1. INTRODUCTION

1.1. INFORMATION FROM THE TECHNICAL EVALUATION REPORT

Reference component	IDSA Dataspace connector
Version	Version 5.1.2
Author or authors	Javier Martínez Temiño
Approved by	
Date	15/07/2021
File code	v0.3
Type of evaluation	Evaluation Technical Report
Taxonomy of the product	

1.2. DEVELOPER AND TOE INFORMATION

Developer Data (Name and Address)	Fraunhofer ISST
Developer Contact Details (Name and e-mail)	
Name of the TOE	
Version of the TOE	

2. DECRIPTION OF THE TOE

The Dataspace connector is a software meant to work inside a dataspace as a connector. This component act as a data provider and a data consumer and it is the central technical component of the International Data Space.

2.1. FUNCTIONAL DESCRIPTION OF THE TOE

The Dataspace Connector is a software program meant to communicate inside an International Data Space. The same connector can operate as a data provider or a data consumer. All these communications are done complying the trust profile specified of the connector, with the security methods associated.

The goal of this implementation is to demonstrate the usage of the IDS Information Model for core communication tasks showing an actual application of the concepts introduced in the Handshake Document.

The readme file contains the IDS Dataspace Connector link to documentation explaining the purpose of this component together with the repository structure and the steps to follow to deploy and create an instance of the connector.

As this connector is a reference implementation of the architecture it is expected for it to fulfill all the trust profiles of the IDS certification, that way, every IDS component can be tested using this as the reference connector regardless of its trust profile.

2.2. INVENTORY OF SECURITY FUNCTIONS IDENTIFIED IN THE SECURITY STATEMENT

The security functions of this component, as well as the rest of components of the IDS architecture, are collected in the criteria of each trust profile and type of component. That way, if applying to the base profile, the security requirements of the component will be less strong to the ones required for a higher security trust profile. As the criteria of the different trust profiles have not been completely defined yet, as well as the test cases required for them, this evaluation focuses mainly on the useability of the component in an IDS reference architecture.

The security features listed in the security statement must be described and classified by functionality and must be assigned a unique identifier that must be used throughout the report to reference that functionality.

Encryption in communication would be tested using WireShark.

Integrity checks, encryption, PKI… etc

Security functions based on the IDS criteria for its trust profile.

3. EXECUTION ENVIRONMENT

3.1. DESCRIPTION OF THE EXECUTION ENVIRONMENT

The Dataspace Connector makes use of the technologies described below:

Java, Maven, Spring Boot, Spring Data JPA, Spring Security, OpenAPI, HATEOAS, Swagger, LOG4J2, Docker, Kubernetes, JSON(-LD), Jaeger TLS.

It can be run either via java, through a docker container or with kubernetes.

3.2. HIPÓTESIS SOBRE EL ENTORNO DE EJECUCIÓN

4. EXECUTIVE SUMMARY OF THE EVALUATION

This evaluation is done focusing on the implementation of the component as part of the reference testbed implementation. This series of test are focused on ensuring the correct implementation of the component against the reference IDS architecture to ensure it is an operational and certificated IDS connector.

The evaluation process of the Dataspace Connector is based on three main aspects of the software.

4.1. DOCUMENTATION EVALUATION.

The first aspect studies its documentation. This includes ensuring that the installation procedures, the deploying mechanisms, the technologies used, and the functionalities of the component are correctly explained, and you can do all this actions by following this documentation.

The documentation is accessible via the GitHub repository of the component. This documentation contains the necessary procedures for installing, running, and communicating with the component.

4.2. CODE SAFETY EVALUATION.

This aspect focuses on the security of the component focusing on the code and technologies used to develop the component. To test this, two procedures have been followed.

- First, the code is scanned with the sonarcube framework. This framework analyzes the code searching for vulnerabilities, bugs, and possible security risks. Once this scan is done, the results are reported to the development team for them to assess whether it is necessary to act on the possible vulnerabilities.
- It is also studied what vulnerabilities have the technologies being used by the component, and if they are solved in the implementation. If further questions are required, the development team is contacted to ensure all these vulnerabilities had been mitigated.

4.3. API EVALUATION.

This aspect studies the component API's functionalities. As this is the main way to configure and communicate with the connector it is mandatory for it to work exactly as explained in the documentation.

The API documentation is accessible via the component, once it has been launched, in the path:

https://localhost:8080/api/docs

In this documentation is explained every function the API can handle and its expected response.

To test all these functions, the karate framework was used. This framework, based on java and javascript, is focused on API testing. The reports of the validation done are also uploaded to the testbed repository in the path:

https://github.com/International-Data-Spaces-Association/IDS-testbed/tree/master/Testbed/Validation/DSC/Karate%20API%20tests/target/karate-reports

4.4. INTEROPERABILITY EVALUATION.

As a last test, it is tested that the component can interoperate with other components of the IDS architecture as expected. To test this interoperability, it has been first the interoperation between the connector with the DAPS and once they are working together, the connector gets tested with the broker.

DAPS operation

To test the operation between the connector and the DAPS it has been tested with the levels of scenarios where only one combination of them will be a correct one. These possible scenarios are:

1. TEST_DEPLOYMENT
2. PRODUCTIVE_DEPLOYMENT

3. LOCALHOST CERT
4. EXPIRED CERT
5. VALID CERT
6. UNRECOGNIZED CERT by DAPS

7. VALID DAPS ENDPOINT
8. INVALID DAPS ENDPOINT

The combination of these scenarios results in the following possible scenarios:

- 137 TEST_DEPLOYMENT + LOCALHOST CERT + VALID DAPS ENDPOINT
- 138 TEST_DEPLOYMENT + LOCALHOST CERT + INVALID DAPS ENDPOINT
- 147 TEST_DEPLOYMENT + EXPIRED CERT + VALID DAPS ENDPOINT
- 148 TEST_DEPLOYMENT + EXPIRED CERT + INVALID DAPS ENDPOINT
- 157 TEST_DEPLOYMENT + VALID CERT + VALID DAPS ENDPOINT
- 158 TEST_DEPLOYMENT + VALID CERT + INVALID DAPS ENDPOINT
- 167 TEST_DEPLOYMENT + UNRECOGNIZED CERT by DAPS + VALID DAPS ENDPOINT
- 168 TEST_DEPLOYMENT + UNRECOGNIZED CERT by DAPS + INVALID DAPS ENDPOINT
- 237 PRODUCTIVE_DEPLOYMENT + LOCALHOST CERT + VALID DAPS ENDPOINT
- 238 PRODUCTIVE_DEPLOYMENT + LOCALHOST CERT + INVALID DAPS ENDPOINT
- 247 PRODUCTIVE_DEPLOYMENT + EXPIRED CERT + VALID DAPS ENDPOINT
- 248 PRODUCTIVE_DEPLOYMENT + EXPIRED CERT + INVALID DAPS ENDPOINT

- 257 PRODUCTIVE_DEPLOYMENT + VALID CERT + VALID DAPS ENDPOINT (CORRECT COMBINATION)
- 258 PRODUCTIVE_DEPLOYMENT + VALID CERT + INVALID DAPS ENDPOINT
- 267 PRODUCTIVE_DEPLOYMENT + UNRECOGNIZED CERT by DAPS + VALID DAPS ENDPOINT
- 268 PRODUCTIVE_DEPLOYMENT + UNRECOGNIZED CERT by DAPS + INVALID DAPS ENDPOINT

5. EVALUATION VERDICT

In this section the evaluator assigns in the Evaluation Technical Report a final verdict for the evaluation. The possible verdict results of every test case executed are the following:

a) **OK**: The security functionality of the TOE complies with the Security Statement and the test case resolution is satisfactory and matches the expected result of the test performed.

b) **FAIL**: The security functionality of the TOE does not comply with the provisions of the Security Declaration and / or the test case resolution is not satisfactory, not matching the expected result of the test performed.

The evaluation verdict is going to be divided in the four main evaluation areas that had been tested from the component.

- **Documentation evaluation** – OK. The documentation provided for the component fulfill all the requirements of operability for the connector.
- **Code safety evaluation**
  - Ok. Sonarcube report did not find any major vulnerability and all the little bugs and faults have already been communicated to the developers and are working on reduce them as much as possible.
  - OK. The technologies used do not have any open vulnerabilities or, if they had, the code implemented does not use the vulnerable functionalities.
- **API evaluation** – OK. All the functions specified in the API's documentation had been tested and are working as expected.
- **Interoperability evaluation** – OK. Interoperability has been tested and the connector is capable of logging to an IDS broker and communicate with other instance of the connector.

6. COMPONENT INSTALLATION

The evaluator is in charge of verifying that the documentation is correct in terms of the installation and implementation of the component. Including the configuration required of the TOE so that for both Linux and Windows system it is specified in a complete way the steps to be performed by the user.

This document aims to aid IT administrators or developers in the installation of the IDS Dataspace Connector.

Note: this guide works with provided images and is not targeted for development purposes. Thus, instructions for building and editing the docker image file will not be included here.

At first, clone the repository:

```
git clone https://github.com/International-Data-Spaces-Association/DataspaceConnector.git
```

The resource folder `resources/conf` provides three important files that are loaded at application start:

- `keystore-localhost.p12` : The provided keystore, on the one hand, is used as IDS certificate that is loaded by the IDS Messaging Services for requesting a valid Dynamic Attribute Token (DAT) from the Dynamic Attribute Provisioning Service (DAPS). Each message to an IDS participant needs to be signed with a valid DAT. On the other hand, it can be used as SSL certificate for TLS encryption.
- `truststore.p12` : The truststore is used by the IDS Messaging Services for any HTTP/S communication. It ensures the connection to trusted addresses.
- `config.json` : The configuration is used to set important properties for IDS message handling.

6.1. Step 1: Connector Properties

When starting the application, the `config.json` will be scanned for important connector information, e.g. its ID, address, contact information, or proxy settings. Please keep this file up to date to your custom settings. In case you want to use the demo cert, you don't need to change anything except the proxy settings.

For outgoing requests, the connector needs information about an existing system proxy that needs to be set in the `resources/conf/config.json` .

```
"ids:connectorProxy" : [ {
  "@type" : "ids:Proxy",
  "@id" : "https://w3id.org/idsa/autogen/proxy/548dc73a-ccfb-4039-9569-4b8e219b90bc",
  "ids:proxyAuthentication" : {
    "@type" : "ids:BasicAuthentication",
    "@id" : "https://w3id.org/idsa/autogen/basicAuthentication/47e3cd59-d351-4f5b-99fc-561c94bad5e1"
  },
  "ids:proxyURI" : {
    "@id" : "http://host:port"
  },
  "ids:noProxy" : [ {
```

```
    "@id" : "https://localhost:8080/"
  }, {
    "@id" : "http://localhost:8080/"
  } ]
} ]
```

Check if your system is running behind a proxy. If this is the case, specify the `ids:proxyURI` and change `ids:noProxy` if necessary. Otherwise, delete the key `ids:connectorProxy` and its values.

A full configuration example may look like this:

```
{
 "@context" : {
   "ids" : "https://w3id.org/idsa/core/",
   "idsc" : "https://w3id.org/idsa/code/"
 },
 "@type" : "ids:ConfigurationModel",
 "@id" : "https://w3id.org/idsa/autogen/configurationModel/7672b568-7878-4f62-8032-5c73de969414",
 "ids:configurationModelLogLevel" : {
   "@id" : "idsc:MINIMAL_LOGGING"
 },
 "ids:connectorDeployMode" : {
   "@id" : "idsc:TEST_DEPLOYMENT"
 },
 "ids:connectorDescription" : {
   "@type" : "ids:BaseConnector",
   "@id" : "https://w3id.org/idsa/autogen/baseConnector/7b934432-a85e-41c5-9f65-669219dde4ea",
   "ids:publicKey" : {
     "@type" : "ids:PublicKey",
     "@id" : "https://w3id.org/idsa/autogen/publicKey/78eb73a3-3a2a-4626-a0ff-631ab50a00f9",
     "ids:keyType" : {
       "@id" : "idsc:RSA"
     },
     "ids:keyValue" : "[...]"
   },
   "ids:description" : [ {
     "@value" : "IDS Connector with static example resources hosted by the Fraunhofer ISST",
     "@type" : "http://www.w3.org/2001/XMLSchema#string"
   } ],
   "ids:version" : "1.0",
   "ids:hasDefaultEndpoint" : {
     "@type" : "ids:ConnectorEndpoint",
     "@id" : "https://w3id.org/idsa/autogen/connectorEndpoint/e5e2ab04-633a-44b9-87d9-a097ae6da3cf",
     "ids:accessURL" : {
       "@id" : "https://localhost:8080/api/ids/data"
     }
   },
   "ids:outboundModelVersion" : "4.0.4",
   "ids:inboundModelVersion" : [ "4.0.0", "4.0.4" ],
   "ids:title" : [ {
     "@value" : "Dataspace Connector",
     "@type" : "http://www.w3.org/2001/XMLSchema#string"
   } ],
   "ids:securityProfile" : {
     "@id" : "idsc:BASE_SECURITY_PROFILE"
```

```
  },
  "ids:curator" : {
    "@id" : "https://www.isst.fraunhofer.de/"
  },
  "ids:maintainer" : {
    "@id" : "https://www.isst.fraunhofer.de/"
  }
},
"ids:trustStore" : {
  "@id" : "file:///conf/truststore.p12"
},
"ids:connectorStatus" : {
  "@id" : "idsc:CONNECTOR_ONLINE"
},
"ids:keyStore" : {
  "@id" : "file:///conf/keystore.p12"
},
"ids:connectorProxy" : [ {
  "@type" : "ids:Proxy",
  "@id" : "https://w3id.org/idsa/autogen/proxy/548dc73a-ccfb-4039-9569-4b8e219b90bc",
  "ids:proxyAuthentication" : {
    "@type" : "ids:BasicAuthentication",
    "@id" : "https://w3id.org/idsa/autogen/basicAuthentication/47e3cd59-d351-4f5b-99fc-561c94bad5e1"
  },
  "ids:proxyURI" : {
    "@id" : "http://proxy.dortmund.isst.fraunhofer.de:3128"
  },
  "ids:noProxy" : [ {
    "@id" : "https://localhost:8080/"
  }, {
    "@id" : "http://localhost:8080/"
  }, {
    "@id" : "https://localhost:8081/"
  }, {
    "@id" : "http://localhost:8081/"
  } ]
} ]
}
```

Note: If you are not familiar with the IDS Information Model, the API provides an endpoint `GET /api/examples/configuration` to print a filled in Java object as JSON-LD. Adapt this to your needs, take the received string and place it in the `config.json`.

If you want to connect to a running connector or any other system running at `https://`, keep in mind that you need to add the keystore to your truststore. Otherwise, the communication will fail. With the provided truststore, the Dataspace Connector accepts its own localhost certificate, public certificates, and any IDS keystore that was provided by the Fraunhofer AISEC.

6.2. Step 2: IDS Certificate

In the provided `config.json`, the `ids:connectorDeployMode` is set to `idsc:TEST_DEPLOYMENT`. This allows to use the `keystore-localhost.p12` as an IDS certificate. For testing purpose, the existing cert can be used, as on application start, the IDS Messaging Services will not get a valid DAT from the DAPS and for received messages, the sent DAT will not be checked.

To turn on the DAT checking, you need to set the `ids:connectorDeployMode` to `idsc:PRODUCTIVE_DEPLOYMENT`. For getting a trusted certificate, contact Gerd Brost. Add the keystore with the IDS certificate inside to the `resources/conf` and change the filename at `ids:keyStore` accordingly. In addition, set your connector id to uniquely identify your connector towards e.g. the IDS Metadata Broker:

```
"ids:connectorDescription" : {
   "@type" : "ids:BaseConnector",
   "@id" : "CONNECTOR_URL",
```

Note: The `TEST_DEPLOYMENT` mode and accepting a demo cert is for testing purposes only! This mode is a **security risk** and cannot ensure that the connector is talking to a verified IDS participant. Furthermore, messages from the Dataspace Connector without a valid IDS certificate may not be accepted by other Connector implementations and will not be accepted by the IDS Metadata Broker running in the IDS lab.

6.3. Step 3: General Settings (optional)

The `application.properties` specifies several Spring Boot and IDS configurations.

Tomcat

To define on which port the connector should be running, change `server.port={PORT}`.

OpenApi

You can change Swagger properties by changing the following settings:

```
springdoc.swagger-ui.path=/api/docs
springdoc.swagger-ui.operationsSorter=alpha
springdoc.swagger-ui.disable-swagger-default-url=true
```

SSL

If you want to add your own SSL certificate, check the corresponding path. As the provided certificate only supports the application running at `localhost`, you may replace this with your IDS keystore, if you want to host the connector in a productive environment.

```
server.ssl.enabled
server.ssl.key-store-type
server.ssl.key-store
server.ssl.key-store-password
server.ssl.key-alias
configuration.path
configuration.keyStorePassword
configuration.keyAlias
configuration.trustStorePassword
```

Http Connections

For customizing timeout settings for incoming and outgoing requests, you may customize the following lines:

```
http.timeout.connect=10000
http.timeout.read=10000
http.timeout.write=10000
http.timeout.call=10000
```

Not that either the call timeout is used, or the other three values.

Authentication

The application uses Spring Security. Each endpoint behind `/**`, needs a user authentication, except the open IDS endpoint at `/api/ids/data`.

Have a look at the blocked endpoints in the `ConfigurationAdapter` class to add or change endpoints yourself. In case you don't want to provide authentication for your backend maintenance, feel free to remove the corresponding lines.

For changing the default credentials, the properties are located at `spring.security.user.name` and `spring.security.user.password`.

Database

The Dataspace Connector uses Spring Data JPA to set up the database and manage interactions with it. Spring Data JPA supports many well-known relational databases out of the box. Thus, the internal H2 can be replaced by e.g. MySQL, PostgreSQL, or Oracle databases with minimal effort.

To use another database for the Dataspace Connector, follow these steps.

Settings are provided within the `application.properties` at:

```
spring.datasource.url
spring.datasource.driverClassName
spring.datasource.username
spring.datasource.password

spring.h2.console.enabled=false
spring.h2.console.path=/database
spring.h2.console.settings.web-allow-others=true
```

Logging

The Dataspace Connector provides multiple ways for logging and accessing information. Please find a detailed description on how to set up static and runtime configurations here.

Settings are provided within the `application.properties` at:

```
management.endpoints.enabled-by-default=false
management.endpoints.web.exposure.include=logfile, loggers
management.endpoint.loggers.enabled=true
management.endpoint.logfile.enabled=true
management.endpoint.logfile.external-file=./log/dataspaceconnector.log
```

Http tracing is disabled by default: `httptrace.enabled=false`.

Jaeger

If your want to access open telemetry, have a look at this guide. You can customize the deployment with these lines:

```
opentracing.jaeger.udp-sender.host=localhost
opentracing.jaeger.udp-sender.port=6831
opentracing.jaeger.log-spans=true
```

Bootstrapping

If you want to change the base path, which will be used to find properties and catalogs for bootstrapping, you can customize the following line:

```
bootstrap.path=.
```

IDS Settings

URLs of the DAPS for IDS identity management and the Clearing House for contract agreement and data usage logging can be changed within the following lines:

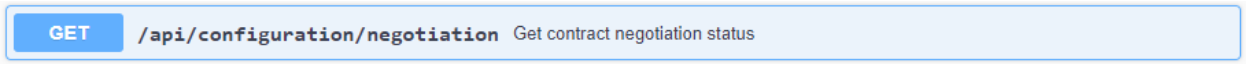
```
daps.token.url=https://daps.aisec.fraunhofer.de
daps.key.url=https://daps.aisec.fraunhofer.de/v2/.well-known/jwks.json
clearing.house.url=https://ch-ids.aisec.fraunhofer.de/logs/messages/
```

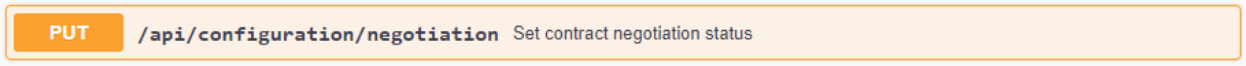
If you leave the Clearing House address blank, the connector will ignore sending IDS messages to it.

Also, for usage control, some settings are provided:

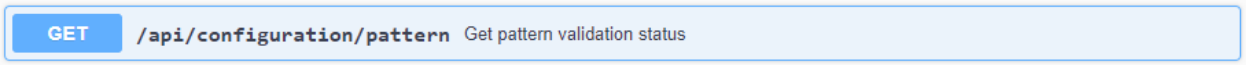
```
policy.negotiation=true
policy.allow-unsupported-patterns=false
policy.framework=INTERNAL
```

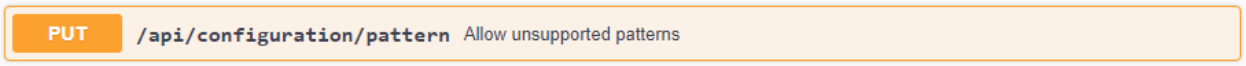
Contract negotiation is enabled by default. This forces other Connectors to refer to a valid contract agreement when requesting data access via an `ArtifactRequestMessage`. If you want to deactivate the policy negotiation, as data provider or data consumer, use the following endpoints or the corresponding line within the `application.properties`.





Note that the Dataspace Connector is able to received resources with usage policies that follow the IDS policy language but not one of the supported patterns. As, by default, the policy check on the data consumer side would not allow accessing data whose policies cannot be enforced, you are able to ignore unsupported patterns with setting the boolean at the endpoint `/api/configuration/pattern` or the property `policy.allow-unsupported-patterns` in the `application.properties` to `true`. As a data consumer, you are bound to concluded contract agreements that are technically mapped to IDS usage policies. Therefore, you have to ensure, that your backend applications technically enforce the usage policies instead.





7. CONFORMITY ANALYSIS

7.1. ANALYSIS OF THE SECURITY STATEMENT

The information that should be included in this section is at least as follows:

a) Evaluator or evaluators in charge of this activity.
b) Reference to the identifier of the evaluated Security Declaration.
c) Results of the evaluator's tasks.
d) No conformities found.
e) Time used for analysis.

7.2. ANALYSIS OF DOCUMENTATION

The information that should be included in this section is at least as follows:

a) Evaluator or evaluators in charge of this activity.
b) Documents analyzed.
c) The approach used to perform the analysis.
d) Results of the evaluator's tasks.
e) No conformities found.
f) Time used for analysis.

7.3. PROVEN FUNCTIONALITIES

The information that should be included in this section is at least as follows:

a) Evaluator or evaluators in charge of this activity
b) Justification of the sample used.
c) Information of each test.
d) Nonconformities found and their associated results.
e) Time used for testing.

For each function tested, the evaluator must fill in the following "Test Case" template:

Test identifier	(Ej.- TEST_0xx)
Proven functionality:	Evaluator:
	Objetive of the test:
Test scenario:	

Procedure	Expected results	Results
Conclusion and verdict		

After meeting the software requirements and having prepared, in the appropriate location, the SSL certificates (server.crt and server.key) information detailed in section 1.2 and section 2.2 of this validation report, it is proceeded the checking installation procedure of the component.

Test identifier	TEST_XXX_YYY
Evaluator: Josu Fernández	**Objetive:** Check installation procedure

Test_Ins_001:

Objective: Clone the repository

Description: clone the MetadataBroker repository to your local environment and check that the docker-compose file is available in the mentioned path.

Initial condition: valid SSL certificates files located in the folder /etc/idscert/localhost

Trigger:

Expected Behavior:

Result:

Test_Ins_002:

Objective: Check installation procedure in Linux

Description:

Initial condition:

Trigger:

Expected Behavior:

Result:

8. VULNERABILITY SCANNING

9. REFERENCES

10. ACRONYMS