ETSI GS CIM 014 V1.1.1 (2021-04)

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
DGS/CIM-0014v111

Keywords
API, IoT, 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 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://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/CommiteeSupportStaff.aspx

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is 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 IPR Policy, no investigation, 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.

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 organisation 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 (V1.3.1): "Context Information Management (CIM); NGSI-LD API".

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] Example code to implement NGSI-LD Test Purposes Descriptions.

NOTE: Available at https://forge.etsi.org/rep/cim/ngsi-ld-test-suite.

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

context registry: software functional element where Context Sources register the information that they can provide

context source: source of context information which implements the NGSI-LD consumption and subscription (and possibly provision) interfaces defined by the present document

entity: informational representative of something that is supposed to exist in the real world, physically or conceptually

3.2 Symbols

Void.

3.3 Abbreviations

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

ATM	Abstract Test Method
JSON	Javascript Object Notation
MQTT	Message Queuing Telemetry Transport
NGSI-LD	Next Generation Service Interfaces - Linked Data
SUT	System Under Test
TC	Test Case
TSS	Test Suite Structure

4 Abstract Test Method (ATM)

4.1 Test Architecture

NGSI-LD test cases referenced in the present document relate to ETSI GS CIM 009 (V1.3.1) [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.

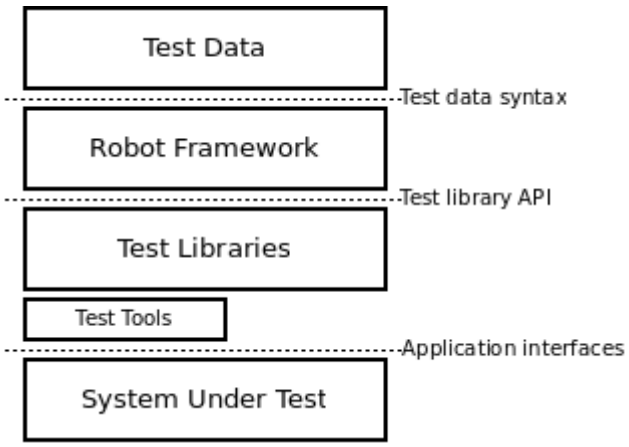

Figure 1: Robot Framework architecture

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
- **schemas** folder contains schemas for response schema validation
- **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

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
- Test data used for expectations in test cases
- 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 the endpoint concerned.

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

4.2.4 Resources folder

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

- **ApiUtils.resource**: contains Keywords for NGSI-LD API calls such as sending HTTP requests to the SUT
- **AssertionUtils.resource**: contains assertions Keywords for checking the expected behaviour of the SUT
- **HttpUtils.resource**: contains Keywords used to retrieve data from HTTP responses
- **JsonUtils.resource**: contains Keywords used for manipulating JSON data

- **NotificationUtils.resource**: contains Keywords used to interact with the component receiving notifications from the SUT

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** 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:

- **RESTinstance**
- **RequestsLibrary**
- **JSONSchemaLibrary**

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** file is composed of the following different sections:

- **Settings:** where resources are imported (e.g. **ApiUtils.resource**)
- **Variable:** 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
  - **Tag**: to indicate whether the test case is mandatory or not, so we are able to select test cases by their priority/importance

  - **Steps**: the steps of the test where keywords are called from the imported resources to perform the test actions and assertions
  - **Teardown**: to clean up the test environment after executing the test steps

- **Keywords** (optional): contains implementation of keywords that are specific to the current test scenario 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 (V1.3.1): NGSI-LD API [1], 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
Decembre,10th 2020	V0.0.1	First draft of document
February, 4th 2021	V1.0.1	Stable draft agreed by ISG-CIM
February, 16th 2021	V1.1.0	Final draft for review by ISG-CIM
March, 15th 2021	V1.1.1	Technical Officer review for EditHelp Publication pre-processing

History

Document history		
V1.1.1	April 2021	Publication