



## Towards an Interoperable MARinE Knowledge GRAPH

Project Title

Project Acronym MareGraph

Grant Agreement No. 101100771

Start Date of Project 23/01/2022

Duration of Project 36 Months

## D4.3 - Testbed publishing and republishing of the data

|                              |   |
|------------------------------|---|
| Work Package                 | WP4 – Data Architecture                                   |
| Lead Author (Org)            | Simon Claus, & Samuel Van Ackere (Digitaal Vlaanderen)    |
| Contributing Author(s) (Org) | Koenraad Verduyn, Sander Van Dooren (Digitaal Vlaanderen) |
| Due Date                     | M24   |
| Date                         | 30.12.2024  |
| Version                      | V1.0  |

Dissemination Level

- ☒ PU: Public  
☐ SEN - Sensitive  
☐ RE: Restricted to a group specified by the consortium (including the Commission)

|   |    |
|---|----|
| D4.3 - Testbed publishing and republishing of the data .....              | 1  |
| 1. European Interoperability Testbed.....                                 | 2  |
| 2. Adaptation for VSDS.....   | 3  |
| 2.1 Part II: Conformance testing of domain model with SHACL .....         | 3  |
| 2.2 TestSuite/TestCase .....  | 4  |
| 2.3 GUI .....   | 6  |
| 3. Use case: Testing conformance MAREGRAPH Data assets with Testbed ..... | 10 |

## 1. European Interoperability Testbed

The European Interoperability Test Bed (ITB) is a service by the European Commission's DIGIT that supports conformance testing of IT systems. It is available as downloadable software or through an online installation and features an intuitive Web interface for setting up and running tests. This tool is vital for ensuring IT systems can interact seamlessly and supports technical and business-level testing scenarios. You can find more detailed information and resources on the [official website](#).

The testbed offers validators to validate RDF, XML, JSON and CSV, but it also allows a custom validator plugin.

The GITB project represents a CEN standardisation initiative funded by the European Commission's DG GROW to provide the specifications for a generic interoperability test bed and their implementation in the form of the GITB testbed software. The focus of these specifications and software is not any kind of testing (e.g. performance, regression, penetration) but rather conformance and interoperability testing. Simply put, conformance testing verifies that the requirements of a given specification are met. In contrast, interoperability testing is a second step to verify that two or more parties can interact as expected.

A key element of the GITB specifications is the set of GITB test service APIs. SOAP web service interfaces (defined using WSDLs and XSDs) allow extension of the core GITB test bed software using decoupled components that offer domain-specific capabilities.

Following the initial work from the CEN GITB workgroup, the maintenance and evolution of the GITB specifications has been taken over by the European Commission, specifically, the [Interoperability Test Bed Team](#) of DIGIT D.2. DIGIT D.2 maintains the GITB software and specifications based on the needs of the testing community and carries out evolutive maintenance with regular releases.

## 2. Adaptation for VSDS

### 2.1 Part II: Conformance testing of domain model with SHACL

Building further upon the GITB Testbed, we added functionality to validate Linked Data Event Streams (LDES), the technical standard being used in MARGERAPH. The LDES Validation Service for the GITB Testbed facilitates data validation via a Linked Data Event Stream within a specific domain. The service operates through the following process:

To achieve this, a Github Repository has been set up (<https://github.com/Informatievlaanderen/VSDS-TestBed-Shacl-Validator.git>) that extends the GITB testbed functionality. The two following services were added :

**Replication Service:** The testbed has been expanded to include an LDES client instance replicating the given LDES endpoint into a GraphDB instance.

This service is responsible for creating an LDIO pipeline, checking if the LDES Client is still replicating and destroying the pipeline afterwards. It is available through the endpoint's WSDL: <http://localhost:8080/services/process?wsdl>. This processing service has three operations:

#### **startReplicating**

This operation creates the pipeline, which immediately starts the replication process.

Input parameters:

| Name     | Description                     | Required |
|----------|---------------------------------|----------|
| ldes-url | the url of the LDES to validate | true     |

#### **haltWhenReplicated**

This operation is responsible for polling and returning the status of the LDES Client.

#### **destroyPipeline**

This operation is responsible for deleting the LDIO pipeline and the GraphDB repository.

This service uses the VSDS components (<https://github.com/Informatievlaanderen/VSDS-Linked-Data-Interactions.git>), which were developed as part of the Flemish Smart DataSpace project. More info on this project can be found here : <https://www.vlaanderen.be/digitaal-vlaanderen/onze-diensten-en-platformen/vlaamse-smart-data-space>.

In particular, LDES Client and Repository Sink components have been used to create a dynamically configured pipeline that ingests all members from the LDES endpoint given as input and writes the data to a GraphDB triplestore instance.

The GITB testbed has been configured to instantiate the pipeline through API and check the pipeline status until this status reaches 'synchronising'. The pipeline is then destroyed. In case of errors during the ingestion, the testbed reports an error in the replication phase of the test.

### ShaclValidationService

This service is responsible for executing the SHACL validation and is accessible through the endpoint's WSDL: <http://localhost:8080/services/validation?wsdl>

Input parameters:

| Name        | Description   | Required |
|-------------|---|----------|
| shacl-shape | the SHACL shape that will be used to validate the server against to | true     |

This validation service requires in the validation call two parameters:

1. **ldes-url**: the URL of the LDES to validate
2. **shacl-shape**: the SHACL shape that will be used to validate the server against to

This is done by calling the built-in SHACL validation features of GraphDB through the API and providing the SHACL shapes in TTL format. The validation result is fed back through API to the testbed and processed as part of the report.

The testbed can be spun up by cloning the abovementioned Github Repository and starting the docker commands. The testbed is then made available as a web application with the GITB user interface. Documentation on username/password and other setup tasks is included in the repository. For setting up a conformance test, the workflow is identical to other conformance tests as described in the GITB testbed documentation and won't be repeated here.

## 2.2 TestSuite/TestCase

The GITB testbed needs a test suite setup to function. The necessary files to set up a test suite and the corresponding test case for SHACL validation are included in the Github Repository. A scriptlet file is provided that introduces the necessary user feedback in the testbed GUI, and the following test case is provided to do a SHACL validation (XML format):

```
<testcase id="ts1_tc1" xmlns="http://www.gitb.com/td1/v1/"
xmlns:gitb="http://www.gitb.com/core/v1/">
  <metadata>
    <gitb:name>[TC1] Validate LDES</gitb:name>
    <gitb:version>1.0</gitb:version>
    <gitb:description>Validate an LDES against a specific SHACL shape</gitb:description>
  </metadata>
```

```

<actors>

  <gitb:actor id="LDESServer" role="SUT"/>

  <gitb:actor id="SHACLValidator"/>

</actors>

<steps stopOnError="true">

  <interact id="validationData" desc="Setup parameters to validate the LDES">

    <request desc="URL of the LDES to validate" name="ldesUrl"/>

    <request desc="SHACL shape that must be used for validating" name="shaclShape"
inputType="UPLOAD"/>

    <request desc="Amount of seconds between each polling attempt"
name="pollingInterval" inputType="TEXT"/>

  </interact>

  <assign to="delayDuration">concat($validationData{pollingInterval}, '000')</assign>

  <assign to="addresses{processing}">"http://testbed-shacl-
validator:8080/services/process?wsdl"</assign>

  <assign to="addresses{validation}">"http://testbed-shacl-
validator:8080/services/validation?wsdl"</assign>

  <call id="validateLdes" path="scriptlets/validate-ldes.xml">

    <input name="ldesUrl">$validationData{ldesUrl}</input>

    <input name="shaclShape">$validationData{shaclShape}</input>

    <input name="delayDuration">$delayDuration</input>

    <input name="addresses">$addresses</input>

  </call>

</steps>

<output>

  <success>

    <default>"Test session completed successfully."</default>

  </success>

  <failure>

    <default>

      "Test session failed. Please check the failed step report and the test session
log for details."

    </default>

  </failure>

</output>

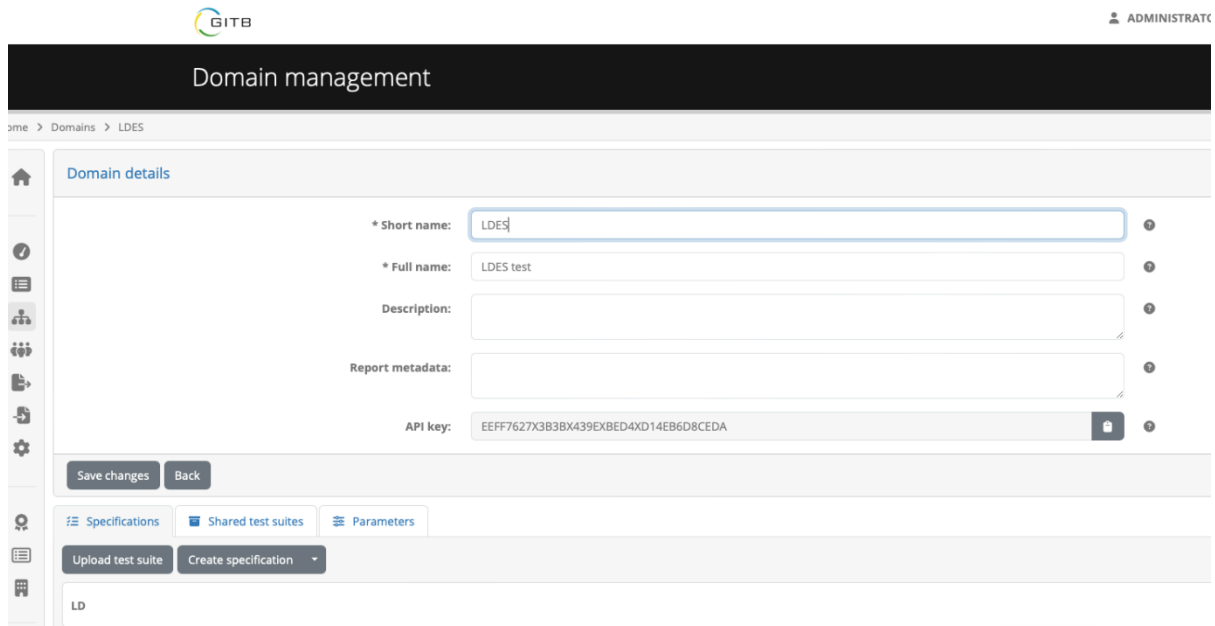
</testcase>

```

In case the procedures need to be changed (for instance, for automated SHACL validation with URL instead of filename content), the corresponding scriptlet file and/or test case file can be adapted without reprogramming.

## 2.3 GUI

The following sections describe the UI for using the testbed for SHACL Validation. The first image shows the general UI with the active Domain management tab.

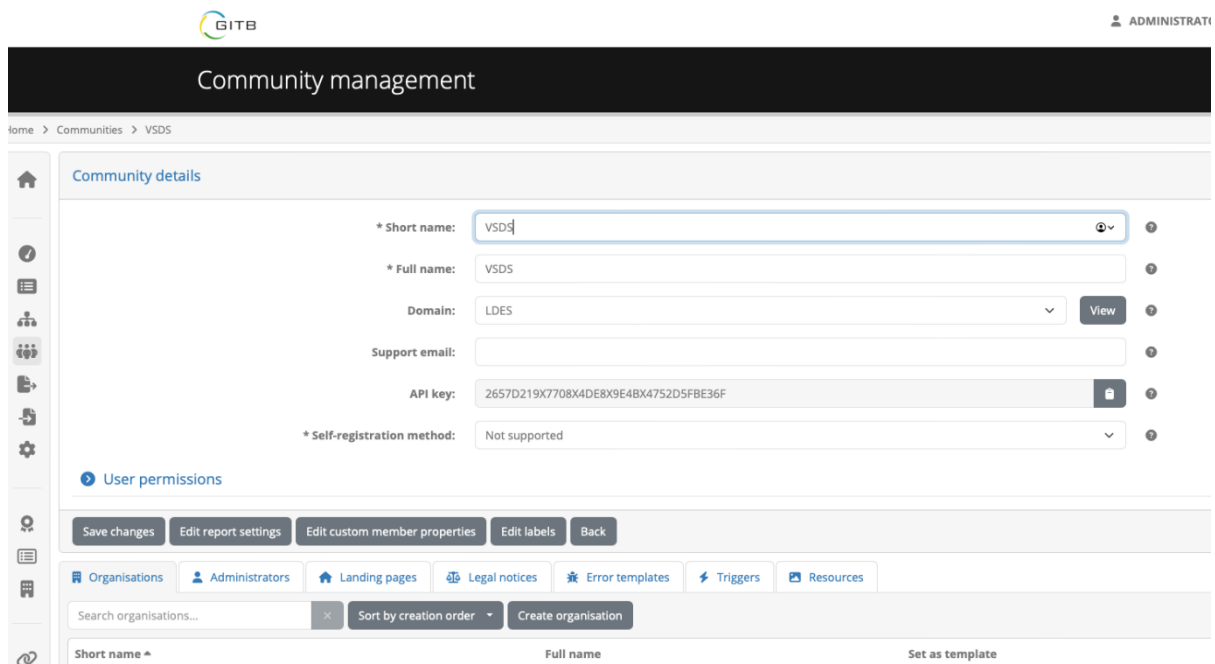


The screenshot shows the 'Domain management' interface. At the top, there's a header with the GITB logo and a user profile 'ADMINISTRATOR'. The main title is 'Domain management'. Below it, a breadcrumb trail reads 'Home > Domains > LDES'. The left sidebar contains various icons for navigation. The main content area is titled 'Domain details' and contains several form fields:
 

- \* Short name: LDES
- \* Full name: LDES test
- Description: (empty text area)
- Report metadata: (empty text area)
- API key: EEF7627X3B3BX439EXBED4XD14EB6D8CEDA

 At the bottom of the form are 'Save changes' and 'Back' buttons. Below the form, there are tabs for 'Specifications', 'Shared test suites', and 'Parameters'. Under 'Specifications', there are buttons for 'Upload test suite' and 'Create specification'. A table below shows one entry with the short name 'LD'.

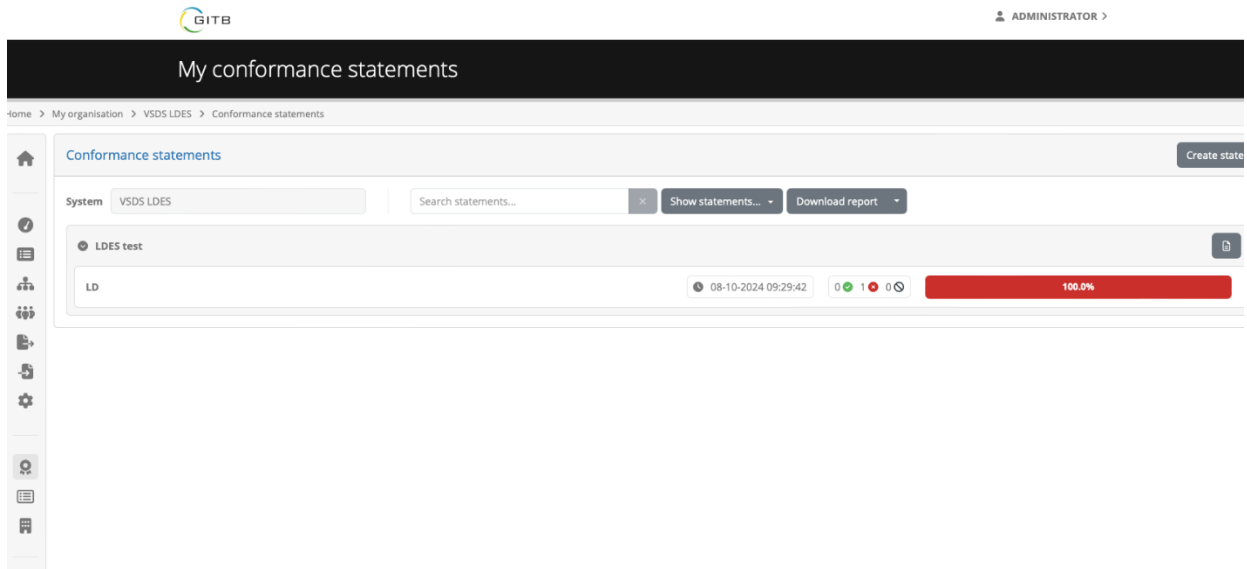
Within the testbed, a Community is managed in this GUI:



The screenshot shows the 'Community management' interface. At the top, there's a header with the GITB logo and a user profile 'ADMINISTRATOR'. The main title is 'Community management'. Below it, a breadcrumb trail reads 'Home > Communities > VSDS'. The left sidebar contains various icons for navigation. The main content area is titled 'Community details' and contains several form fields:
 

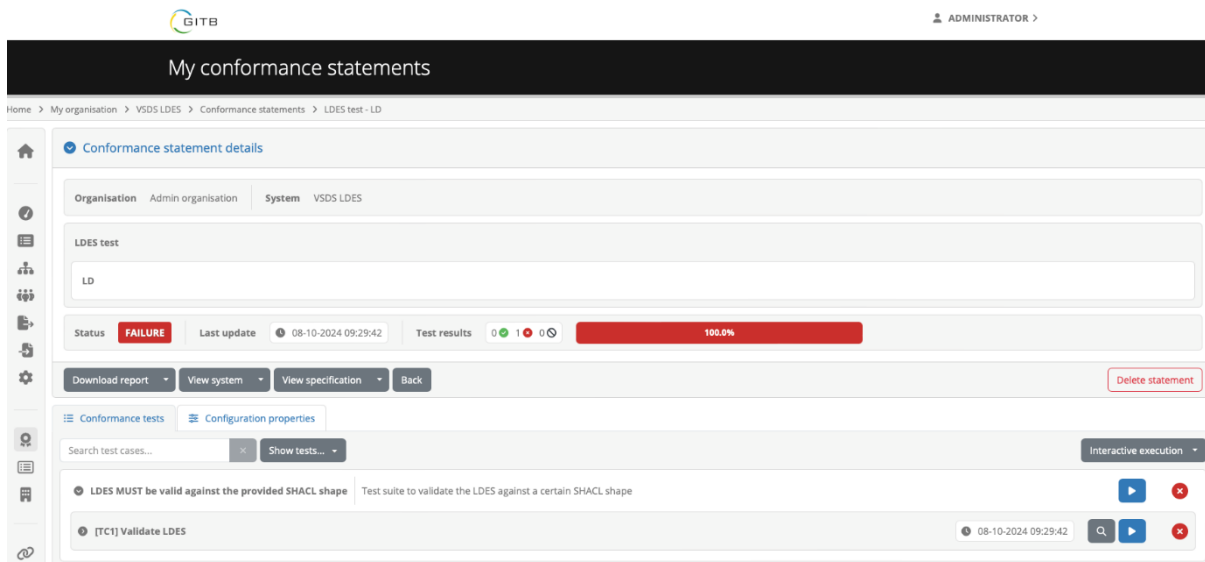
- \* Short name: VSDS
- \* Full name: VSDS
- Domain: LDES (with a 'View' button)
- Support email: (empty text area)
- API key: 2657D219X7708X4DE8X9E4BX4752D5FBE36F
- \* Self-registration method: Not supported

 Below the form is a section titled 'User permissions'. At the bottom of the form are buttons for 'Save changes', 'Edit report settings', 'Edit custom member properties', 'Edit labels', and 'Back'. Below this, there are tabs for 'Organisations', 'Administrators', 'Landing pages', 'Legal notices', 'Error templates', 'Triggers', and 'Resources'. Under 'Organisations', there is a search bar 'Search organisations...', a 'Sort by creation order' dropdown, and a 'Create organisation' button. A table below shows columns for 'Short name', 'Full name', and 'Set as template'.



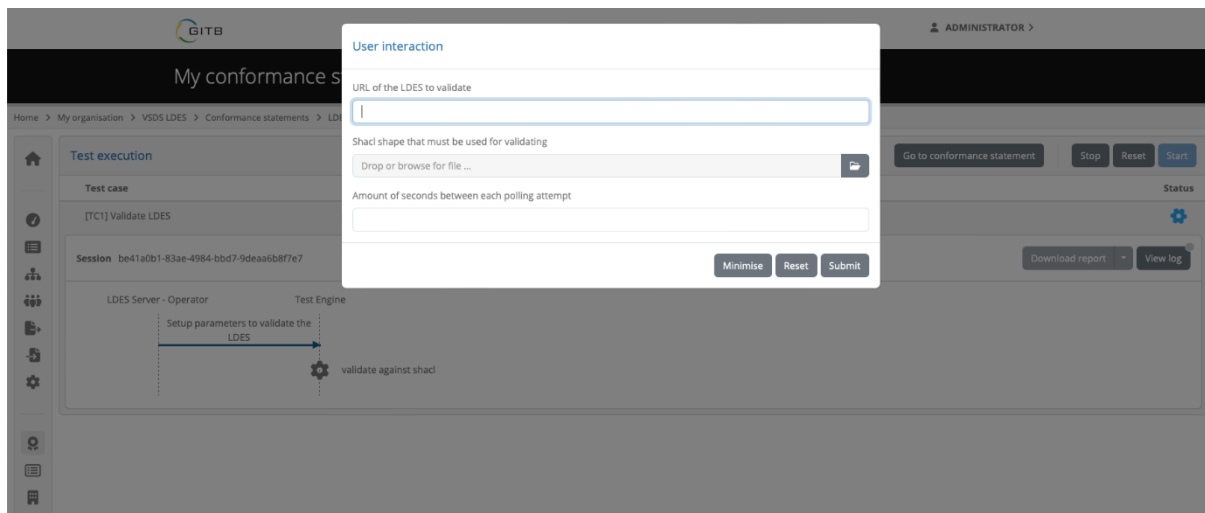
The screenshot shows the 'My conformance statements' page. At the top, there's a header with the GITB logo and the user role 'ADMINISTRATOR'. Below the header, the page title is 'My conformance statements'. A breadcrumb trail indicates the path: Home > My organisation > VSDS LDES > Conformance statements. The main content area shows a table of conformance statements. The first statement is for 'LDES test' with a status of 'FAILURE' and a progress bar at 100.0%.

This shows a failed conformance test for LDES. This is elaborated in the following screen:



The screenshot shows the 'Conformance statement details' page. The page title is 'Conformance statement details'. It shows the 'LDES test' statement with a status of 'FAILURE' and a progress bar at 100.0%. Below the statement details, there's a section for 'Conformance tests' which lists two tests: 'LDES MUST be valid against the provided SHACL shape' and '[TC1] Validate LDES'. The first test is highlighted, showing its description: 'Test suite to validate the LDES against a certain SHACL shape'.

Starting a rerun of the conformance test looks like this:



The screenshot shows the 'Test execution' page with a 'User interaction' dialog box open. The dialog box has three input fields: 'URL of the LDES to validate', 'Shacl shape that must be used for validating', and 'Amount of seconds between each polling attempt'. There are also buttons for 'Minimise', 'Reset', and 'Submit'. The background shows the 'Test execution' page with a test case '[TC1] Validate LDES' and a session ID 'be41a0b1-83ae-4984-bbd7-9deaa6b8f7e7'. A diagram below the session ID shows the workflow: 'LDES Server - Operator' -> 'Setup parameters to validate the LDES' -> 'Test Engine' -> 'validate against shacl'.

In case of a successful conformance test, the following report is generated:

## Test Case Report

### Overview

**Organisation:** Admin organisation

**System:** LDES Server AWV

**Domain:** LD

**Test name:** [TC1] Validate LDES

**Specification:** LDES

**Description:** Validate an LDES against a specific SHACL shape

**Actor:** LDES Server

**Result:** **SUCCESS**

**Start time:** 18/11/2024 12:39:24 **End time:** 18/11/2024 12:41:29

Test session completed successfully.

### References

**Annexes:** [Test session log](#)

## Step reports

[Step 1:](#) Setup parameters to validate the LDES ✓

[Step 2:](#) Check if replication has ended ✓

[Step 3:](#) validate against shacl ✓

### Step 1: Setup parameters to validate the LDES

[Top](#)

#### Overview

**Date:** 18/11/2024 12:40:53

**Result:** **SUCCESS**

#### Report data

##### URL of the LDES to validate

`https://brugge-ldes.geomobility.eu/observations/by-time?year=2023&month=05&day=25&hour=21`

##### Shacl shape that must be used for validating

[File content]

##### Amount of seconds between each polling attempt

1



Step 2: Check if replication has ended

[Top](#)

## Overview

**Date:** 18/11/2024 12:41:27**Result:** **SUCCESS**

Step 3: validate against shacl

[Top](#)

## Overview

**Date:** 18/11/2024 12:41:27**Result:** **SUCCESS****Findings:** 0 error(s), 0 warning(s), 1 message(s)

## Report details

**i** @prefix sh: <http://www.w3.org/ns/shacl#> . @prefix rsx: <http://rdf4j.org/shacl-extensions#> . @prefix dash: <http://datashapes.org/dash#> . @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> . @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . @prefix owl: <http://www.w3.org/2002/07/owl#> . @prefix xsd: <http://www.w3.org/2001/XMLSchema#> . @prefix rdf4j: <http://rdf4j.org/schema/rdf4j#> . \_:g31d0e4812d24b9b9995082a080863b811 a sh: ValidationReport; sh:conforms true; rdf4j:truncated false .

In case of a failed test due to failure to ingest the LDES stream, the following is reported :

## Test Case Report

## Overview

**Organisation:** Admin organisation**System:** LDES Server AWV**Domain:** LD**Test name:** [TC1] Validate LDES**Specification:** LDES**Description:** Validate an LDES against a specific SHACL shape**Actor:** LDES Server**Result:** **FAILURE****Start time:** 18/11/2024 11:13:22 **End time:** 18/11/2024 11:13:47

Test session failed. Please check the failed step report and the test session log for details.

## References

**Annexes:** [Test session log](#)

Step 2: Check if replication has ended

[Top](#)

## Overview

**Date:** 18/11/2024 11:13:45**Result:** **FAILURE**

If the test has failed due to non-conformance to the SHACL shape, the following is reported:

Funded by  
the European Union

Step 2: Check if replication has ended

[Top](#)

## Overview

Date: 13/11/2024 10:09:20

Result: **SUCCESS**

Step 3: validate against shacl

[Top](#)

## Overview

Date: 13/11/2024 10:09:23

Result: **FAILURE**

Findings: 1 error(s), 0 warning(s), 0 message(s)

## Report details

```

✖ @prefix sh: <http://www.w3.org/ns/shacl#> . @prefix rsx: <http://rdf4j.org/shacl-extensions#> . @prefix dash: <http://datashapes.org/dash#> . @prefix rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#> . @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . @prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> . @prefix rdf4j: <http://rdf4j.org/schema/rdf4j#> . _:6970a54172ee41b2b279a3403ae2d1d164 a sh:
ValidationReport; sh:conforms false; rdf4j:truncated true; sh:result _:6970a54172ee41b2b279a3403ae2d1d165, _:6970a54172ee41b2b279a3403ae2d1d166, _:
6970a54172ee41b2b279a3403ae2d1d167, _:6970a54172ee41b2b279a3403ae2d1d168, _:6970a54172ee41b2b279a3403ae2d1d169, _:
6970a54172ee41b2b279a3403ae2d1d170, _:6970a54172ee41b2b279a3403ae2d1d171, _:6970a54172ee41b2b279a3403ae2d1d172, _:
6970a54172ee41b2b279a3403ae2d1d173, _:6970a54172ee41b2b279a3403ae2d1d174, _:6970a54172ee41b2b279a3403ae2d1d175, _:
6970a54172ee41b2b279a3403ae2d1d176, _:6970a54172ee41b2b279a3403ae2d1d177, _:6970a54172ee41b2b279a3403ae2d1d178, _:
6970a54172ee41b2b279a3403ae2d1d179, _:6970a54172ee41b2b279a3403ae2d1d180, _:6970a54172ee41b2b279a3403ae2d1d182, _:

```

Page 3 of 995

This includes the SHACL violations of the individual members of the stream. Take note: the testbed is configured to test every LDES member of the LDES stream. This can result in an extensive report (the example here is 950 pages long) when deviations with the expected SHACL shape (like missing classes) are present in every member. This can also result in long test times with large LDES streams. The LDES client can process fragments, so a good fragmentation strategy is perhaps advised when used for testing purposes.

This is by design choice, as it allows us to state with certainty that the entire data stream complies with the provided SHACL shape.

### 3. Use case: Testing conformance MAREGRAPH Data assets with Testbed

The expanded testbed has been used with the Marine Species and Marine regions feeds. The SHACL shapes for the Taxon name have been used with the marine Species feed and have run a successful test. Screenshots above show the LDES Endpoint <https://aphia.org/feed> tested against the Taxonomy shacl file from [https://github.com/MareGraph-EU/assets/blob/feature/shacl\\_shapes/ontologies/taxon-name/latest/taxon-name.sh.ttl](https://github.com/MareGraph-EU/assets/blob/feature/shacl_shapes/ontologies/taxon-name/latest/taxon-name.sh.ttl). The results is an identical report on shacl validation for all shapes in the feed.