },
                {
                  "$strVal": "PUBLIC_READABLE"
                }
              ]
            }
          ]
        },
        {
          "$match": [
            {
              "$eq": [
                {
                  "$field": "$aasdesc#specificAssetIds[].name"
                },
                {
                  "$strVal": "customerPartId"
                }
              ]
            },
            {
              "$eq": [
                {
                  "$field": "$aasdesc#specificAssetIds[].value"
                },
                {
                  "$strVal": "ACME001"
                }
              ]
            }
          ]
        }
      ]
```

```
    },
    "FILTER": {
      "FRAGMENT": "$aasdesc#assetInformation.specificAssetIds[]",
      "CONDITION": {
        "$or": [
          {
            "$match": [
              {
                "$eq": [
                  {
                    "$field": "$aasdesc#specificAssetIds[].name"
                  },
                  {
                    "$strVal": "manufacturerPartId"
                  }
                ]
              },
              {
                "$eq": [
                  {
                    "$field": "$aasdesc#specificAssetIds[].value"
                  },
                  {
                    "$strVal": "99991"
                  }
                ]
              }
            ]
          },
          {
            "$match": [
              {
                "$eq": [
                  {
                    "$field": "$aasdesc#specificAssetIds[].name"
                  },
                  {
                    "$strVal": "customerPartId"
                  }
                ]
              },
              {
                "$eq": [
                  {
                    "$field": "$aasdesc#specificAssetIds[].value"
                  },
                  {
                    "$strVal": "ACME001"
                  }
                ]
              }
            ]
```

```
              },
              {
                "$eq": [
                  {
                    "$field": "$aasdesc#specificAssetIds[].name"
                  },
                  {
                    "$strVal": "partInstanceId"
                  }
                ]
              },
              {
                "$eq": [
                  {
                    "$field": "$aasdesc#specificAssetIds[].externalSubjectId"
                  },
                  {
                    "$attribute": {
                      "CLAIM": "BusinessPartnerNumber"
                    }
                  }
                ]
              },
              {
                "$eq": [
                  {
                    "$field": "$aasdesc#specificAssetIds[].externalSubjectId"
                  },
                  {
                    "$strVal": "PUBLIC_READABLE"
                  }
                ]
              }
            ]
          }
        }
      ]
    }
}
```

Change Log

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

Bugfixes:

- changed: Removed incorrect but required whitespaces from grammar and examples [#477 of API](https://github.com/admin-shell-io/aas-specs-api/issues/477)
- changed: fixed idShortPath definition in the BNF Grammar for the Query Language [#34](https://github.com/admin-shell-io/aas-specs-security/issues/34)
- changed: text serialization of Reference does not follow text serialization of Part 1 of References, the updated grammar will be added in the V3.1 [#33](https://github.com/admin-shell-io/aas-specs-security/issues/33)

Minor Changes:

- changed: correct example in Annex
- removed: remove <FieldIdentifierString> in grammar and use <FieldIdentifier> directly

Changes V3.0

This is the first release

Bibliography

[1] IDTA 01001 Specification of the Asset Administration Shell. Part 1: Metamodel, Version 3.1. Industrial Digital Twin Association (IDTA), January 2025. Online. Available: https://industrialdigitaltwin.io/aas-specifications/IDTA-01001/v3.1/index.html

[2] IDTA 01002 Specification of the Asset Administration Shell. Part 2: Application Programming Interfaces, Version 3.1. Industrial Digital Twin Association (IDTA), January 2025. Online. Available: https://industrialdigitaltwin.io/aas-specifications/IDTA-01002/v3.1/index.html

[3] IDTA 01003-a Specification of the Asset Administration Shell. Part 3a: Data Specification – IEC 61360, Version 3.1. Industrial Digital Twin Association (IDTA), January 2025. Online. Available: https://industrialdigitaltwin.io/aas-specifications/IDTA-01003-a/v3.1/index.html

[4] IDTA-01005 Specification of the Asset Administration Shell. Part 5: Package File Format, Version 3.1. Industrial Digital Twin Association (IDTA), January 2025. Online. Available: https://industrialdigitaltwin.io/aas-specifications/IDTA-01005/v3.1/index.html

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

[6] Secure Download Service, Plattform Industrie 4.0, October 2020. Online. Available: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/secure_downloadservice.pdf

[7] Dataspace Protocol 2024-01. Edited by Sebastian Steinbuß, Julia Pampus, James Marino. International Dataspaces Association. Online. Available: https://docs.internationaldataspaces.org/ids-knowledgebase/v/dataspace-protocol

[8] Vincent Hu, David Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller and Karen Scarfone, “Guide to Attribute Based Access Control (ABAC) Definition and Considerations”, NIST Special Publication 800-162, Jan. 2014. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-162

[9] “eXtensible Access Control Markup Language (XACML)”, OASIS Standard, Version 3.0, 22. Jan. 2013, [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf

www.industrialdigitaltwin.org