

ETSI GS CIM 014 V2.1.1 (2024-04)



Context Information Management (CIM); NGSI-LD Test Suite

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

RGS/CIM-0014v211

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

Important notice

The present document can be downloaded from:

<https://www.etsi.org/standards-search>

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status.

Information on the current status of this and other ETSI documents is available at

<https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx>

If you find errors in the present document, please send your comment to one of the following services:

<https://portal.etsi.org/People/CommitteeSupportStaff.aspx>

If you find a security vulnerability in the present document, please report it through our
Coordinated Vulnerability Disclosure Program:

<https://www.etsi.org/standards/coordinated-vulnerability-disclosure>

Notice of disclaimer & limitation of liability

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.

No recommendation as to products and services or vendors is made or should be implied.

No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.

In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.

The content of the PDF version shall not be modified without the written authorization of ETSI.

The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2024.
All rights reserved.

Contents

Intellectual Property Rights	4
Foreword.....	4
Modal verbs terminology.....	4
Executive summary	4
Introduction	4
1 Scope	6
2 References	6
2.1 Normative references	6
2.2 Informative references.....	6
3 Definition of terms, symbols and abbreviations.....	6
3.1 Terms.....	6
3.2 Symbols.....	7
3.3 Abbreviations	7
4 Abstract Test Method (ATM).....	7
4.1 Test Architecture	7
4.2 Test source code structure	8
4.2.1 Folder Structures.....	8
4.2.2 Data folder	8
4.2.3 TP folder.....	9
4.2.4 Libraries.....	9
4.2.5 Resources folder	9
4.2.6 Doc folder	10
4.3 Test interfaces	11
4.4 Test framework components	11
4.5 Test strategy	12
5 Untestable Test Purposes.....	12
Annex A (informative): Robot library modules.....	13
A.1 Robot code repository	13
Annex B (informative): Bibliography.....	14
Annex C (informative): Change history	15
History	16

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

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**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 is providing a detailed description of the implementation of the NGSI-LD Test Suite. It describes the architecture of the Test Suite, then goes into the details of the source code structure, the tests interfaces and the tests libraries.

Introduction

The ISG CIM group has defined an API for exchange of information contextualised 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) conformant 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 complete set of sample requests.

The present document describes the general architecture of the Test Suite, the overall organization of the source code, the tests interfaces, the tests framework components, then the tests strategies. It concludes with a list of identified untestable Tests Purposes.

1 Scope

The present document defines the organization or grouping of test cases based on the functionality to be tested (e.g. registration, subscription, query, etc.) and - most importantly - selects minimal subsets ("narrower scope") of functionality to permit testing of the main features of an operating NGSI-LD system.

2 References

2.1 Normative references

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

Referenced documents which are not found to be publicly available in the expected location might be found at <https://docbox.etsi.org/Reference>.

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 necessary for the application of the present document.

- [1] [ETSI GS CIM 009 \(V1.5.1\)](#): "Context Information Management (CIM); NGSI-LD API".
- [2] [ETSI GS CIM 013](#): "Context Information Management (CIM); NGSI-LD Test Purposes Descriptions".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-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] [ISG CIM repository at ETSI Forge](#): "Example code to implement NGSI-LD Test Purposes Descriptions".
- [i.2] [ETSI GR CIM 015](#): "Context Information Management (CIM); NGSI-LD Testing Environment Validation".

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

NOTE: The letters "NGSI-LD" were added to most terms to confirm that they are distinct from other terms of similar/same name in use in other organizations, however, in the present document the letters "NGSI-LD" are generally omitted for brevity.

NGSI-LD Context Registry: software functional element where Context Sources register the information that they can provide

NOTE: It is used by Distribution Brokers and Federation Brokers to find the appropriate Context Sources which can provide the information required for serving an NGSI-LD request.

NGSI-LD Context Source: source of context information which implements the NGSI-LD consumption and subscription (and possibly provision) interfaces

NOTE: It is usually registered with an NGSI-LD Registry so that it can announce what kind of information it can provide, when requested, to Context Consumers and Brokers.

3.2 Symbols

Void.

3.3 Abbreviations

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

ATM	Abstract Test Method
HTTP	Hyper Text Transfer Protocol
JSON	Javascript Object Notation
JSON-LD	JSON for Linked Data
MQTT	Message Queuing Telemetry Transport
NGSI-LD	Next Generation Service Interfaces - Linked Data
SUT	System Under Test
TC	Test Case
TSS	Test Suite Structure
URL	Uniform Resource Locator

4 Abstract Test Method (ATM)

4.1 Test Architecture

NGSI-LD test cases referenced in the present document relate to ETSI GS CIM 009 [1] and have been implemented using Robot Framework, a generic, application and technology independent framework.

The Robot Framework offers a highly modular architecture described in Figure 1:

- The test data is an easy-to-edit tabular format.
- The Robot Framework processes the test data, executes test cases and generates logs and reports. The core framework does not know anything about the target under test, and the interaction with it is handled by test libraries.
- The test libraries can either use application interfaces directly or use lower-level test tools as drivers.

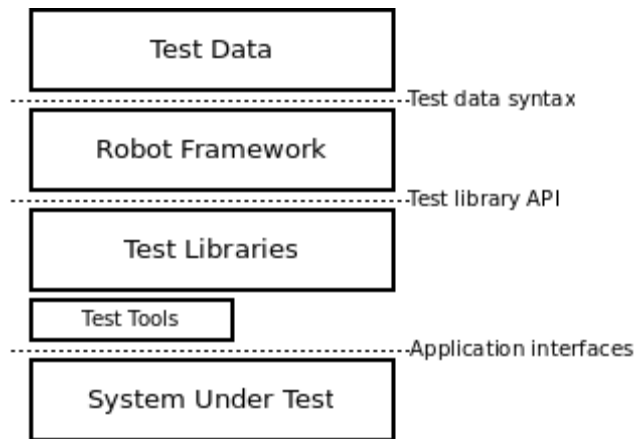


Figure 1: Robot Framework architecture

4.2 Test source code structure

4.2.1 Folder Structures

The overall code is organized following the Robot Framework standard of project structure:

- **data** folder contains test data samples
- **TP** folder contains test cases following the Test Suite Structure
- **libraries** folder contains custom developed Python functions that are made available as keywords
- **resources** folder contains custom keywords and variables declarations
- **doc** folder contains Python code to automatically generate the documentation information of the TCs for ETSI GS CIM 013 [2] and Unit Test to check that there is no change in the generated documentation.
- **scripts** folder contains a list of shell commands for launching the tests in different configurations
- **README.md** file contains instructions on the setup and configuration of the Test Suite
- **requirements.txt** contains the required Python libraries
- **LICENSE** contains the ETSI license of the corresponding Tests
- **pyproject.toml** contains the description of the project to be used with poetry tool
- **.pre-commit-config.yaml** contains the configuration of the pre-commit operation in order to use robotidy tool
- **.gitignore** contains the files and folder that have to be ignored by git tool

4.2.2 Data folder

Under **data** folder, tests data files are grouped in sub-folders such as **entities**, **temporalEntities**, **csourceRegistrations** and **csourceSubscriptions** according to the data type. These sub-folders may contain:

- Sample and/or template **test data**
- An **expectations** folder containing test data used for expectations in test cases
- A **fragments** folder containing fragment payloads used for test cases related to modifications of existing data

4.2.3 TP folder

Test cases are located under **TP** folder following the TSS, each test case implements one Test Purpose in a Robot file.

The TC filename is the sequence number of the Test Purpose; thus, the path of a TC is similar to the identifier of the Test Purpose.

For instance, the Test Purpose TP/NGSI-LD/CI/Prov/BE/003_01 is located at TP/NGSI-LD/ContextInformation/Provision/BatchEntities/CreateBatchOfEntities/003_01.robot.

In order to have a more readable structure, the Test Cases of a sub-sub group are grouped into sub-folders according to the endpoint concerned.

For instance, Test Cases of Create Batch of Entities are grouped into a folder named CreateBatchOfEntities.

4.2.4 Libraries

Under **libraries** folder, specific libraries are provided to help in the execution of the tests. It currently contains the following libraries:

- **assertionUtils.py**: contains the code to compare dictionaries ignoring some keys and provide a pretty difference result.
- **convertMD.py**: contains the code to transform the failed Test Cases information into Markdown format.
- **dateTimeUtils.py**: contains the code to check if a specific text is a date following a specific date-time format.
- **ErrorListener.py**: specific listener created to automatically generate the information of the failed Test Cases in text format.
- **githubIssue.py**: contains the code to optionally, automatically generate an issue with the corresponding information of the failed Test Cases in the GitHub repository of the tested Context Broker.
- **logUtils.py**: contains the code to provide a specific format representation of the execution of the Test Suite both in console and log file.

4.2.5 Resources folder

Common keywords are declared in files with the **.resource** extension, under **resources** folder. It currently contains the following utils:

- **ApiUtils** folder contains Keywords for NGSI-LD API calls such as sending HTTP requests to the SUT:
 - a) **ContextInformationConsumption.resource**: contains Keywords for NGSI-LD API related to the Context Information Consumption operations.
 - b) **ContextInformationProvision.resource**: contains Keywords for NGSI-LD API related to the Context Information Provision operations.
 - c) **ContextInformationSubscription.resource**: contains Keywords for NGSI-LD API related to the Context Information Subscription operations.
 - d) **ContextSourceDiscovery.resource**: contains Keywords for NGSI-LD API related to the Context Source Discovery operations.
 - e) **ContextSourceRegistration.resource**: contains Keywords for NGSI-LD API related to the Context Source Registration operations.
 - f) **ContextSourceRegistrationSubscription.resource**: contains Keywords for NGSI-LD API related to the Context Source Registration Subscription operations.
 - g) **jsonldContext.resource**: contains Keywords for NGSI-LD API related to the Storing, Managing and Serving @contexts operations.

- h) **TemporalContextInformationConsumption.resource:** contains Keywords for NGSI-LD API related to the Temporal Context Information Consumption operations.
- i) **TemporalContextInformationProvision.resource:** contains Keywords for NGSI-LD API related to the Temporal Context Information Provision operations.
- **jsonld-contexts** folder: contains the example contexts used in the Test Suite.
- **AssertionUtils.resource:** contains assertions Keywords for checking the expected behaviour of the SUT.
- **ContextServerUtils.resource:** contains Keywords used for mocking HTTP server when testing Storing, Managing and Serving @contexts features.
- **HttpUtils.resource:** contains Keywords used to retrieve data from HTTP responses.
- **JsonUtils.resource:** contains Keywords used for manipulating JSON data.
- **MockServerUtils.resource:** contains Keywords used for mocking HTTP server when testing Context Source Registration features.
- **NotificationUtils.resource:** contains Keywords used to interact with the component receiving notifications from the SUT.

Additionally, this folder includes the file **variables.py** which includes the configuration parameters to execute the Test Suite:

- **url:** contains the URL of the context broker which is to be tested (including the '/ngsi-ld/v1' path, e.g. `http://localhost:8080/ngsi-ld/v1`).
- **temporal_api_url:** contains the URL of the temporal API, in case a Context Broker splits this portion of the API on different service (e.g. `http://localhost:8080/ngsi-ld/v1`).
- **ngsild_test_suite_context:** contains the URL of the default JSON-LD context used in the requests (e.g. `'https://forge.etsi.org/rep/cim/ngsi-ld-test-suite/-/raw/develop/resources/jsonld-contexts/ngsi-ld-test-suite-compound.jsonld'`).
- **notification_server_host** and **notification_server_port:** contains the address and port used to create the local server that listen for notifications (the address has to be accessible by the context broker). Default value: `'0.0.0.0'` and `'8085'`.
- **context_source_host** and **context_source_port:** contains the address and port used for the context source operations (the address has to be accessible by the context broker). Default value: `'0.0.0.0'` and `'8086'`.
- **core_context:** the default cached core context used by the Brokers.
- **github_owner:** (optional) contains the GitHub user account to automatically publish the error message obtained in the execution of any Test Case into an issue in the Context Broker repository.
- **github_broker_repo:** (optional) contains the URL of the Context Broker repository where it is created the issues.
- **github_token:** (optional) contains the personal GitHub access token to create the issue (see GitHub documentation here: <https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens> to generate personal access tokens).

4.2.6 Doc folder

Under **doc** folder, there is the content to generate automatically the information used to create ETSI GS CIM 013 [2] and testing that there is no changes in the data (e.g. new Robot Test Suites or Test Cases or changes on them). It contains the following:

- **analysis:** contains the corresponding classes to generate the information in JSON format describing each of the Test Cases.

- **files:** contains the expected JSON files generated for each of the Test Cases in order to check any modification on the test.
- **results:** contains the corresponding generated JSON file for each of the Test Cases analysed.
- **tests:** contains the Unit Test files to check each of the Test Cases defined into the TP folder.
- **generateDocumentationData.py:** main file to execute the generation of the JSON files with the description of the Test Cases for each robot file.
- **statisticsDocumentationData.py:** allows the execution of the generation of JSON description files together with some statistical analysis of the Test Cases covered inside the results folder.
- **README.md:** contains a description about the use of the generation of JSON description files tool as well as the execution of the Unit Tests to check that there is no change or missing Test Cases to be added.

4.3 Test interfaces

Thanks to the Keywords defined (and presented in the previous clause), there is already a first level of abstraction between the Test Cases and the technical details of the API exposed by the SUT.

For instance, the creation of an entity is performed by calling the **Create Entity** Keyword defined in the `ApiUtils/ContextInformationProvision` resource file. All the technical details pertaining to the HTTP binding are then implemented by the Keyword, thus abstracting the Test Cases using this Keyword from the underlying API binding.

So, an alternative implementation of these Keywords could be later defined, using another API binding, and adapt the existing Test Cases so that it is possible to dynamically inject them one or another implementation of the suite of Keywords.

However, it is to be noted that the current implementation only provides a first general level of abstraction. Indeed, current Test Cases make use of some terms that are specific to the HTTP binding. For instance, a successful creation of an entity is checked by calling the following keyword: **Check Response Status Code Set To 201**. To achieve a totally abstracted implementation, a first step would then consist in abstracting away the existing Test Cases (in the example given just before, that would consist in abstracting the expected status code).

In conclusion, the currently developed Test Suite provides the necessary ground for a total abstraction over the API binding. However, it requires first some refactoring work on the existing Test Cases, but the foundations would still be the same and it would not require any destructive action of the existing implementation.

4.4 Test framework components

The base Robot Framework architecture described in clause 4.1 is applied in the majority of test scenarios that are based on simple interactions with the SUT such as create, update, get and delete. To perform the HTTP requests and validate the responses, the following libraries are used:

- **RequestsLibrary**
- **JSONLibrary**

In most complex test scenarios such as subscriptions, the trigger of notifications sent by a context broker has to be checked. For those cases, the following two libraries, that allow to run a **server** and a **client** for both **HTTP** and **MQTT** bindings that will be listening for notifications, are used:

- **HttpCtrl**
- **MqttLibrary** (not implemented yet)

4.5 Test strategy

Each Robot Test Suite file is composed of the following different sections:

- **Settings:**
 - a) **Resource:** where resources are imported (e.g. `ContextInformationConsumption.resource`).
 - b) **Suite Setup:** optional step to create initial status required for the overall Robot Test Suite.
 - c) **Suite Teardown:** optional step to delete the data created during the Robot Test Suite step.
 - d) **Test Setup:** optional step to create initial data required by the Test Case.
 - e) **Test Teardown:** optional step to delete data created during the Test Setup step.
 - f) **Test Template:** optional Keyword used when the Test Case is making use of permutations.
- **Variables:** where variables common to all Test Cases are defined.
- **Test Cases:** where each test case is defined. Each Test Case usually has:
 - **Documentation:** a brief description of the Test Case purpose.
 - **Tags:** make reference to the specific section in the NGSI-LD API specification document (e.g. 6_3_4) where the functionality being tested is defined and include the corresponding tag defined in clause 6 of ETSI GR CIM 015 [i.2] (e.g. sub-update).
 - **Steps:** the steps of the test where keywords are called from the imported resources to perform the test actions and assertions.
- **Keywords** (optional): contains implementation of keywords that are specific to the current Robot Test Suite and, thus, are not present in the imported resources files.

5 Untestable Test Purposes

Some Test Purposes are difficult, or impossible, to test:

- Notifications that never expire: it is possible to check it has not expired after a certain amount a time, but it is not possible to check it never expires.
- JSON-LD @context resolution when deleting a resource: JSON-LD @context has no real use when deleting a resource by identifier.
- Error types that either need to know a specific NGSI-LD implementation or to do a manual intervention on the running context broker instance:
 - `InternalError`.
 - `OperationNotSupported`.
 - `TooComplexQuery`.
 - `TooManyResults`.

Annex A (informative): Robot library modules

A.1 Robot code repository

The code written in the Python language for use in the Robot Framework, that implements the test description purposes for ETSI GS CIM 009 [1]: NGSI-LD API, is provided as an informative resource for users of the present document, at the following location [i.1] in ETSI forge repository: <https://forge.etsi.org/rep/cim/ngsi-ld-test-suite>.

Annex B (informative): Bibliography

- ETSI GR CIM 011: "Context Information Management (CIM); NGSI-LD Testing Framework: Test Purposes Description Language (TPDL)".
- ETSI GS CIM 016: "Context Information Management (CIM); NGSI-LD Testing Framework: Test Template".

Annex C (informative): Change history

Date	Version	Information about changes
December, 10 th 2020	V0.0.1	First draft of document
February, 4 th 2021	V1.0.1	Stable draft agreed by ISG-CIM
February, 16 th 2021	V1.1.0	Final draft for review by ISG-CIM
March, 15 th 2021	V1.1.1	Technical Officer review for EditHelp Publication pre-processing
May, 30 th 2023	V1.1.2	Early draft of the document corresponding to the TTF2 activity
September, 30 th 2023	V1.2.1	Stable draft of the document corresponding to the TTF2 activity
February 2024	V1.3.2	Alignment with repository structure
March 2024	V1.3.3	Final clean-up. Technical Officer review for EditHelp Publication pre-processing

History

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