ETSI GR CIM 015 V2.1.1 (2024-04)

Context Information Management (CIM); NGSI-LD Testing Environment Validation

Disclaimer

The present document has been produced and approved by the cross-cutting Context Information Management (CIM) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
RGR/CIM-0015v211

Keywords
API, IoT, NGSI-LD, testing

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) cross-cutting Context Information Management (CIM).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Executive summary

The present document presents the results of the validation of the NGSI-LD Test Suite and its test environment. Besides the testing environment validation, it also provides experts who are implementing the NGSI-LD API specification a description of how to efficiently run the Test Suite and how it can be extended to add more permutations in the Test Cases. The present document concludes by listing the envisioned future work on the Test Suite to extend it and improve it as well as to maintain it for future versions of the NGSI-LD API.

Introduction

The ISG CIM group has defined an API for exchange of information contextualized in time, space and relation to other information using a property graph model with the intent that the associated protocol (called NGSI-LD) becomes the "glue" between all kinds of applications and databases associated with services for Smart Cities, Smart Agriculture, Smart Manufacturing, etc.

To be successful, the NGSI-LD API specification needs to be well understood and well implemented. The community of users will not be solely highly professional engineers employed by big companies but will include many small teams and SMEs and even hobbyists. Therefore, it is essential that the developers have access to not only the standard but also a test specification and a testing environment to check that their work is (and remains) conform to the ETSI NGSI LD specification.

The developers will usually write integration tests to validate the behaviour of their NGSI-LD implementation, but it is important to assert compliance to the specification based on a test suite agreed by the group creating the API specification, i.e. ETSI ISG CIM. Therefore, it is very important to create a set of ETSI-approved test cases.

What is more, the existence of such a test suite will likely help to increase the adoption of the NGSI-LD specification by giving developers a ready to use and extensive set of sample tests.

1 Scope

The present document presents the experience of running the Test Suite against a set of open-source context brokers implementing the NGSI-LD specifications.

It also gives some guidelines on how to use the Test Suite and how to extend it for simple use-cases like adding new permutations in a Test Case.

Finally, it presents some ideas for later improvements of the Test Suite.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI GS CIM 009 (V1.5.1): "Context Information Management (CIM); NGSI-LD API".

[i.2] ETSI GS CIM 012 (V2.1.1): "Context Information Management (CIM); NGSI-LD Test Suite Structure".

3 Definition of terms, symbols and abbreviations

3.1 Terms

Void.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Programming Interface
CSR	Context Source Registration
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
NGSI-LD	Next Generation Service Interfaces Linked Data
TC	Test Case

XML eXtensible Markup Language

4 Tests Execution Options

4.1 Introduction

All the NGSI-LD Test Cases can be executed by launching one single command, but it is also possible to run subsets of those tests.

By specifying a folder when launching the Test Suite, all the tests inside that folder and subsequent subfolders will be executed. Folder selection is an easy way of selecting different tests groups. That also makes the folder structure an important asset on how to run tests subsets.

The generic command to execute tests is:

```
robot [options] robot_files
```

Where the arguments are:

- options: options that can be included in the command;
- robot_files: path to the file or folder where the tests files are stored.

4.2 Executing all tests

The easiest way to run all the tests is to select a folder that has all the tests inside of it or inside of all its subfolders. In that case, the folder path "./TP/NGSI-LD" is a good choice.

The matching command is:

```
robot ./TP/NGSI-LD
```

4.3 Executing tests subsets from a folder

When a folder is selected, it will only run the tests inside of it or inside of all its subfolders. For instance, to run all the tests related to the creation of an entity, the command is:

```
robot ./TP/NGSI-LD/ContextInformation/Provision/Entities/CreateEntity
```

4.4 Executing tests from a single Test Case

To run all the tests in a specific Test Case file, e.g. 001_01.robot, the command is:

```
robot ./TP/NGSI-
LD/ContextInformation/Provision/Entities/CreateEntity/001_01.robot
```

4.5 Results output

An important feature is to output results when running tests. The results are comprised of three different files:

- log.html: logs information from the executed tests in HTML format
- output.xml: logs information from the executed tests in XML format
- report.html: shows information about the success/failures of the executed tests

In case the ErrorListener has been activated (see clause 4.8 for a description of its behaviour), two new specific files are generated:

- errors.log: logs information from the failed tests in text format
- errors.md: logs information from the failed tests in Markdown format

The command option to get the results output in a specific folder is -outputdir, followed by the path to the folder where the files will be created:

```
robot --outputdir ./results ./TP/NGSI-LD
```

4.6 Overriding base URL

Overriding the base context broker URL can be dynamically specified by using the option variable and selecting the new URL. The command is:

```
robot --variable url:"new_URL" ./TP/NGSI-LD
```

4.7 Executing tests by specific tag

Another way of running tests subsets is executing tests matching a specified tag (see clause 6 for a list of available tags). E.g. for running tests related to an entity creation, the tag used is e-create, and the command is:

```
robot --include e-create ./TP/NGSI-LD
```

4.8 Rerun failed tests

Another useful option is to execute tests that failed during a previous execution of the Test Suite. To use this feature the output.xml file is needed, and the command is:

```
robot --rerunfailedsuites ./results/output.xml
```

4.9 Generate log file of failed tests

Another useful option is to execute tests and generate a log file only for those test cases that failed. To use this feature, the ErrorListener listener has to be used, and the command is:

```
robot --listener libraries/ErrorListener.py
```

This execution provides two special files in text and Markdown format located in the folder where the robot is executed. In case that the `--outputdir` <dir> option is defined when executing the Test Suite, the files will be located in this <dir>.

Additionally, if it is defined the GitHub variables, it is possible to create a GitHub Issue or Issues with the information of the error(s) in the corresponding Context Broker repository.

4.10 Management of Test Cases

4.10.0 Foreword

This script is designed to facilitate the selection and execution of the Test Suites, especially if not all the endpoints of the API have been implemented for a specific Context Broker. This script provides a set of commands to enable or disable Test Cases, Test Suites or Collection (Test Suite Groups), visualize the current status of the different Test Cases, execute the selected Test Cases and perform other related operations as described below.

The `tsm` script generates a pickle file named `tsm.pkl`, which stores the tuples list corresponding to each Test Case with the information:

```
(switch, running status, Test Case long name)
```

where the values and their meaning are the following:

- switch:
  - **ON:** the Test Case is on, which means that it will be executed by the script.
  - **OFF:** the Test Case is off, and therefore is not selected to be executed by the script.
  - **MISSING:** the Test Case is not any more in the Test Suite Structure. An update operation should be run to update the `tsm.pkl` with the current set of available Test Cases in the filesystem.
  - **NEW:** new Test Case discovered by the `tsm` after an update operation.
- status:
  - **PASSED:** the robot framework executes the Test Case with result `PASS`.
  - **FAILED:** the robot framework executes the Test Case with result `FAIL`.
  - **PENDING:** the Test Case is pending to be executed by the robot framework.
- test case long name: the Test Case long name set by robot framework based on the Test Suite number and the Test Case name (e.g. `NGSILD.032 02.032_02_01 Delete Unknown Subscription`).

4.10.1 Installation

The `tsm` script is integrated with arguments auto-completion for bash, therefore it is needed the installation of the `argcomplete` python package on your system:

```
pip install argcomplete
```

and to enable the completion after the installation executing the following command:

```
activate-global-python-argcomplete
```

and then:

```
eval "$(register-python-argcomplete tsm)"
```

Now, it is possible to autocomplete the commands and show possible options executing the script. Also -help argument is available to obtain more information about the options.

4.10.2 Execution

The `tsm` cases update command updates the `tsm.pkl` file with all the .robot files under the local path. If the pickle file does not exist, it is created. After the creation of this file, it is possible to execute the script to maintain and run the selected Test Cases from the pickle file. The list of commands is the following:

Commands:

- Test Cases (cases)
  - Switch ON Test Cases

    ```
    tsm cases on [test_cases]
    ```

  - Switch OFF Test Cases

    ```
    tsm cases off [test_cases]
    ```

```
tsm cases off "NGSILD.032 01.032_01_02 InvalidId"
```

- List Test Cases based on the specific flag.

```
tsm cases list [on, off, missing, new, passed, failed, pending, all]
```

```
tsm cases list ./TP/NGSI-LD/CommonBehaviours
```

- Run Test Cases that are enabled

```
tsm cases run [on, off, missing, new, passed, failed, pending,
[test_cases]]
```

```
tsm cases run NGSILD.048\ 01.048_01_06\ Endpoint\ post\
/temporal/entities/
```

```
tsm cases run pending
```

- Update the pickle file with the current Test Cases

```
tsm cases update
```

- Clean Test Cases, remove the Test Cases that were marked as MISSING

```
tsm cases clean
```

- Test Suites (suites)
  - Switch ON Test Suites

```
tsm suites on [suites]
```

  - Switch OFF Test Suites

```
tsm suites off [suites]
```

- Test Collections (collections)
  - Switch ON Test Collections

```
tsm collections on [collections]
```

```
tsm collections on ./TP/NGSI-LD/CommonBehaviours
```

  - Switch OFF Test Collections

```
tsm collections off [collections]
```

5 Permutations

5.1 Introduction

Test cases are implemented using keyword or data driven approach, the major advantage of the second approach is that it makes the work easy for testing with different inputs and so adding new permutations (i.e. running the same Test Case with different test data and inputs and, sometimes, different expectations).

5.2 Data driven approach

Permutations of TCs with a data driven approach are the most readable and easier to change and extend. With this approach, the permutation is done by creating parameters for different scenarios, each scenario being a permutation.

For instance, the Test Purpose TP/NGSI-LD/CI/PROV/BE/003_01, tagged with be-create, is a multi-permutation TC, it contains a keyword for Batch Entity Creation that takes the input from the Test Cases permutation table.

Since 4 entries are present in the Test Cases permutation table, the TC will execute the following 4 permutations:

- 003_01_01 MinimalEntity: permutation to create a batch of minimal entities.
- 003_01_02 EntityWithSimpleProperties: permutation to create a batch of entities with simple properties.
- 003_01_03 EntityWithSimpleRelationships: permutation to create a batch of entities with simple relationships.
- 003_01_04 EntityWithRelationshipsProperties: permutation to create a batch of entities with relationships of properties.

In this case, adding a new permutation is done by adding a new line in the Test Cases table with the filename that contains entities payload for the new permutation.

5.3 Keyword approach

In some cases, the data-driven approach is not applicable, especially when it is not possible to reuse the same base Test Case code for each permutation. In these cases, permutations of TCs are expressed in different files, each file implementing one permutation.

For instance, the Test Purpose TP/NGSI-LD/CI/CONS/DISC/027_01, tagged with ed-attr, is an example of a TC with two permutations implemented in two separate files:

- 027_01.robot: permutation to retrieve detailed representation of an unknown NGSI-LD attribute that is expected to fail.
- 027_02.robot: permutation to retrieve detailed representation of an NGSI-LD attribute that is expected to succeed.

As already mentioned, each permutation is based on a custom keyword. To add a new permutation to such a TC, a new Test Case file has to be created along with a keyword that defines the TC workflow similar to what is done for other permutations but with the specifics of the new permutation.

6 Tags

6.1 Introduction

Tags allow to run groups of tests where each test contains the same selected tag.

There are two types of tags available: tags related to the resource and request being tested and tags that consist of a reference to the clause in the ETSI GS CIM 009 [i.1] that specifies the operation being tested.

6.2 Resource and request tags

This tag is comprised of an abbreviation, related to the resource the tag refers to, and the type of request being tested.

The abbreviations used are:

- **e:** Entity
- **be:** Batch Entity
- **ed:** Entity Discovery
- **te:** Temporal Entity
- **ea:** Entity Attributes
- **tea:** Temporal Entity Attributes

- **sub:** Subscription
- **csr:** Context Source Registration
- **csrsub:** Context Source Registration Subscription.
- **cb:** Common Behaviour Responses
- **ctx:** Storing, Managing and Serving @contexts

The available tags are:

- Subgroup 1.1:
  - e-create
  - e-delete
  - be-create
  - be-upsert
  - be-update
  - be-delete
  - te-create
  - te-update
  - te-delete
  - ea-append
  - ea-update
  - ea-partial-update
  - ea-delete
  - tea-append
  - tea-delete
  - tea-partial-update
  - tea-instance-delete
- Subgroup 1.2:
  - e-retrieve
  - e-query
  - te-retrieve
  - te-query
  - ed-types
  - ed-types-details
  - ed-type
  - ed-attrs
  - ed-attrs-details
  - ed-attr

- Subgroup 1.3:
  - sub-create
  - sub-update
  - sub-retrieve
  - sub-query
  - sub-delete
  - sub-notification
- Subgroup 2.1:
  - csr-create
  - csr-update
  - csr-delete
- Subgroup 2.2:
  - csr-retrieve
  - csr-query
- Subgroup 2.3:
  - csrsub-create
  - csrsub-update
  - csrsub-retrieve
  - csrsub-query
  - csrsub-delete
  - csrsub-notification
- Subgroup 3.1:
  - cb-ldcontext
  - cb-mergepatch
  - cb-get
  - cb-unsupport-medtype
  - cb-noacceptable-medtype
- Subgroup 4.1:
  - ctx-add
  - ctx-delete
- Subgroup 4.2:
  - ctx-list
  - ctx-serve

6.3 Section reference tags

Each Test Case also includes a reference to the clause in the ETSI GS CIM 009 [i.1] where it is defined. For instance, the Test Cases related to the creation of an entity contains a tag 5_6_1. Additionally, a new tag with the pattern `since_v1.x.y` is introduced in order to specify that a corresponding Test Case is created since the version of the specification `v1.x.y`.

7 Test Suite Maintenance

During the execution of the current activity, a test suite maintenance process has been developed to upgrade the robot framework and libraries together with the proper python version (currently working in python3.11) that are used by the Test Suite. This activity involves the migration of the following python packages:

- robotframework 3.2.2 → 6.1.1

  robotframework-jsonlibrary 0.3.1 → 0.5

  robotframework-requests==0.8.0 → 0.9.6

  deepdiff 5.2.1 → 6.7.1

  robotframework-httpctrl 0.1.6 → 0.3.1

  + prettydiff==0.1.0

  + robotframework-tidy 4.9.0

- The following libraries are no longer used:

  RESTinstance 1.0.2

  robotframework-jsonschemalibrary 1.0

  robotframework-metrics 3.2.0

Additionally, the resource files have been reorganized to be more aligned with the resources and tags described in the Test Suite Structure how it is described in clause 4.4 of ETSI GS CIM 012 [i.2]:

- **ContextInformationConsumption.resource**, contains all the operations (Keywords) related to the subgroup 1.2, Context Information Consumption.
- **ContextInformationProvision.resource**, contains all the operations (Keywords) related to the subgroup 1.1, Context Information Provision.
- **ContextInformationSubscription.resource**, contains all the operations (Keywords) related to the subgroup 1.3, Context Information Subscription.
- **ContextSourceDiscovery.resource**, contains all the operations (Keywords) related to the subgroup 2.2, Context Source Discovery.
- **ContextSourceRegistration.resource**, contains all the operations (Keywords) related to the subgroup 2.1, Context Source Registration.
- **ContextSourceRegistrationSubscription.resource**, contains all the operations (Keywords) related to the subgroup 2.3, Context Source Registration Subscription.
- **jsonldContext.resource**, contains all the operations (Keywords) related to the subgroups 4.1 and 4.2, Storing, Managing and Serving @contexts operations.
- **TemporalContextInformationConsumption.resource**, contains all the operations (Keywords) related to the subgroup 1.2.2, Temporal Entity Consumption.

- **TemporalContextInformationProvision.resource**, contains all the operations (Keywords) related to the subgroup 1.1.5, Temporal Entity Provision.

The group 3, Common Behaviour operations is not defined due to the operations used in this group are included into the previous list of resources.

8 Test Suite Usability

During the corresponding activity, the usability of the testing process has been increased. It is reflected in the content of clause 4. Additionally, some scripts under the `/scripts` folder make an analysis of the test data and resource files that there are in the system:

- **apiutils.py**, analyses of the resources files under the `/resources/ApiUtils` folder in order to extract the complete list of variables and keywords that are defined on those files.
- **find_tc_using_test_data.py**, facilitates the execution of all Test Cases in which use a specific file that has to be modified in the `/data` folder.
- **find_unused_test_data.py**, checks all the examples data contained in the `/data` folder and try to find if they are used or not in the corresponding robot files.

Additionally, a set of scripts have been generated to facilitate the installation of the Test Suite in different Operating System (e.g. Windows, Mac, Linux - Ubuntu systems):

- **configure.ps1**, configures all the python environment, installing the required components (e.g. git, python3.11, pip and virtualenv), clone the repository from https://forge.etsi.org/rep/cim/ngsi-ld-test-suite.git, create the corresponding python virtual environment, and install the python package required to execute the Test Cases for Windows systems.
- **configure.sh**, configures all the python environment, installing the required components (e.g. git, python3.11, pip and virtualenv), clone the repository from https://forge.etsi.org/rep/cim/ngsi-ld-test-suite.git, create the corresponding python virtual environment, and install the python package required to execute the Test Cases for MacOS and Ubuntu systems.
- **run_tests.ps1**, provides a set of commands examples to execute the Test Cases on Windows systems.
- **run_tests.sh**, provides a set of commands examples to execute the Test Cases on MacOS and Ubuntu systems.

9 Future work

9.1 Introduction

Clause 9 elaborates on some identified needs and ideas for future improvements in the NGSI-LD Test Suite.

It is divided in four sub-sections:

- Test Suite usability is focused on improvements related to the usability of the Test Suite for a better developer experience.
- Test Suite coverage of the specification is focused on specific areas of the specification which could be improved in the Test Suite or which are not yet implemented.
- Other Test Suite improvements is focused on other improvements that do not fall in the previous two subsections.
- Test Suite maintenance covers all operations related to keeping the Test Suite in line with newer releases of used libraries and to fix bugs that may be discovered.

9.2 Test Suite usability

9.2.1 Identification of failed checks when comparing results

When a check fails because an actual response does not match the expected response, the two responses are displayed, which allows the user to make a visual comparison.

However, the differences could be presented in a prettier way, for instance by displaying the two responses side-by-side and emphasizing the lines where there is a mismatch.

9.3 Test Suite coverage of the specification

Here are some first ideas for improvements in the coverage of ETSI GS CIM 009 [i.1]:

- Increase the number of permutations
- Check the response body is empty on queries returning a 204 status code
- Improve the coverage of the NGSI-LD Query Language
- Improve the coverage of the NGSI-LD Geo-query Language
- Add Test Cases addressing the count of the number of results
- Add Test Cases addressing multi-tenancy

In addition, as long as ETSI GS CIM 009 [i.1] evolves, the Test Suite has to be updated to cover the new features added to the specification, as well as any modification to the existing behaviours and endpoints.

Regarding the coverage for the new version of the specification, the following operations are to be developed:

- Version 1.6
  - Concise representation
  - deletedAt temporal property
  - Support for NGSI-LD Null
  - Merge Patch Behaviour
  - Replace Entity endpoint
  - Merge Entity endpoint
  - Consistency for Append Entity Attributes
  - Consistency for Update Entity Attributes
  - Replace Attribute endpoint
  - Batch Entity Merge endpoint
  - Support for notificationTrigger in subscriptions
  - Timeout and cooldown for Notification endpoint
  - Support for showChanges in notifications
- Version 1.7
  - Multi-attribute update for simplified representation
  - VocabularyProperty

  - Relaxing of tenant names
  - Updates for simplified representation in Merge Entity endpoint
- Any future version of the specification will be analysed to generate the corresponding new Test Cases.

Finally, the status of the execution of the Test Cases, including review and cleanup, will be reflected in the Test Requirements Summary spreadsheet.

9.4 Test Suite improvements

9.4.1 Severity of the errors

Currently, any failed check in a Test Case will mark the test as failed, without assigning any severity to the failure. With respect to the NGSI-LD specification, some failures could be considered as minor, while some others could be considered as critical. For instance, a typo in the title of an error can be considered as a minor error, while receiving an error response status upon the creation of an entity is a critical one.

For instance, here are some first ideas for the classification of the severity of an error:

- Critical Error: the broker crashes, the database has been altered when it should not have been (use GET to find out), the database has been altered in a way that was not expected (use GET to find out).
- Error: wrong HTTP status code.
- Minor: HTTP status code conveys a successful result code, but the payload data does not totally follow the specification.
- Not Implemented: the context broker does not implement the requested feature (it implies to define such an error code in the specification).

9.4.2 Optional and mandatory requests

Currently, the Test Suite does not distinguish between mandatory and optional requests. This is mainly because there is not really such a definition in the specification where features are mostly mandatory (except for a few ones).

If such a concept is introduced, it would then be possible to improve the test with a corresponding tag:

- mandatory: for mandatory requests
- optional: for optional requests

9.5 Test Suite maintenance

As with any software, the Test Suite has to be maintained. The issue of updating the Test Suite for other NGSI-LD API releases is not considered here.

That implies first to upgrade the frameworks and libraries that are used by the Test Suite. For instance, a major version of Robot Framework (release 7.0) has been released in January, 11th 2024 (https://github.com/robotframework/robotframework/blob/master/doc/releasenotes/rf-4.0.rst https://github.com/robotframework/robotframework/blob/master/doc/releasenotes/rf-7.0.rst). Migration to this new major release was tested without any problem in the execution of the Test Cases. Nevertheless, it has a huge impact in the automatic generation of the documentation, so migration to the new release was not done at this stage (https://github.com/MarketSquare/robotframework-requests/releases/tag/v0.9.0).

As more Test Cases are added, but also to benefit from new features added in the underlying frameworks and libraries, some code may have to be refactored or migrated. This is an important aspect of any sustainable software and it should not be neglected.

And finally, it covers the resolution of bugs that may be discovered in the Test Cases.

Annex A:
NGSI-LD Implementations

The Test Suite has been used on several open-source implementations of NGSI-LD context brokers: Orion-LD (https://github.com/Fiware/context.Orion-LD), Scorpio (https://github.com/ScorpioBroker/ScorpioBroker) and Stellio (https://github.com/stellio-hub/stellio-context-broker).

It has run successfully with each of the three context brokers and no problems were found during the setup and the execution of the Test Suite. The present document was designed to validate the Test Suite and does not imply anything concerning the completeness of the open-source implementations used. For that reason, the version numbers of the open-source implementations are not referenced here.

Of course, the Test Suite is designed to be run against any context broker implementing the NGSI-LD API specification and not specifically with one of the three context brokers mentioned above.

Annex B:
Change history

Date	Version	Information about changes
April, 14th 2021	V0.0.1	First draft of the document
May, 9th 2021	V1.0.0	Final draft for approval
May, 12th 2021	V1.0.1	Final draft approved
May, 12th 2021	V1.1.1	Technical Officer review for ETSI ***editHelp!*** submission for publication pre-processing
May, 30th 2023	V1.1.2	Early draft of the document corresponding to the TTF2 activity
September, 30th 2023	V1.2.1	Stable draft of the document corresponding to the TTF2 activity
March 2024	V1.3.2	Alignment with the repository
March 2024	V1.3.3	Final clean-up. Technical Officer review for ETSI ***editHelp!*** submission for publication pre-processing

History

Document history		
V1.1.1	June 2021	Publication
V2.1.1	April 2024	Publication