# INTRODUCTION

## INFORMATION FROM THE TECHNICAL EVALUATION REPORT

|  |  |
| --- | --- |
| **Reference component** | metadata-broker-open-core |
| **Version component** | Version 4.0.3 |
| **Author(s)** | Josu Fernández Pereda |
| **Approved by** |  |
| **Date** | 31/08/2021 |
| **File code** | v2.0 |
| **Type of evaluation** | Evaluation Technical Report |
| **Taxonomy of the product** |  |

## DEVELOPER AND TOE INFORMATION

|  |  |
| --- | --- |
| **Developer Data (Name and Address)** | Fraunhofer IAIS |
| **Developer Contact Details (Name and e-mail)** | contact@ids.fraunhofer.de |
| **Name of the TOE** | IDS Metadata Broker |
| **Version of the TOE** | 4.0.3 |

# DESCRIPTION OF THE TOE

To perform a security analysis, the evaluator must first gain knowledge of the target of evaluation (TOE) by analyzing the available documentation.

The evaluator must then complete the following sections of the Evaluation Technical Report (ETR).

## FUNCTIONAL DESCRIPTION OF THE TOE

This component under evaluation is an implementation of an International Data Spaces (IDS) Metadata Broker, which is a registry for IDS connector self-description documents. It intends to act as a reference component for members of the International Data Spaces Association (IDSA) to help with the implementation of custom broker solutions.

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.

For this version of the component (version 4.0.3) the structure of the component repository is the following:

* Broker-core: this is the main Maven artifact.
* Open-broker-common: this contains the shared code that the broker-core requires.
* Open-index-common: shared functionalities for open-broker-common and further IDS index services.
* Docker: files to deploy the IDS Metadata Broker.

The readme file contains the IDS Metadata Broker 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 Broker.

## INVENTORY OF SECURITY FUNCTIONS IDENTIFIED IN THE SECURITY STATEMENT

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.

# EXECUTION ENVIRONMENT

## DESCRIPTION OF THE EXECUTION ENVIRONMENT

It must be specified the operational environment that is required to enable the execution of the product. Operating system and required software along with their versions.

* Hardware: recommended 20GB of free disk storage.
* Software: recommended Linux based operating system.

Tested on Ubuntu 20.04 and Windows 10.

For the technical evaluation report of this component, it has been tested the version 4.0.3 of the Metadata Broker and it has been run both for Linux and Windows users.

# 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 of the Metadata Broker is based on the following software aspects:

## DOCUMENTATION EVALUATION

The first aspect is focused on verifying the documentation provided for the component, this documentation is available at the GitHub repository [International-Data-Spaces-Association/metadata-broker-open-core: This is the repository of the open-core reference implementation of the IDS Metadata Broker. (github.com)](https://github.com/International-Data-Spaces-Association/metadata-broker-open-core). Here it is specified the necessary procedures for installing, running, and communicating with the component.

It is ensured that the installation procedures, deploying mechanisms, technologies used, and functionalities of the Metadata Broker are correctly specified, so that if the user follows the installation guide of the component it is launched successfully.

## 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 SonarQube 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 used by the component, and if they are solved in the implementation. If further questions are required, the development team is contacted to ensure that all these vulnerabilities have been mitigated.

## API EVALUATION

This aspect studies the component API’s functionalities. As this is the main way to communicate with the metadata broker it is mandatory for it to work exactly as it is explained in the documentation.

The API description is accessible in the path: [IDS-Broker | 1.3.1 | idsa | SwaggerHub](https://app.swaggerhub.com/apis/idsa/IDS-Broker/1.3.1)

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

## INTEROPERABILITY EVALUATION

In the last step, it is tested the Metadata Broker interoperability with other components of the IDS architecture. To test this interoperability, it has been tested the interoperation between the Metadata Broker and the DSC (Data Space Connector) and the Metadata Broker with the DAPS.

DAPS Operation

To test the interoperability between the DAPS and the Metadata Broker. The following scenarios will be checked where only one of them will be the correct one.

Put here the scenarios to be checked

DSC Operation

Put here the scenarios to be checked

CA Operation

Put here the scenarios to be checked

# 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) **PASS**: 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 Metadata Broker.

- **Code safety evaluation**

* Ok. Sonarqube 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 have, 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 Metadata Broker is capable to communicate with the different components of the reference TestBed.

# COMPONENT INSTALLATION

### **Installation Guide**

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 Metadata Broker.

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.

### **1 Prerequisites**

In this section, it is provided some guidance as to recommendations for the number of resources that should be available to smoothly operate the IDS Metadata Broker. The actual number of resources required heavily depends on the load. In case of very little traffic, fewer resources than listed below might be required.

#### **1.1 Hardware**

It is required 2GB of disk space for operating the IDS Metadata Broker, though it is recommended providing at least 20GB of free disk storage to avoid running out of disk space with increasing number of registered items.

To provide enough processing power for all Docker containers, it is recommended using a 64bit quad core processor or better.

#### **1.2 Software**

* **OS**: it is recommended using a Linux based operating system. However, any operating system with a Docker installation can be used (tested on Ubuntu 20.04 and Windows 10). More strict hardware requirements than listed above might apply if a non-Linux operating system is used.
* **Git**: it is used git version 2.25.1 for the installation of this component. However, any version of git higher than version 2.0.0 is sufficient to perform this installation.
* **Docker:** it is used Docker version 19.03.8 for the installation of this component. It is recommended to have a docker version equal to or higher than the one mentioned for this installation.
* **Docker compose:** it is used Docker version 1.25.0 for the installation of this component. It is recommended to have a docker-compose version equal to or higher than the one mentioned for this installation.
* **OpenSSL**: A valid X.509 certificate, signed by a trusted certification authority, it is strongly recommended to avoid warnings about insecure HTTPS connections.

Docker must be installed on the target machine. For the SSL certificate you need to have these two files:

-- **server.crt:** an x509 certificate, either self-signed or from an official CA.

-- **server.key:** the private key for the certificate. The certificate needs to be of .crt format and must have the name server.crt and the file for private key should have the name server.key.

In case your certificate is of .pem format, it can be converted with the following commands, which require OpenSSL to be installed:

openssl x509 -in mycert.pem -out server.crt

openssl rsa -in mykey.pem -out server.key

mkdir cert

mv server.crt cert/

mv server.key cert/

### **2 Installation Steps**

#### **2.1 Cloning the repository**

Use the following command to clone the repository that contains the Metadata Broker in the current path of your terminal:

git clone <https://github.com/International-Data-Spaces-Association/metadata-broker-open-core.git>

#### **2.2 Prepare the SSL certificate**

On your host system, create the following directory and put the files server.crt and server.key into this directory:

For Linux users:

Create the following directory path → /etc/idscert/localhost

For Windows users:

Create the following directory path → C:\etc\idscert\localhost

#### **2.3 Configuring the Docker-Compose File**

Once the repository is cloned, the docker-compose file will be found in this path:

metadata-broker-open-coredocker/composefiles/Meta-Data-Broker/broker-localhost/docker-compose.yml

The most crucial part of adapting the configuration is to provide the correct location of the X.509 certificate created in **Section 1.2** in the broker-reverseproxy service.

**For Linux users:** if the location of the certificate is “/home/ids/cert”, the corresponding configuration in the yml file is:

services: broker-reverseproxy:

image: registry.gitlab.cc-asp.fraunhofer.de:4567/eis-ids/broker/reverseproxy

volumes:

- /home/ids/cert: /etc/cert/

[…]

**For Windows users:** if the location of the certificate is “c:/etc/ids/cert”, the corresponding configuration in the yml file is:

services: broker-reverseproxy:

image: registry.gitlab.cc-asp.fraunhofer.de:4567/eis-ids/broker/reverseproxy

volumes:

- c:/etc/ids/cert:/etc/cert/

[…]

#### **2.4 Downloading the Docker Images**

All the IDS Metadata Broker Docker images are hosted at the GitLab of Fraunhofer IAIS and no credentials are needed to download the images. The following command is for pulling all docker images:

docker-compose pull

Note that this command should be executed in the same path of docker-compose.yml file.

#### **2.5 Starting up the IDS Metadata Broker**

To start up the IDS Metadata Broker, run the following command inside the directory of the docker-compose.yml file:

docker-compose up –d

This process can take several minutes to complete. You can test whether the IDS Metadata Broker has successfully started by opening [https://localhost](https://localhost/). The result should be a JSON document, providing some general metadata about the IDS Metadata Broker.

Furthermore, the docker-compose logs command can be used to access the logs for a docker-compose.yml file.

#### **2.6 Interacting with the IDS Metadata Broker**

The IDS Metadata Broker accepts and sends messages according to the IDS information model. This model uses the Resource Description Framework (RDF) to leverage the power of linked data. Many examples about representations of IDS concepts can be found at:

<https://github.com/International-Data-Spaces-Association/InformationModel/tree/develop/examples>

The multipart endpoint of IDS Metadata Broker is “/infrastructure”. If the IDS Metadata Broker is running using docker-compose as mentioned earlier, an HTTP POST request can be sent to interact with it. We provide some example messages, illustrating all core functions of the IDS Metadata Broker in this [postman collection](https://www.getpostman.com/collections/1cecd0def2941a993e80).

In addition to the multipart endpoint, the IDS Metadata Broker also serves a prototypical [IDS-REST](https://www.getpostman.com/collections/01d6bf596f67303c08ce) endpoint at “/catalog”. This endpoint will reach a non-prototype state soon after the final specification of the IDS-REST protocol.

### **3 Updating and Stopping the IDS Metadata Broker**

#### **3.1 Updating the IDS Metadata Broker**

To update an existing installation of the IDS Metadata Broker, first repeat the steps explained in section “Downloading the Docker Images”. Containers can be either hot updated or restarted to apply the changes.

To hot update a container, run the following command:

docker-compose up -d --no-deps --build <container name>

Alternatively, one can restart the entire service by running:

docker-compose down

docker-compose up –d

#### **3.2 Stopping the IDS Metadata Broker**

To stop the Broker run the following in the terminal in the same path as the docker-compose.yml file:

docker-compose down

# CONFORMITY ANALYSIS

## ANALYSIS OF THE SECURITY STATEMENT

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

1. Evaluator or evaluators in charge of this activity.
2. Reference to the identifier of the evaluated Security Declaration.
3. Results of the evaluator's tasks.
4. No conformities found.
5. Time used for analysis.

## ANALYSIS OF DOCUMENTATION

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

1. Evaluator or evaluators in charge of this activity.
2. Documents analyzed.
3. The approach used to perform the analysis.
4. Results of the evaluator's tasks.
5. No conformities found.
6. Time used for analysis.

## PROVEN FUNCTIONALITIES

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

1. Evaluator or evaluators in charge of this activity
2. Justification of the sample used.
3. Information of each test.
4. Nonconformities found and their associated results.
5. 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:

# VULNERABILITY SCANNING

For this section use SonarQube and put the results obtained with some explanation.

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

1. Evaluator or evaluators in charge of this activity.
2. Methodology used to perform vulnerability analysis.
3. Vulnerability analysis of the TOE (Process, Tools used, mechanisms analyzed, etc.).
4. Listing potential vulnerabilities of the product together with the analysis of the resistance of the mechanisms/functions.
5. Nonconformities found.
6. Time used for analysis.

# REFERENCES

[CC ] Common Criteria for Information Technology Security Evaluation. Its latest version approved and published on the Certification Body website should be considered. (hhtps://oc.ccn.cni.es)

Evaluation: Evaluation Methodology.

Its latest version approved and published on the Certification Body website should be considered. (hhtps://oc.ccn.cni.es)

# ACRONYMS

|  |  |
| --- | --- |
| **TOE** | Target Of Evaluation – Objeto a evaluar |