Fraunhofer
ISST

FRAUNHOFER-INSTITUT FÜR SOFTWARE- UND SYSTEMTECHNIK, ISST

IMPLEMENTATION CONCEPT

ISST Base Connector: Dataspace Connector

VERSION 4.0

IMPLEMENTATION CONCEPT

ISST Base Connector: Dataspace Connector

Heinrich Pettenpohl

Fraunhofer-Institut für Software- und Systemtechnik ISST in Dortmund.

Content

1 Introduction

This document addresses the requirements of the IDS-ready component review based on "Base Security Profile" by describing the concepts of the envisioned fulfillment by the Dataspace Connector implementation.

The Dataspace Connector is a Java application following basic Spring Boot functionalities and architecture patterns. It currently supports the IDS Information Model version 4.0.0 and integrates the IDS Connector Framework Library for IDS functionalities and message handling.

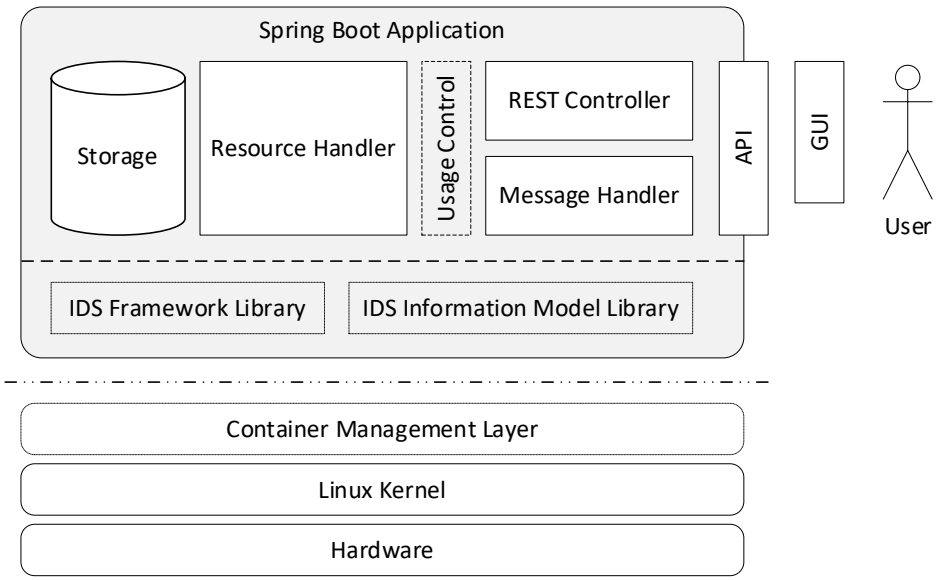

Figure 1.1 Architecture Overview of the Dataspace Connector

It provides a REST API for loading, updating, and deleting IDS persisted resources in a local database. Next to the internal database, external REST endpoints may be connected as data sources. The Dataspace Connector supports IDS conform message handling with other IDS connectors and IDS brokers and implements usage control for seven IDS usage policy patterns (Bader, 2020):

- Allow the Usage of the Data
- Interval-restricted Data Usage
- Duration-restricted Data Usage
- Restricted Number of Usages
- Use data and delete it after
- Local Logging
- Remote Notifications

The Dataspace Connector is an open source project on GitHub, whose development is currently being promoted in order to meet the requirements of industrial use cases. A detailed description of the implementation and its usage is available in the provided Readme and in the Wiki. A comprehensive description in a separate document is planned.

2 Overview table

ID	Criteria Title	Base	Trust	Trust+
IDS Specification (Component: Connector)				
Communication Integrity				
COM 01	Protected connection	x	x	x
COM 02	Mutual authentication	x	x	x
COM 03	State of the art cryptography	x	x	x
COM 04	Remote attestation	-	x	x
COM 05	Platform integrity	-	x	x
COM 06	Configuration and app integrity	-	-	x
Data Usage Control				
USC 01	Definition of usage policies	x	x	x
USC 02	Sending of usage policies	-	x	x
USC 03	Usage policy enforcement	-	x	x
USC 04	Usage policy changes	-	x	x
USC 05	Usage policy changes by administrator	-	-	x
Information Model				
INF 01	Self-Description (at Connector)	x	x	x
INF 02	Self-Description (at Broker)	x	x	x
INF 03	Self-Description content	x	x	x
INF 04	Self-Description evaluation	x	x	x
INF 05	Dynamic attribute tokens	x	x	x
Identity and Access Management				
IAM 01	Connector identifier	x	x	x
IAM 02	Time Service	x	x	x
IAM 03	Online certificate status check	x	x	x
IAM 04	Attestation of dynamic attributes	x	x	x
Broker Service				
BRK 01	Broker service inquiries	x	x	x
BRK 02	Broker registration	x	x	x
BRK 03	Broker registration update	x	x	x
Operating System				
OS 01	Container support	x	x	x
OS 02	App separation	-	x	x
OS 03	Service authenticity and integrity	-	x	x
OS 04	System component authenticity and integrity	-	x	x
OS 05	Container separation	-	x	x
OS 06	Backup encryption	-	x	x
Apps and App Store Connection				
APS 01	App signature	x	x	x
APS 02	App signature verification	x	x	x
APS 03	Terms of use	-	x	x
APS 04	Requirements for the runtime environment	-	x	x
APS 05	App installation	x	x	x
APS 06	App Store	x	x	x
Data Usage Transparency				
AUD 01	Access control logging	x	x	x
AUD 02	Data access logging	x	x	x
AUD 03	Configuration changes logging	x	x	x
AUD 04	Resource availability logging	-	x	x

62433-4-2				
IAC: Identification and authentication control				
CR 1.1	Human user identification and authentication	x	x	x
CR 1.1 (1)	Unique identification and authentication	x	x	x
CR 1.1 (2)	Multifactor authentication for all interfaces	-	x	x
CR 1.2	Software process and device identification and authentication	x	x	x
CR 1.2 (1)	Unique identification and authentication	x	x	x
CR 1.3	Account management	x	x	x
CR 1.4	Identifier management	x	x	x
CR 1.5	Authenticator management	x	x	x
CR 1.5 (1)	Hardware security for authenticators	-	x	x
CR 1.7	Strength of password-based authentication	x	x	x
CR 1.7 (1)	Password generation and lifetime restrictions for human users	-	x	x
CR 1.8	Public key infrastructure certificates	x	x	x
CR 1.9	Strength of public key-based authentication	x	x	x
CR 1.9 (1)	Hardware security for public key-based authentication	-	x	x
CR 1.10	Authenticator feedback	x	x	x
CR 1.11	Unsuccessful login attempts	x	x	x
CR 1.12	System use notification	x	x	x
CR 1.14	Strength of symmetric key-based authentication	x	x	x
CR 1.14 (1)	Hardware security for symmetric key-based authentication	-	x	x
UC: Use Control				
CR 2.1	Authorization enforcement	x	x	x
CR 2.1 (1)	Authorization enforcement for all users (humans, software processes and devices)	-	x	x
CR 2.1 (2)	Permission mapping to roles	-	x	x
CR 2.1 (3)	Supervisor override	-	x	x
CR 2.2	Wireless use control	x	x	x
CR 2.5	Session lock	x	x	x
CR 2.6	Remote session termination	-	x	x
CR 2.7	Concurrent session control	-	x	x
CR 2.8	Auditable events	x	x	x
CR 2.9	Audit storage capacity	x	x	x
CR 2.9 (1)	Warn when audit record storage capacity threshold reached	-	x	x
CR 2.10	Response to audit processing failures	x	x	x
CR 2.11	Timestamps	x	x	x
CR 2.11 (1)	Time synchronization	-	x	x
CR 2.11 (2)	Protection of time source integrity	-	x	x
CR 2.12	Non-repudiation	x	x	x
CR 2.12 (1)	Non-repudiation for all users	-	x	x
SI: System integrity				
CR 3.1	Communication integrity	x	x	x

CR 3.1 (1)	Communication authentication	x	x	x
CR 3.3	Security functionality verification	x	x	x
CR 3.4	Software and information integrity	x	x	x
CR 3.4 (1)	Authenticity of software and information	-	x	x
CR 3.4 (2)	Automated notification of integrity violations	-	x	x
CR 3.5	Input validation	x	x	x
CR 3.6	Deterministic output	x	x	x
CR 3.7	Error handling	x	x	x
CR 3.8	Session integrity	x	x	x
CR 3.9	Protection of audit information	-	x	x
DC: Data confidentiality				
CR 4.1	Information confidentiality	x	x	x
CR 4.2	Information persistence	-	x	x
CR 4.2 (1)	Erase of shared memory resources	x	x	x
CR 4.2 (2)	Erase verification	-	x	x
CR 4.3	Use of cryptography	x	x	x
RDF: Restricted data flow				
CR 5.1	Network segmentation	x	x	x
TRE: Timely response to events				
CR 6.1	Audit log accessibility	x	x	x
CR 6.1 (1)	Programmatic access to audit logs	-	x	x
CR 6.2	Continuous monitoring	-	x	x
RA: Resource availability				
CR 7.1	Denial of service protection	x	x	x
CR 7.1 (1)	Manage communication load from component	-	x	x
CR 7.2	Resource management	x	x	x
CR 7.3	Control system backup	x	x	x
CR 7.3 (1)	Backup integrity verification	-	x	x
CR 7.4	Control system recovery and reconstitution	x	x	x
CR 7.6	Network and security configuration settings	x	x	x
CR 7.6 (1)	Machine-readable reporting of current security settings	-	x	x
CR 7.7	Least functionality	x	x	x
CR 7.8	Control system component inventory	-	x	x
SAR: Software Application Requirements				
SAR 2.4	Mobile code	x	x	x
SAR 2.4 (1)	Mobile code integrity check	x	x	x
SAR 3.2	Protection from malicious code	x	x	x
NDR: Network device requirements				
NDR 1.6	Wireless Access Management	x	x	x
NDR 1.6 (1)	Unique identification and authentication	-	x	x
NDR 1.13	Access via untrusted networks	x	x	x
NDR 1.13 (1)	Explicit access request approval	-	x	x
NDR 2.4	Mobile code	x	x	x
NDR 2.4 (1)	Mobile code authenticity check	-	x	x
NDR 2.13	Use of physical diagnostic and test interfaces	-	x	x

NDR 2.13 (1)	Active monitoring	-	x	x
NDR 3.2	Protection from malicious code	x	x	x
NDR 3.10	Support for updates	x	x	x
NDR 3.10 (1)	Update authenticity and integrity	-	x	x
NDR 3.11	Physical tamper resistance and detection	-	x	x
NDR 3.11 (1)	Notification of a tampering attempt	-	x	x
NDR 3.12	Provisioning product supplier roots of trust	-	x	x
NDR 3.13	Provisioning asset owner roots of trust	-	x	x
NDR 3.14	Integrity of the boot process	x	x	x
NDR 3.14 (1)	Authenticity of the boot process	-	x	x
NDR 5.2	Zone boundary protection	x	x	x
NDR 5.2 (1)	Deny all, permit by exception	-	x	x
NDR 5.2 (2)	Island mode	-	x	x
NDR 5.2 (3)	Fail close	-	x	x
NDR 5.3	General purpose, person-to-person communication restrictions	x	x	x
Secure Development				
D: Development Documentation				
D_AD.1	Secure initialisation	x	x	x
D_AD.2	Tamper protection	x	x	x
D_AD.3	Security-enforcing mechanisms	x	x	x
D_IS.1	Interface purpose and usage	x	x	x
D_IS.2	Interface parameters	x	x	x
D_IS.3	Error messages	-	x	x
D_DD.1	Subsystem structure	x	x	x
D_DD.2	Module structure	-	x	x
D_DD.3	Subsystem-Module mapping	-	x	x
D_DD.4	Parameters, invocation conventions and return values	-	x	x
D_SC.1	Source code	-	x	x
G: Guidance Documentation				
G_AP.1	Acceptance procedures	x	x	x
G_AP.2	Installation procedures	x	x	x
G_OG.1	Interface usage for each user role	x	x	x
G_OG.2	Possible modes of operation	x	x	x
S: Secure Development				
S_CM.1	Unique component reference	x	x	x
S_CM.2	Consistent usage of component reference	x	x	x
S_CM.3	Configuration management access control measures	-	x	x
S_CM.4	Automated procedures for production	-	x	x
S_CM.5	Component reflecting source code	-	x	x
S_CM.6 (1)	Configuration list content (1)	x	x	x
S_CM.6 (2)	Configuration list content (2)	-	x	x
S_CM.7	Unique identification based on configuration list	x	x	x
S_CM.8	Developer Information	x	x	x

S_DL.1	Secure delivery	x	x	x
S_DS.1	Operational security measures	-	x	x
S_FR.1	Tracking of reported security flaws	x	x	x
S_FR.2	Security flaw description	x	x	x
S_FR.3	Status of corrective measures	x	x	x
S_FR.4	Safeguards	-	x	x
S_FR.5	Contact for user reports and enquires	-	x	x
S_LC.1	Life-cycle model	-	x	x
T: Developer Testing				
T_CA.1	Test coverage analysis	x	x	x
T_CA.2	Test procedures for subsystems	x	x	x
T_CA.3	Test procedures for interfaces	-	x	x
T_TD.1	Test documentation	x	x	x
T_TD.2	Test configuration	x	x	x
T_TD.3	Ordering Dependencies	x	x	x

3 Implementation Concept

3.1 IDS Specification (Component: Connector)

Communication Integrity

COM 01 Protected connection

Connectors communicate with each other only via authenticated, encrypted and integrity protected connections.
The Connector authenticates itself with an IDS certificate at the DAPS hosted by the Identity Provider to get a DAT that is added to every single IDS message. This connection and other connections to external Connectors are TLS encrypted by providing a SSL certificate. Metadata and data are transmitted solely via this connection so that integrity is continuously ensured. An example log of a request from the Connector to another one is shown in Figure 3.1.

```
2020-10-23 14:24:59 INFO  TokenManagerService:103 - ConnectorUUID: 69:D1:31:8E:56:F6:2(
2020-10-23 14:24:59 INFO  TokenManagerService:104 - Retrieving Dynamic Attribute Token.
2020-10-23 14:24:59 INFO  TokenManagerService:124 - Request token: eyJhbGciOiJSUzI1NiJ9
2020-10-23 14:24:59 INFO  ClientProvider:162 - Address: proxy.dortmund.isst.fraunhofer.
2020-10-23 14:24:59 INFO  TokenManagerService:151 - Response body of token request:
{"access_token":"eyJ0eXAiOiJKV1QiLCJraWQiOiJkZWZhdWx0IiwiYWxnIjoiUlMyNTYifQ.eyJzZWN1cm1
2020-10-23 14:24:59 INFO  TokenManagerService:156 - Dynamic Attribute Token: eyJ0eXAiO:
2020-10-23 14:24:59 INFO  IDSHTTPCommunication:140 - URL is valid: https://simpleconne
2020-10-23 14:24:59 INFO  IDSHTTPCommunication:180 - Request is HTTPS: true
```

Figure 3.1: Log of Connector Request

COM 02 Mutual authentication

Connector certificates (see DIN SPEC 6.4.5) facilitate mutual authentication of Connectors every time connection is established.
With each connection the Connector checks if the other Connector has a valid DAT from the DAPS. With the token exchange and the TLS encrypted communication, a mutual authentication is ensured. In addition, the Connector signs every outgoing IDS message with its own valid DAT that is updated every hour.

COM 03 State of the art cryptography

Encryption and integrity protection is facilitated by means of mechanisms considered state of the art by BSI TR 02102-1, NIST SP 800-175b, or an equivalent crypto catalogue.
The Connector encrypts its connection with the help of an X.509v3 certificate, following ISO/IEC 9594-8, provided by the central IDS Identity Provider. The connection is encrypted and authenticated using TLS 1.3, X25519, and AES_128_GCM. As currently there is no central IP, the Connector uses the certificates provided by the Fraunhofer AISEC and connects to a DAPS hosted at https://daps.aisec.fraunhofer.de/. As soon as available, the connection to a central IP is planned.

Data Usage Control

USC 01 Definition of usage policies

Connector allows data providers to define usage policies that will be published together with the data offered.

For each resource, the Connector offers the possibility to add a usage policy. It is defined that at least one policy should be set. The user can choose from seven available patterns that will be extended in the future. Within the self-description, the usage policy is offered as a resource contract the consumer can access. Currently, the Connector is able to provide and process seven policy patterns (see USC 03).

USC 02 Sending of usage policies

Connector offering data sends usage policy to be applied to Connector requesting data every time connection is established.
When exchanging information between provider and consumer, the Connector ensures that the usage policy has been read and accepted before the actual data exchange. This is done by technically (validation key) forcing the data consumer to send a Description Request Message before sending an Artifact Request Message is allowed.

USC 03 Usage policy enforcement

Connector facilitates technical enforcement of data usage policy specified.
The Connector reads, classifies, verifies, and enforces all supported usage policies at runtime. In total, the Connector supports the following policy patterns:

- Allow the Usage of the Data
- Interval-restricted Data Usage
- Duration-restricted Data Usage
- Restricted Number of Usages
- Use data and delete it after
- Local Logging
- Remote Notifications

There are three points in time when the policies are checked. We distinguish between the provider side and the consumer side. When the data provider receives an Artifact Request Message from a consumer Connector, the Artifact Message Handler of the provider Connector checks the pattern of the policy that was added to the requested resource. If the pattern matches one of the following two, an appropriate policy check is performed: provide access or usage during interval. According to the specified rules, the access permission will be set to true or false. If it is true, the Connector returns the data. If not, it will respond with a Rejection Message: not authorized.

After the requested data and its metadata are saved to the consumer Connector's internal database, it can be accessed by using the according endpoint. If a user or application wants to retrieve the data from the internal database, the policies of the requested data resource are checked for policies: usage during interval, duration usage, usage until deletion, usage logging, usage notification, and n time usage. The policy is then implemented using the detected pattern. As described above, the access permission will be set to true or false, and correspondingly, the data is either displayed or not.

On top of that, the Connector performs an automated policy check every minute. If a duty determining the deletion date and time, as in the usage until deletion pattern, is detected, usage control is executed. The Connector removes the data exclusively from its internal database. External third-party systems are responsible for the deletion of the data themselves. If these are e.g. IDS-certified apps, the implementation of the policies will still be guaranteed.

Information Model

INF 01 Self-Description (at Connector)

Connector provides self-description (i.e. metadata) via a defined interface.
The Connector provides a self-description at runtime. It contains all required information about the connector including its latest resource offer. The self-description can be requested from

another Connector by sending an IDS Description Request Message without a requested element. On top of that, the data provider is able to display the self-description by using admin endpoints.

INF 02 Self-Description (at Broker)

To register with a Broker, a Connector must be able to provide the Broker with this self-description.
The Connector is able to send a Connector Update Message with its self-description in the payload to the Broker to register and update itself.

INF 03 Self-Description content

The self-description contains at least the following information: a) cryptographic hash of Connector certificate, b) Connector operator, c) data endpoints offered by Connector, d) log format of data endpoints offered, e) security profile of Connector (i.e. security features supported), f) Connector ID.
The Connector's self-description will contain the following information: a) cryptographic hash of Connector certificate, b) connector operator (maintainer and curator), c) data endpoints offered by Connector, d) log format of data endpoints offered, e) security profile of Connector, f) Connector ID. Furthermore it contains information about: maintainer, curator, outbound model version, inbound model version, title, description, and version.

INF 04 Self-Description evaluation

Connector offering data evaluates self-description of Connector requesting data.
The Connector as data provider is able to receive the self-description of another Connector and will support its evaluation.

INF 05 Dynamic attribute tokens

Dynamic attribute tokens belonging to two communicating Connectors are transmitted every time a connection is established (see DIN Spec 6.4.2) and can therefore be used for access control decisions.
Every time the Connector establishes a connection to another Connector, the DAT is transmitted within the IDS message and can therefore be used for access control decisions.

Identity and Access Management

IAM 01 Connector identifier

Connector is unambiguously identified by means of an identifier derived from a X.509v3 certificate (see also CR 1.2 (1)).
The Connector receives a derived X.509v3 certificate from the Identity Provider in order to identify itself unambiguously.

IAM 02 Time Service

Connector supports central time service (e.g. to verify certificates).
The Connector establishes an indirect connection to a NTP time server each time it communicates with the DAPS by using the hardware time of the system it is running at. Therefore, the hardware time should be synchronized at regular intervals with a valid central time service. A correct time is essential for managing resource timestamps and usage control enforcement.

IAM 03 Online certificate status check

Connector supports online status check of certificates issued (e.g. Online Certificate Status Protocol, OCSP).
The Connector checks the validity of incoming DATs from other Connectors. If a token is no longer valid, the connector notices this and rejects an exchange of information.

IAM 04 Attestation of dynamic attributes

Connector supports the external attestation of dynamic attributes, from which it receives certified attribute information (e.g. through JSON Web Tokens).
The Connector supports the possibility to check dynamic attributes from the DAT. It checks whether the received DAT is certified with the DAPS public key.

Broker Service

BRK 01 Broker service inquiries

Connector supports broker service inquiries by means of browsing self-descriptions of Connectors registered there.
The Connector provides the possibility to send Query Messages with defined SPARQL queries to the Broker to browse registered Connectors and resources.

BRK 02 Broker registration

Connector supports registration with a broker by transmitting self-description.
The Connector is able to send a Connector Update Message with its self-description in the payload to the Broker to register itself.

BRK 03 Broker registration update

Connector supports updates of self-description stored at broker (e.g. when new service is offered) and marking itself as available / unavailable.
The Connector is able to send a Connector Update Message with its self-description in the payload to the Broker to update itself.

Operating System

OS 01 Container support

Connector supports installation and execution of containers.
The Connector can be deployed in a container and will support the installation and execution of other containers to build a coherent cluster.

Apps and App Store Connection

APS 01 App signature

Connector supports only apps possessing a valid signature. This signature is the signed check sum of the software artefact, which was created by means of a private key of the app publisher.
The Connector will support app integration and check for a valid signature.

APS 02 App signature verification

Connector verifies signature after app was downloaded and before it is installed, and before every execution of app. Public key of app publisher is contained in an X.509v3 certificate signed by a Certification Authority accepted by data provider and data consumer.
With app integration, the Connector will be able to verify an app's signature after it was downloaded, before it is installed, and before each execution.

APS 05 App installation

Connector supports apps delivered and installed as independent software containers (i.e. apps bring along possible dependencies of e.g. software modules themselves and can be used irrespective of Connector's configuration).

The Connector will support the delivery and installation of apps as independent software containers.

APS 06 App Store

Connector receives apps from a central app store.
The Connector will be able to receive, integrate, and run apps from a central app store.

Data Usage Transparency

AUD 01 Access control logging

Connector logs each access control decision in the form of an integrity protected log entry in its domain.
The Connector will log each successful or failed access control as integrity protected log entries in the Connector's database, with integration of Log4j.

AUD 02 Data access logging

Connector logs every access to data in the form of an integrity protected entry in its domain.
The Connector will log each successful or failed data access as integrity protected log entries in the Connector's database, with integration of Log4j.

AUD 03 Configuration changes logging

Connector logs any changes made to its configuration in the form of integrity protected entries in its domain.
The Connector will log each configuration change as an integrity protected log entry in the Connector's database, with integration of Log4j.

3.2 62433-4-2

IAC: Identification and authentication control

CR 1.1 Human user identification and authentication

The component shall provide the capability to identify and authenticate all human users according to IEC 62443-3-3 SR 1.1 on all interfaces capable of human user access. This capability shall enforce such identification and authentication on all interfaces that provide human user access to the component to support segregation of duties and least privilege in accordance with applicable security policies and procedures. This capability may be provided locally by the component or by integration into a system level identification and authentication system.
The Connector uses Spring Boot functionality to distinguish between users with administrative rights and those without. Based on this, particular API endpoints, e.g. for managing resources or changing configurations, are protected by authentication and only accessible with credentials (username and password).

CR 1.1 (1) Unique identification and authentication

The component shall provide the capability to uniquely identify and authenticate all human users.
The Connector distinguishes between users with administrative rights and those without. Admin users, on the provider side, have to enter login credentials to access API backend endpoints for managing resources, editing configurations, triggering the communication with an IDS Broker or other Connectors (for requesting or searching resources). On the provider side, an API endpoint (api/ids/data) without any admin rights listens on incoming IDS messages. As described in COM 02, only messages with a valid DAT will be accepted, authentication and unique identification can be ensured for all endpoint interactions.

On the consumer side, the user needs to have admin rights in order to start the request from the consumer Connector to the provider Connector. A unique identification of human users will be provided by integrating identity and access management software as e.g. Keycloak (Keycloak, 2020).

CR 1.2 Software process and device identification and authentication

The component shall provide the capability to identify itself and authenticate with any other component (software application, embedded devices, host devices and network devices), according to IEC 62443-3-3 SR1.2. If the component, as in the case of an application, is running in the context of a human user, in addition, the identification and authentication of the human user according to IEC 62443-3-3 SR1.1 may be part of the component identification and authentication process towards the other components.
The Connector can identify itself to other systems using its X.509 certificate and a unique identifier. It is able to authenticate itself to other components by supporting their authorization processes. Information for this, e.g. credentials, can be stored in the Connector's configuration settings.

CR 1.2 (1) Unique identification and authentication

The component shall provide the capability to uniquely and securely identify and authenticate itself to any other component.
In IDS context, the Connector can identify itself to other systems using its certificate and a unique identifier. If authentication is required for access to other systems, this information can be provided to the Connector by adding e.g. the credentials to the metadata of a resource (currently supported) or via configuration files (future extension). This settings can be added by using the Swagger UI or a future GUI.

CR 1.3 Account management

The component shall provide the capability to support the management of all accounts directly or integrated into a system that manages accounts according to IEC 62443-3-3 SR 1.3.
The Connector provides admin rights management via Spring Security (Alex, et al., 2020) and will support user management by integrating identity and access management software as e.g. Keycloak (Keycloak, 2020).

CR 1.4 Identifier management

The component shall provide the capability to integrate into a system that supports the management of identifiers and/or provide the capability to support the management of identifiers directly according to IEC62443-3-3 SR 1.4.
The Connector has a device identifier as a UUID that is embedded in a X.509 device certificate.

CR 1.5 Authenticator management

Components shall provide the capability to: a) support the use of initial authenticator content; b) support the recognition of changes to default authenticators made at installation time; c) function properly with periodic authenticator change/refresh operation; and d) protect authenticators from unauthorized disclosure and modification when stored, used and transmitted.
The Connector authenticates itself at the DAPS as soon as the application is started. The retrieved DAT is refreshed once per hour. Pre-set authenticators for role-based user management may be changed in the corresponding configuration file and are loaded at installation time. A recommendation to change passwords is explicitly stated in the Connector's description. The user has to identify him- or herself every time a backend endpoint should be used. The safe storage, usage, and transmission is managed by Spring Security (Alex, et al., 2020).

CR 1.7 Strength of password-based authentication

For components that utilize password-based authentication, those components shall provide or integrate into a system that provides the capability to enforce configurable password strength according to internationally recognized and proven password guidelines.
The admin interfaces/endpoints are password protected via Spring Security (Alex, et al., 2020) by integrating identity and access management software as e.g. Keycloak (Keycloak, 2020). This will also ensure the capability to enforce configurable password strength according to internationally recognized and proven password guidelines.

CR 1.8 Public key infrastructure certificates

When public key infrastructure (PKI) is utilized, the component shall provide or integrate into a system that provides the capability to interact and operate in accordance with IEC 62443-3-3 SR1.8.
As the Connector uses the x.509 IDS certificate to identify itself and encrypt the communication with other Connectors, it is part of a PKI. The comparison of private and public keys ensures that tokens are only signed by verified participants.

CR 1.9 Strength of public key-based authentication

For components that utilize public key-based authentication, those components shall provide directly or integrate into a system that provides the capability within the same IACS environment to: a) validate certificates by checking the validity of the signature of a given certificate; b) validate the certificate chain or, in the case of self-signed certificates, by deploying leaf certificates to all hosts that communicate with the subject to which the certificate is issued; c) validate certificates by checking a given certificate's revocation status; d) establish user (human, software process or device) control of the corresponding private key; e) map the authenticated identity to a user (human, software process or device) by checking either subject name, common name or distinguished name against the requested destination; and f) ensure that the algorithms and keys used for the public key authentication conform to 8.5.
The Connector uses the x.509 IDS certificate of the central DAPS maintained by the Fraunhofer AISEC. It will support a future central IDS Identity Provider.

CR 1.10 Authenticator feedback

When a component provides an authentication capability, the component shall provide the capability to obscure feedback of authentication information during the authentication process.
When typing user credentials for login to the admin interface/endpoints, typed passwords are encrypted by characters. Currently, the form looks as shown in Figure 1.2.

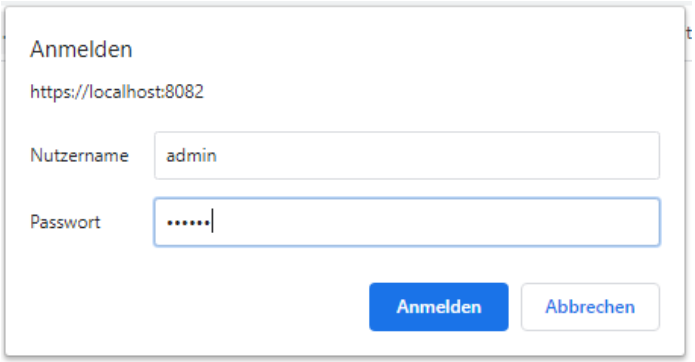

Figure 1.2 Login Form

CR 1.11 Unsuccessful login attempts

When a component provides an authentication capability, the component shall provide the capability to: a) enforce a limit of a configurable number of consecutive invalid access attempts by any user (human, software process or device) during a configurable time period; and b) deny access for a specified period of time or until unlocked by an administrator when this limit has

been reached. An administrator may unlock an account prior to the expiration of the timeout period.
The user retrieves feedback at the Connector's interface if a login was not successful. The form reappears with blank input fields. It is planned to display an error message. The login process will have a limit of three tries in the future. The login will be locked a defined time interval or until an administrator unlocks the account prior to the expiration of the timeout period.

CR 1.12 System use notification

When a component provides local human user access/HMI, it shall provide the capability to display a system use notification message before authenticating. The system use notification message shall be configurable by authorized personnel.
Currently, system use notifications are not relevant. As soon as they become relevant, this requirement will be supported.

CR 1.14 Strength of symmetric key-based authentication

For components that utilize symmetric keys, the component shall provide the capability to: v) establish the mutual trust using the symmetric key; w) store securely the shared secret (the authentication is valid as long as the shared secret remains secret); x) restrict access to the shared secret; and y) ensure that the algorithms and keys used for the symmetric key authentication conform to 8.5.
For user authentication, no symmetric keys are used. For IDS authentication, the Connector owns a keystore that contains the IDS certificate and a truststore that contains all trusted SSL certificates for establishing a trusted connection to backend systems or other Connectors.

UC: Use Control

CR 2.1 Authorization enforcement

The component shall provide an authorization enforcement mechanism for all identified and authenticated human users based on their assigned responsibilities.
The Connector uses Spring Boot functionality to distinguish between users with administrative rights and those without. With the help of user management software, e.g. Keycloak, the Connector will be able to implement authorization enforcement. Based on this, particular endpoints, including those for triggering IDS functionality, are protected by authentication and only accessible with credentials (username and password).

CR 2.2 Wireless use control

If a component supports usage through wireless interfaces it shall provide the capability to integrate into the system that supports usage authorization, monitoring and restrictions according to commonly accepted industry practices.
The Connector does not support usage through wireless interfaces.

CR 2.5 Session lock

If a component provides a human user interface, whether accessed locally or via a network, the component shall provide the capability a) to protect against further access by initiating a session lock after a configurable time period of inactivity or by manual initiation by the user (human, software process or device); and b) for the session lock to remain in effect until the human user who owns the session, or another authorized human user, re-establishes access using appropriate identification and authentication procedures
The Connector will provide session lock. The user will be able to logout and login to his/her user account. In addition, the session will be closed automatically in case of inactivity. After session lock, the user has to re-login with credentials.

CR 2.8 Auditable events

The component shall provide the capability to generate audit records relevant to security for the following categories: a) access control; b) request errors; c) control system events; d) backup and restore event; e) configuration changes; and f) audit log events. Individual audit records shall include: g) timestamp; h) source (originating device, software process or human user account); i) category; j) type; k) event ID; and l) event result.
The Connector will support audit records for all listed categories.

CR 2.9 Audit storage capacity

The component shall a) provide the capability to allocate audit record storage capacity according to commonly recognized recommendations for log management; and b) provide mechanisms to protect against a failure of the component when it reaches or exceeds the audit storage capacity.
The Connector will support audit storage management. A space reserved for logs will be separated from any other used storage so a reached or exceeded storage capacity will not result in the component crashing. To prevent storage problems, old logs will be deleted periodically.

CR 2.10 Response to audit processing failures

The component shall a) provide the capability to protect against the loss of essential services and functions in the event of an audit processing failure; and b) provide the capability to support appropriate actions in response to an audit processing failure according to commonly accepted industry practices and recommendations.
Audit failures will not have any impact on the Connector's internal database or integrated systems. When the audit storage capacity is reached or exceeded, a warning will be displayed.

CR 2.11 Timestamps

The component shall provide the capability to create timestamps (including date and time) for use in audit records.
With every log, the Connector will create a corresponding timestamp including date and time.

CR 2.12 Non-repudiation

If a component provides a human user interface, the component shall provide the capability to determine whether a given human user took a particular action. Control elements that are not able to support such capability shall be listed in component documents.
The Connector will be able to assign actions to a specific user, as all API endpoints for Connector management are protected by authorization enforcement mechanism (see CR 2.1).

SI: System integrity

CR 3.1 Communication integrity

The component shall provide the capability to protect integrity of transmitted information.
As all information are solely transmitted via TLS encrypted connections, the Connector is able to protect the information's integrity.

CR 3.1 (1) Communication authentication

The component shall provide the capability to verify the authenticity of received information during communication.
Communication authentication is achieved by using communication confidentiality (TLS encryption).

CR 3.3 Security functionality verification

Components shall provide the capability to support verification of the intended operation of security functions according to IEC 62443-3-3 SR3.3.
The Connector does not support verification of the intended operation of security functions.

CR 3.4 Software and information integrity

Components shall provide the capability to perform or support integrity checks on software, configuration and other information as well as the recording and reporting of the results of these checks or be integrated into a system that can perform or support integrity checks.
The Connector will support integrity checks on software and configurations, and provide recording and reporting of their results.

CR 3.5 Input validation

The component shall validate the syntax, length and content of any input data that is used as an industrial process control input or input via external interfaces that directly impacts the action of the component.
The Connector will not support input validation.

CR 3.6 Deterministic output

Components that physically or logically connect to an automation process shall provide the capability to set outputs to a predetermined state if normal operation as defined by the component supplier cannot be maintained.

The Connector will not support deterministic output.CR 3.7 Error handling

Components shall identify and handle error conditions in a manner that does not provide information that could be exploited by adversaries to attack the IACS.
The Connector handles errors in a manner that does not provide information that could be exploited by adversaries. Error logs are only accessible to authorized users and are never disclosed to the public. Caused exceptions during IDS communication are handled IDS compliant with Error Messages and appropriate Rejection Reason.

CR 3.8 Session integrity

The component shall provide mechanisms to protect the integrity of communications sessions including: a) the capability to invalidate session identifiers upon user logout or other session termination (including browser sessions); b) the capability to generate a unique session identifier for each session and recognize only session identifiers that are system-generated; and c) the capability to generate unique session identifiers with commonly accepted sources of randomness.
The Connector will generate, invalidate, and manage unique session identifiers with commonly accepted sources of randomness. On the one hand, every incoming IDS component communication is TLS encrypted, as well as operations via user interfaces. This will prevent attacks on the integrity of communication sessions. On the other hand, Spring Security provides (concurrent) session control and integrity: it handles session timeout, prevents exposing session information for forbidden session tracking, and provides secure session cookies and session scoped beans (Paraschiv, 2020).

DC: Data confidentiality

CR 4.1 Information confidentiality

The component shall a) provide the capability to protect the confidentiality of information at rest for which explicit read authorization is supported; and b) support the protection of the confidentiality of information in transit as defined in IEC 62443-3-3 SR 4.1.

a) Only user with admin rights will have the permission to edit and read the information at rest, to protect the information's confidentiality.
b) The confidentiality of information in transit is protected by the TLS cryptographic protocol for secure communication.

CR 4.2 (1) Erase of shared memory resources

The component shall provide the capability to protect against unauthorized and unintended information transfer via volatile shared memory resources.

This requirement is implemented by the memory management subsystem of the utilized Linux or Windows kernel that is responsible for properly managing shared memory resources in the system.

CR 4.3 Use of cryptography

If cryptography is required, the component shall use cryptographic security mechanisms according to internationally recognized and proven security practices and recommendations.
Information are shared via TLS (support for 1.2 and 1.3) encrypted communication. Passwords are encrypted by Spring Security using bcrypt (Spring, 2020).

RDF: Restricted data flow

CR 5.1 Network segmentation

Components shall support a segmented network to support zones and conduits, as needed, to support the broader network architecture based on logical segmentation and criticality.
The Connector will support a segmented network. Every running container will be associated to a different network zone by providing its own virtual network stack. The Connector as the core container will have root rights and be able to manage network and firewall configurations for all separated containers and their networks. As root namespace, it provides an external IP and can be reached from an external network. An overview is given in Figure 1.3

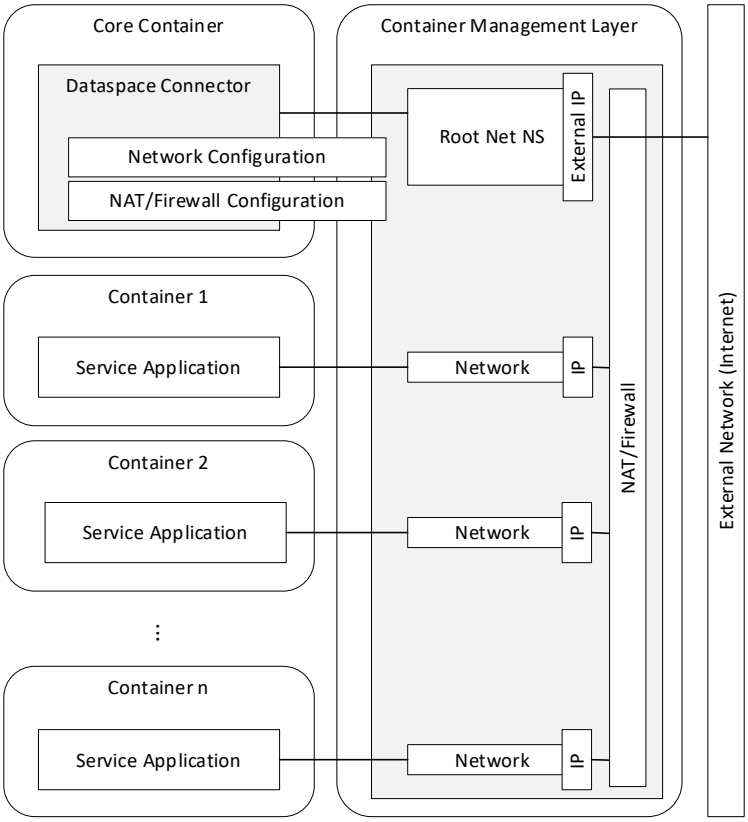

Figure 1.3 Network Segmentation

TRE: Timely response to events

CR 6.1 Audit log accessibility

The component shall provide the capability for authorized humans and/or tools to access audit logs on a read-only basis.
With authorization enforcement mechanisms (see CR 2.1), the Connector will provide the capability for authorized humans and tools to access audit logs on a read-only basis.

RA: Resource availability

CR 7.1 Denial of service protection

Components shall provide the capability to maintain essential functions when operating in a degraded mode during a DoS event.
The Connector does not support a degraded mode.

CR 7.2 Resource management

The component shall provide the capability to limit the use of resources by security functions to protect against resource exhaustion.
The Connector does not provide the capability to limit the use of resources. However, the Linux kernel provides the capability to limit memory usage, CPU time, network bandwidth, or access to hardware devices.

CR 7.3 Control system backup

The component shall provide the capability to participate in system level backup operations in order to safeguard the component state (user- and system-level information). The backup process shall not affect the normal component operations.
The Connector will provide the capability to participate in system level backup operations in order to safeguard the component's state information.

CR 7.4 Control system recovery and reconstitution

The component shall provide the capability to recovered and reconstitute to a known secure state after a disruption or failure.
The Connector will be able to recover to a known secure state after a disruption or failure by recovering service containers from backup snapshots.

CR 7.6 Network and security configuration settings

The component shall provide the capability to be configured according to recommended network and security configurations as described in guidelines provided by the control system supplier. The component shall provide an interface to the currently deployed network and security configuration settings.
The Connector will provide an interface for network and security configuration settings.

CR 7.7 Least functionality

The component shall provide the capability to specifically restrict the use of unnecessary functions, ports, protocols and/or services.
The Connector will provide the capability to specifically restrict the use of unnecessary functions, ports, protocols, and services.

NDR: Network device requirements

NDR 1.6 Wireless Access Management

A network device supporting wireless access management shall provide the capability to identify and authenticate all users (humans, software processes or devices) engaged in wireless communication.
The Connector does not support wireless access management.

NDR 1.13 Access via untrusted networks

The network device supporting device access into a network shall provide the capability to monitor and control all methods of access to the network device via untrusted networks.
The Connector will support monitoring and controlling accesses via untrusted networks. As shown in Figure 1.3, the core container will be able to configure network and firewall settings. Therefore, it has an overview of all incoming communication, can provide it via an interface, and enable an authorized user to agree to a connection establishment or e.g. block certain IPs.

NDR 2.4 Mobile code

In the event that a network device utilizes mobile code technologies, the network device shall provide the capability to enforce a security policy for the usage of mobile code technologies. The security policy shall allow, at a minimum, the following actions for each mobile code technology used on the network device: a) Control execution of mobile code; b) control which users (human, software process, or device) are allowed to transfer mobile code to/from the network device; and c) control the code execution based upon integrity checks on mobile code and prior to the code being executed.
With app integration, the Connector will be able to a) control their execution; b) control which users are allowed to transfer apps from or to the App Store; and c) control the code execution based upon integrity checks.

NDR 3.2 Protection from malicious code

The network device shall provide for protection from malicious code.
The Connector will only allow authenticated and integrity protected software execution. All service software will be isolated from each other and from the management layer. Protection from malicious code is provided by integrity and authenticity checks of system components and service containers prior to their execution.

NDR 3.10 Support for updates

Network devices shall support the ability to be updated and upgraded.
The Connector will be capable of downloading software updates/upgrades and installing it. The update process of the service applications will be equivalent to the application installation process.

NDR 3.14 Integrity of the boot process

Network devices shall verify the integrity of the firmware, software and configuration data needed for the component's boot process prior to it being used in the boot process.
This will not be supported.

NDR 5.2 Zone boundary protection

A network device at a zone boundary shall provide the capability to monitor and control communications at zone boundaries to enforce the compartmentalization defined in the risk-based zones and conduits model.
Network interfaces corresponding to different security zones can be assigned to different service containers. The Connector will be able to monitor and control the communication between different zones.

NDR 5.3 General purpose, person-to-person communication restrictions

A network device at a zone boundary shall provide the capability to protect against general purpose, person-to-person messages from being received from users or systems external to the control system.
This will not be supported.

3.3 Secure Development

D: Development Documentation

D_AD.1 Secure initialization

The development documentation shall include an architectural description stating how the component preserves security during initialization, i.e. how an initial secure state is reached.
The development documentation includes all necessary information of security functionalities including admin rights management and IDS certification aspects. An architectural description will be added.

D_AD.2 Tamper protection

The development documentation shall include an architectural description stating how the component is able to protect itself from tampering by untrusted active entities.
The development description will outline the ability of the Connector to protect itself from manipulation from external entities. This will cover e.g. how user inputs are handled by the Connector.

D_AD.3 Security-enforcing mechanisms

The development documentation shall include an architectural description containing an analysis that adequately describes how the security-enforcing mechanisms of the component cannot be bypassed.
The documentation will include an architectural description containing an analysis that adequately describes how the security-enforcing mechanisms of the component cannot be bypassed.

D_IS.1 Interface purpose and usage

The development documentation shall include an interface specification stating the purpose of and method of use for each interface of the component.
The development documentation includes an explanation regarding every interface (API endpoint) and functionality of the Connector.

D_IS.2 Interface parameters

The development documentation shall include an interface specification completely and accurately describing all parameters associated with every interface.
The documentation provides a detailed description of all interface parameters.

D_DD.1 Subsystem structure

The development documentation shall include a design description stating the structure of the entire component in terms of subsystems.
The description will identify all subsystems of the Connector.

G: Guidance Documentation

G_AP.1 Acceptance procedures

The developer shall provide a guidance documentation describing the acceptance procedures, i.e. the steps necessary for secure acceptance of the component by the end user.
The acceptance procedures will adequately reflect the steps a user has to perform in order to accept the delivered Connector. As a minimum, this will include that the user has to check that all parts of the component as indicated in the IDS certificate have been delivered in the correct version. On top of that, detailed information about how the user can ensure that the delivered Connector is the complete evaluated instance and detect its modification/masquerading.

G_AP.2 Installation procedures

The developer shall provide a guidance documentation describing the installation procedures, i.e. the steps necessary for secure installation of the component and the secure preparation of the operational environment.
The installation procedures provide detailed information about the following: a) minimum system requirements for secure installation; b) requirements for the operational environment; c) the steps the user has to perform in order to get to an operational component being commensurate with its evaluated configuration. For each step, this includes a clear scheme for the decision on the next step depended on success, failure, or problems at the current step; d) changing the installation specific security characteristics, for example parameters, settings or passwords; e) handling exceptions and problems.

G_OG.1 Interface usage for each user role

The developer shall provide an operational user guidance describing for each user role, the secure use of the available interfaces provided by the component.
The operational user guidance will provide full advice regarding effective use of the security functionality (e.g. reviewing password composition practices, suggested frequency of backups, discussion on the effects of changing user access privileges).

G_OG.2 Possible modes of operation

The operational user guidance shall identify all possible modes of operation of the component (including, if applicable, operation following failure or operational error), their consequences and implications for maintaining secure operation.
The operational user guidance will identify all possible modes of operation of the Connector, their consequences and implications for maintaining secure operation.

S: Secure Development

S_CM.1 Unique component reference

The component shall be labelled with a unique reference.
The name and unique version of the Connector can be found on the release page, in the changelog file, and by the operational Connector itself (in the code). Based on this, the version number will be displayed in the IDS self-description and any Connector user interface, so it can be accessed at runtime.

S_CM.2 Consistent usage of component reference

The component shall be labelled with a unique reference.
All labels on the Connector will be used consistently to ensure that consumers can be confident that they have purchased the evaluated version, that they have installed this version, and that they have the correct version of the guidance to operate the Connector. The Connector uses the SemVer for versioning. The release versions are tagged with their respective version.

S_CM.6 (1) Configuration list content (1)

The configuration list shall include the following set of items: a) the component itself; b) the evaluation evidence required for the evaluation;
The configuration list will contain all relevant items.

S_CM.7 Unique identification based on configuration list

The configuration list shall uniquely identify each configuration item.
The configuration list will contain sufficient information to uniquely identify which version of each item has been used, typically by using version numbers.

S_CM.8 Developer Information

The configuration list shall indicate the developer of each security functionality relevant configuration item.

All developing companies involved in the development of the Connector will be mapped to a respective configuration item.

S_DL.1 Secure delivery

The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the component or parts of it to the consumer.
The hosting of the delivered Connector as software package is organized via GitHub. There, a company that wants to use the Connector can download it by cloning or forking the repository. A ready to use image will be provided at Docker Hub (Docker, 2020).The documentation will describe all proper procedures to maintain security during transfer of the Connector as well as the identification procedure by the user. If employed, cryptographic checksums or a software signature will be described.

S_FR.1 Tracking of reported security flaws

The flaw remediation procedures shall describe the procedures used to track all reported security flaws in each release of the component.
The flaw remediation procedures will describe the procedures used to track all reported security flaws in each release of the component.

S_FR.2 Security flaw description

The application of the flaw remediation procedures shall produce a description of each security flaw in terms of its nature and effects.
The application of the flaw remediation procedures will produce a description of each security flaw in terms of its nature and effects.

S_FR.3 Status of corrective measures

The application of flaw remediation procedures shall identify the status of finding a correction to each security flaw.
The flaw remediation procedures will identify the different stages of security flaws. This differentiation will include at least: suspected security flaws that have been reported, suspected security flaws that have been confirmed to be security flaws, and security flaws whose solutions have been implemented.

T: Developer Testing

T_CA.1 Test coverage analysis

The test coverage analysis shall show a complete correspondence between the component's interfaces and the tests in the test documentation.
The Connector will provide tests for all IDS functionalities. Currently implemented:

- SelfDescriptionTest: This class tests whether the connector returns a valid self-description. A mock call is made to its self-description endpoint to check whether the response can be parsed to an Information Model connector object. This is done once without resources and once with resources to see whether available resources are added to the self-description correctly.
- RequestDescriptionTest: This class tests whether the connector can request descriptions (self or artifact descriptions) from another connector. A mock call is made to its endpoint for initiating a description request. The response that would be returned by another connector is mocked. The test case requesting an artifact description verifies that a respective entry has been added to the database.
- RequestArtifactTest: This class tests whether the connector can request artifacts from other connectors. A mock call is made to its endpoint for initiating an artifact request. The response that would be returned by another connector is mocked. The test case then verifies that the data was added to the respective resource correctly.

- DescriptionRequestMessageHandlingTest: This class tests whether the connector can handle incoming DescriptionRequestMessages correctly. A mock call is made to its endpoint for IDS messages, sending a DescriptionRequestMessage. This is done for requesting the connector's self-description as well as an artifact description (once with an invalid requestedArtifact). The response returned by the connector is checked for correct message type (DescriptionResponseMessage, RejectionMessage) and payload.
- ArtifactRequestMessageHandlingTest: This class tests whether the connector can handle incoming ArtifactRequestMessages correctly. A mock call is made to its endpoint for IDS messages, sending an ArtifactRequestMessage, once with a valid and once with an invalid requestedArtifact. The response returned by the connector is checked for correct message type (ArtifactResponseMessage, RejectionMessage) and payload. The PolicyHandler is mocked in these test cases.

A full coverage of each of its interfaces will be added with a future update.

T_CA.2 Test procedures for subsystems

The test procedures shall contain descriptions of the security-related subsystem behavior and interaction that are tested.
The Connector provides descriptions in the test of the security-related subsystem behavior and interaction.

T_TD.1 Test documentation

The test documentation shall include scenarios for performing each test, expected test results and actual test results.
A test documentation will be provided with a future update. Currently, the test are performed with each application packaging with Maven and results are shared in the console log.

T_TD.2 Test configuration

The test configuration shall be consistent with the configuration list (S_CM.6).
The documentation will provide all relevant information about the test configuration being used (for the configuration of the Connector and any test equipment being used). This information will be detailed enough to ensure that the test configuration is reproducible.

T_TD.3 Ordering Dependencies

The test documentation shall provide sufficient instructions for any ordering dependencies of the tests.
Currently, the tests do not include any ordering dependencies. The right workflow is provided within the test classes. For example, demo DATs are added for sending IDS messages.

4 Acronyms

AISEC	Angewandte und Integrierte Sicherheit
API	Application Programming Interface
CPU	Central Processing Unit
DAPS	Dynamic Attribute Provisioning Service
DAT	Dynamic Attribute Token
IDS	Internation Data Spaces
IP	Identity Provider
NTP	Network Time Protocol
PKI	Public Key Infrastructure
REST	Representational State Transfer
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UUID	Universally Unique Identifier

5 References

Alex, B., Taylor, L., Winch, R., Hillert, G., Grandja, J., Bryant, J., . . . Syer, D. (2020, 10 24). *Spring Security Reference*. Retrieved from https://docs.spring.io/spring-security/site/docs/current/reference/html5/

Bader, S. (05. 08 2020). *IDS Usage Policy Language.* Von https://industrialdataspace.jiveon.com/servlet/JiveServlet/download/2264-15-2948/Specification+-+IDS+Usage+Policy+Language_05-08-2020.docx abgerufen

Docker, I. (30. 11 2020). *Docker Hub*. Von https://hub.docker.com/ abgerufen

Keycloak. (10. 11 2020). Von https://www.keycloak.org/ abgerufen

Paraschiv, E. (30. 11 2020). *Control the Session with Spring Security*. Von https://www.baeldung.com/spring-security-session abgerufen

Spring. (15. 11 2020). *Class BCrypt*. Von https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/crypto/bcrypt/BCrypt.html abgerufen