# INTRODUCTION

## INFORMATION FROM THE TECHNICAL EVALUATION REPORT

|  |  |
| --- | --- |
| **Reference component** | IDSA Dataspace connector |
| **Version** | 5.1.2 |
| **Author or authors** | Fraunhofer ISST |
| **Approved by** |  |
| **Date** |  |
| **File code** |  |
| **Type of evaluation** |  |
| **Taxonomy of the product** |  |

## DEVELOPER AND TOE INFORMATION

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

# 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.

## FUNCTIONAL DESCRIPTION OF THE TOE

## INVENTORY OF SECURITY FUNCTIONS IDENTIFIED IN THE SECURITY STATEMENT

Integrity checks, encryption, PKI… etc

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

# EXECUTION ENVIRONMENT

## DESCRIPTION OF THE EXECUTION ENVIRONMENT

The Dataspace Connector makes use of Git, Maven (tested on 3.6.3) and requires at least Java 11.

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

## HIPÓTESIS SOBRE EL ENTORNO DE EJECUCIÓN

# EXECUTIVE SUMMARY OF THE EVALUATION

The content expected in this section of the validation report is a summary of the evaluation that is going to be carried out. In this part it is highlighted the most important aspects and the aim is to provide an overview of how the assessment is going to be developed.

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

## **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.

## **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 take action 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.

## **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.

## **Interoperability evaluation.**

As a last test, it is tested that the component can interoperate with other components of the IDS architecture as expected.

# 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.

# COMPONENT INSTALLATION

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](https://github.com/International-Data-Spaces-Association/IDS-G/blob/master/core/DAPS/README.md#dynamic-attribute-token-content) (DAT) from the [Dynamic Attribute Provisioning Service](https://github.com/International-Data-Spaces-Association/IDS-G/blob/master/core/DAPS/README.md) (DAPS). Each message to an IDS participant needs to be signed with a valid [DAT](https://github.com/International-Data-Spaces-Association/IDS-G/blob/master/core/DAPS/README.md#dynamic-attribute-token-content). 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.

## **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.

## **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](https://github.com/International-Data-Spaces-Association/IDS-G/blob/master/core/DAPS/README.md#dynamic-attribute-token-content) from the [DAPS](https://github.com/International-Data-Spaces-Association/IDS-G/blob/master/core/DAPS/README.md) and for received messages, the sent [DAT](https://github.com/International-Data-Spaces-Association/IDS-G/blob/master/core/DAPS/README.md#dynamic-attribute-token-content) will not be checked.

To turn on the [DAT](https://github.com/International-Data-Spaces-Association/IDS-G/blob/master/core/DAPS/README.md#dynamic-attribute-token-content) checking, you need to set the ids:connectorDeployMode to idsc:PRODUCTIVE\_DEPLOYMENT. For getting a trusted certificate, contact [Gerd Brost](mailto:gerd.brost@aisec.fraunhofer.de). 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.

## **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](https://international-data-spaces-association.github.io/DataspaceConnector/Deployment/Database) 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](https://international-data-spaces-association.github.io/DataspaceConnector/Deployment/Logging).

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](https://international-data-spaces-association.github.io/DataspaceConnector/Deployment/Build#docker). 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](https://github.com/International-Data-Spaces-Association/IDS-G/blob/master/core/DAPS/README.md) 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.

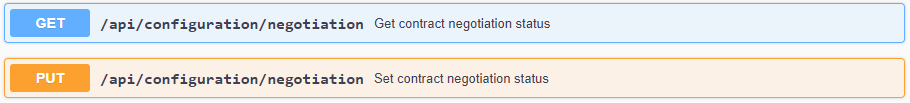
Also, for usage control, some settings are provided:

policy.negotiation=true

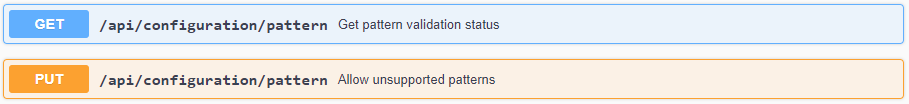
policy.allow-unsupported-patterns=false

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.



# CONFORMITY ANALYSIS

## ANALYSIS OF THE SECURITY STATEMENT

## ANALYSIS OF DOCUMENTATION

## PROVEN FUNCTIONALITIES

# VULNERABILITY SCANNING

# REFERENCES

# ACRONYMS