What is the Asset Administration Shell from a technical perspective?

As of 04/2021

This document has been developed by the Task Force “Secure Asset Administration Shell” of Plattform Industrie 4.0. The Task Force consists of members from the Working Groups “Reference Architectures, Standards and Standardisation” and “Security of Networked Systems” (see below). This document summarizes the common understanding of the Task Force, what are the different shapes of an Asset Administration Shell. This document describes a technical perspective and does not include e.g. value propositions or potentials for new business models.

Which documents define the Asset Administration Shell (AAS)?

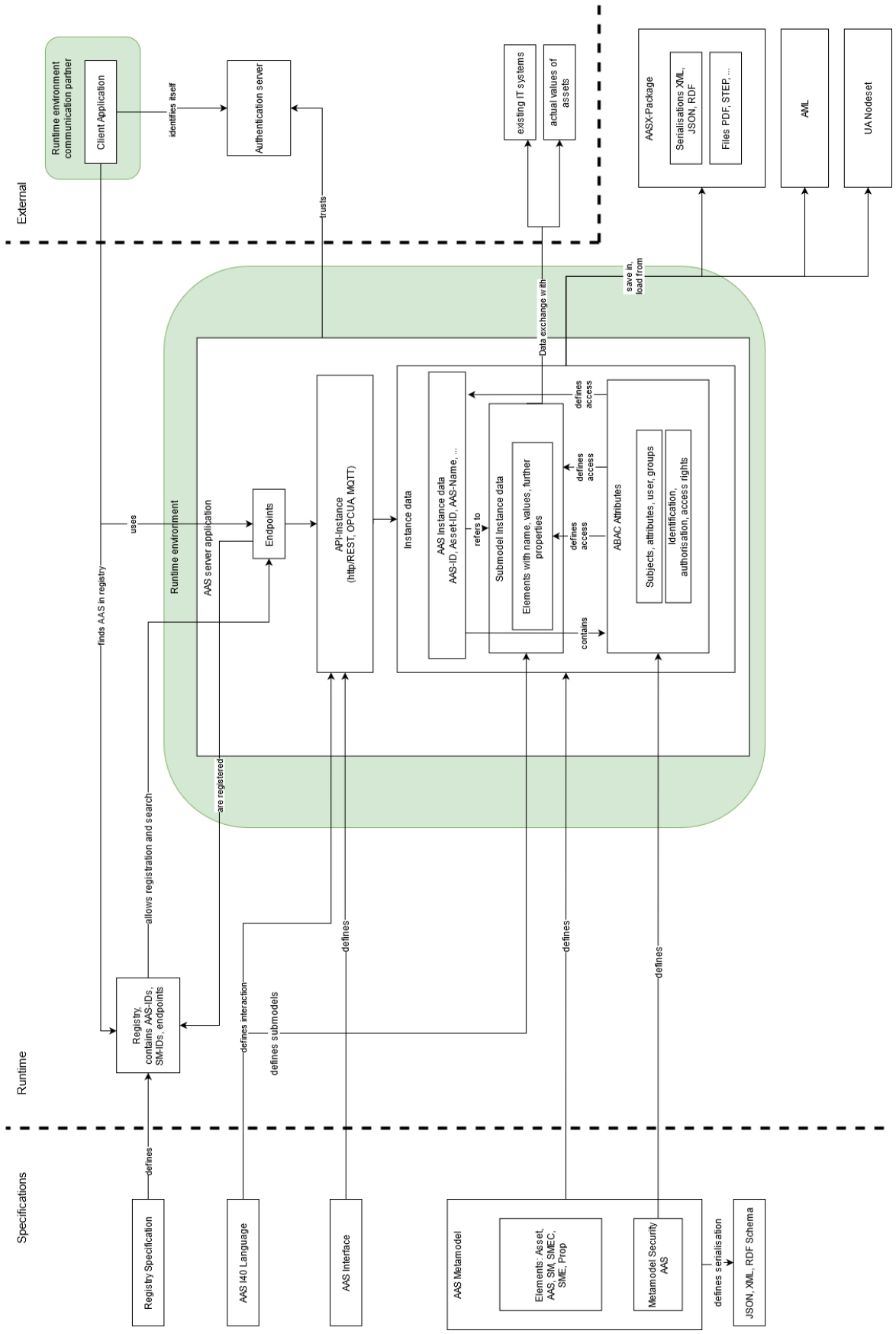
- Details of the Asset Administration Shell (AAS)
  - Part 1: Metamodel, AASX Package and serializations to XML/JSON/RDF/AML
  - Part 2: http/REST API and further APIs, infrastructure and registry
- I40 Language
- OPC UA Companion Specification I4AAS
- Usage View of the AAS
- Asset Administration Shell - Reading Guide

The Asset Administration Shell is a generic term for:

- **AAS metamodel** with UML class diagrams
- **AAS metamodel serialization** with schemas for XML, JSON or RDF
- **AAS instance data** (with static and optionally dynamic data)
- **AAS instance data serializations** to XML, JSON, RDF, AML or OPC UA Nodeset
- **AAS-Package-Container** (.AASX)
- **AAS server application**
- **AAS API** (e.g. by http/REST, OPC UA or MQTT)

Extensions are:

- AAS OPC UA Model I4AAS
- AAS I40 language
- AAS registry


External
Runtime environment communication partner
Client Application
identifies itself
Authentication server
trusts
existing IT systems
actual values of assets
AASX-Package
Serialisations XML, JSON, RDF
Files PDF, STEP, ...
AML
UA Nodeset
save in, load from
Data exchange with
Runtime environment
AAS server application
Endpoints
uses
API-Instance (http/REST, OPCUA, MQTT)
Instance data
AAS Instance data AAS-ID, Asset-ID, AAS-Name, ...
refers to
Submodel Instance data
Elements with name, values, further properties
ABAC Attributes
Subjects, attributes, user groups
Identification, authorisation, access rights
defines access
contains
finds AAS in registry
are registered
allows registration and search
Registry, contains AAS-IDs, SM-IDs, endpoints
defines interaction
defines submodels
defines
Runtime
Specifications
Registry Specification
AAS I40 Language
AAS Interface
AAS Metamodel
Elements: Asset, AAS, SM, SMEC, SME, Prop
Metamodel Security AAS
defines serialisation
JSON, XML, RDF Schema

The Figure shows the aspects described above and graphically displays their relationships.

The **AAS metamodel** defines the available elements to model AAS metamodel instances, e.g. Asset, AssetAdminstrationShell (AAS), Submodel (SM), SubmodelElementCollection (SMEC), Property or additional SubmodelElement(s) (SME). The technology neutral AAS metamodel (UML class diagram) is serialized and stored as schema files for XML, JSON or RDF.

AAS instance data use the defined elements of the metamodel to specify asset types or asset instances. The description of submodels includes both submodel templates and submodel instances. Submodel templates typically do not include values, but submodel instances typically include values for submodel elements. Asset, AAS and submodels include a worldwide unique identifier in the AAS instance data. AAS instance data may also include AAS operations or AAS events.

AAS instance data can be filled with data by different means. On one hand serializations may be loaded. On the other hand, data may be loaded on demand from existing IT systems to use them according to the AAS meta-model. In such case the information of submodels may even be requested and combined from different IT systems. Data may also include dynamic values which can be read from the related assets (e.g. automation devices).

AAS instance data may be transformed into different **AAS instance data serializations** like XML, JSON, RDF, AML or OPC UA nodeset (according **AAS OPC UA model I4AAS**). These serializations can be stored as files to be loaded again later. A special serialisation of the AAS instance data is the **AAS Package Container** (file in AASX format). An AAS Package Container can include the serialized XML or JSON and additional files of the AAS instance data, e.g. PDFs.

AAS instance data may be loaded into a specific **AAS server application** to be instantiated in memory. Alternatively, an AAS server application may also manage several AAS instances. AAS server applications may also host AAS instance data of both complete AAS and/or of separate submodels. In an Industrie 4.0 system multiple decentral AAS server applications interact with each other.

The access to AAS server applications can be made by means of the **AAS API**. AAS server applications may offer a http/REST API and a OPC UA API (according the AAS OPC UA model I4AAS). An AAS server application with http/REST or OPC UA API is a so called reactive AAS. Further technologies like MQTT may also provide such AAS API accordingly.

Submodel instance data are either part of AAS instance data or are separate unique submodel instance data. Submodel instances can be serialized separately and can be hosted on separate AAS server applications, which can be also accessed by the AAS API.

The AAS API specifies services and API operations, which include reading and writing of data and also calling methods (AAS operations).

Each AAS Server Application and each submodel may have an **endpoint** for the API access. Such an endpoint defines an address to access the AAS Server Application by the AAS API specified protocols and operations.
Examples for endpoints are „https://admin-shell-io.com:51410“ for http/REST and „opc.tcp://192.168.1.40:4840“ for OPC UA.

The **AAS I40 language** specifies interaction patterns and submodels, which communicate proactively by the defined interfaces via http/REST, OPC UA or MQTT. An AAS Server Application with a corresponding interface for the I40 language is at the same time a client and a server and is a so called proactive AAS.

In addition, a registry specification is developed to register endpoints and descriptions for AAS Server Applications.

Such AAS Registry contains Endpoints and IDs of AAS instance data and IDs of submodel instance data from a related AAS server application. These entries can be found by requesting the registry. Currently there is no definition available, how the end-points of the registries themselves are found in a manufacturer neutral way.

Further elements are needed to build a real executable system.

An **execution environment** (e.g. hardware, CPU, OS, memory, hard disk) is necessary, into which **computer programs** can be deployed and which will execute these. The AAS server application and related client applications are such computer programs. Serializations and AAS Package Containers can also be stored into and loaded from the execution environment.

Several AAS Server Applications can be implemented within one computer program. Several computer programs can be executed in one execution environment.

Security

Which documents define the security of the AAS?

- Security of the Asset Administration Shell
- Details of the Asset Administration Shell Part 1, Chapter 6, Attribute Based & Role Based Access
- Access control for Industrie 4.0 components for application by manufacturers, operators and integrators
- Secure Download Service

Security of the AAS

Security is a must for all concepts implementting Industrie 4.0. Each practical implementation must be oriented on the security needs of the applications, e.g. protection of data, safety or know how. Even if security must always be conceptually provided, it may be implemented to different degrees depending on security needs, application or use case.

The **AAS metamodel** includes elements to model an **Attribute Based Access Control** (ABAC). This includes the identification of users and user groups, of additional attributes and associated rights, even including the possibility to define the granularity per single element of an AAS. The parameterization of the access control is made via the specified security model. In addition, properties for the secure identification of users and client applications are defined, which **authentication** and **authorization** are based on. Client applications and AAS server applications will prove their authenticity by such properties.

The **AAS Package Container** (.AASX) has been realized using the Open Packaging Conventions (OPC) Format. The OPC Format describes its included elements in XML and supports security elements for authentication according to XMLSIG. An AAS Package Container cannot protect its **authenticity**, but the authenticity can be verified during read of the package. An AAS Package Container cannot protect its **confidentiality** by itself. In that case the parts to be protected or the complete container file must be encrypted. Such encryption can be done for a specific receiver or, in case of digital rights management (DRM), for a group of receivers, which may even not be known at the time of encryption.

The **AAS server application** provides the technical implementation of the **AAS API**. The technically correct and **secure implementation** of the AAS server application and of the **execution environment** is the prerequisite of a continuously secure operation. A secure implementation of AAS server application and execution environment has to rely on a secure development process e.g. according to IEC 62443-4-1 and the secure operation e.g. according to ISO 27000. The **access control** shall be implemented according to the rules defined in the AAS itself and/or by the use cases for the AAS server application. The AAS server application either needs to authenticate its

communication partners by itself or it needs to use a central authentication service. In a next step the AAS server application and the access control will apply the defined rules of the security model.

The communication using the **AAS API** must use protocols which support the needed **security mechanisms**. http/REST over HTTPS and OPC UA support these mechanisms. Depending on the protocol e.g. TLS (Transportation Layer Security) will be used on the transport layer. Methods for authentication can use X.509 certificates and/or established methods for **Identity- and Access Management** (IAM) with specific endpoints or tokens. The AAS server application will offer the **AAS API** in a way, that the authenticity of communication partners plus the disclosure and the authenticity of the communication itself will be realized according to the protection needs. The protection needs are based on the security needs and the use cases, so that security measures can be taken to different degrees.

It is the authors understanding that this text summarizes the actual state of discussion of Plattform Industrie 4.0.

Authors

Sebastian Bader	Fraunhofer IAIS
Vanessa Bellinghausen	Bundesamt für Sicherheit in der Informationstechnik
Dr. Birgit Boss	Robert Bosch GmbH
André Braunmandl	Bundesamt für Sicherheit in der Informationstechnik
Dr. Gerd Brost	Fraunhofer AISEC
Björn Flubacher	Bundesamt für Sicherheit in der Informationstechnik
Kai Garrels	ABB STOTZ-KONTAKT GmbH
Dr. Michael Hoffmeister	Festo SE & Co. KG
Dr. Lutz Jänicke	PHOENIX CONTACT GmbH & Co. KG
Michael Jochem	Robert Bosch GmbH
Andreas Orzelski	PHOENIX CONTACT GmbH & Co. KG
Jens Vialkowitsch	Robert Bosch GmbH
Thomas Walloschke	Industrie KI GmbH
Jörg Wende	IBM Deutschland GmbH

Contact: Office of Plattform Industrie 4.0, Bülowstraße 78, 10783 Berlin

geschäftsstelle@plattform-i40.de

www.plattform-i40.de