|  |
| --- |
| CHANGE REQUEST |
| Meeting ID:\* |  |
| Source:\* |  |
| Date:\* |  |
| Reason for Change/s:\* |  |
| CR against: Release\* |  |
| CR against: WI\* | [x]  Active <WI-0119> [ ]  MNT maintenance / < Work Item number(optional)>Is this a mirror CR? Yes [ ]  No [x] mirror CR number: [ ]  STE Small Technical Enhancements / < Work Item number (optional)>Only ONE of the above shall be ticked |
| CR against: TS/TR\* |  |
| Clauses \* |  |
| Type of change: \* | [ ]  Editorial change[x]  Bug Fix or Correction[ ]  Change to existing feature or functionality[ ]  New feature or functionalityOnly ONE of the above shall be ticked |
| Other TS/TR(s) impacted | None |
| Post Freeze checking:\* | This CR contains only essential changes and corrections? YES [x]  NO [ ] This CR may break backwards compatibility with the last approved version of the TS? YES [ ]  NO [ ]  |
| Template Version: January 2019 (do not modify) |

**oneM2M Notice**

The document to which this cover statement is attached is submitted to oneM2M. Participation in, or attendance at, any activity of oneM2M, constitutes acceptance of and agreement to be bound by terms of the Working Procedures and the Partnership Agreement, including the Intellectual Property Rights (IPR) Principles Governing oneM2M Work found in Annex 1 of the Partnership Agreement.

GUIDELINES for Change Requests:

Provide an informative introduction containing the problem(s) being solved, and a summary list of proposals.

Each CR should contain changes related to only one particular issue/problem.

In case of a correction, and the change apply to previous releases, a separate “mirror CR” should be posted at the same time of this CR

Mirror CR: applies only when the text, including clause numbering are exactly the same.

Companion CR: applies when the change means the same but the baselines differ in some way (e.g. clause number).

Follow the principle of completeness, where all changes related to the issue or problem within a deliverable are simultaneously proposed to be made E.g. A change impacting 5 tables should not only include a proposal to change only 3 tables. Includes any changes to references, definitions, and acronyms in the same deliverable.

Follow the drafting rules.

All pictures must be editable.

Check spelling and grammar to the extent practicable.

Use Change bars for modifications.

The change should include the current and surrounding clauses to clearly show where a change is located and to provide technical context of the proposed change. Additions of complete clauses need not show surrounding clauses as long as the proposed clause number clearly shows where the new clause is proposed to be located.

Multiple changes in a single CR shall be clearly separated by horizontal lines with embedded text such as, start of change 1, end of change 1, start of new clause, end of new clause.

When subsequent changes are made to content of a CR, then the accepted version should not show changes over changes. The accepted version of the CR should only show changes relative to the baseline approved text.

## Introduction

Provided input for Clause 5

<https://git.onem2m.org/specifications/tr-0076/-/merge_requests/2>

----------------------- Start of change 1 -----------------------

---a/TR-0076-Integrating\_NGSI-LD\_API\_in\_oneM2M.md
+++b/TR-0076-Integrating\_NGSI-LD\_API\_in\_oneM2M.md

@@ -77,6 +77,8 @@ The following referenced documents are not necessary for the application of the

- <a name="\_ref\_i.1">[i.1]</a> oneM2M Drafting Rules [https://member.onem2m.org/static\_Pages/others/Rules\_Pages/oneM2M-Drafting-Rules-V1%202%202.doc](https://member.onem2m.org/static\_Pages/others/Rules\_Pages/oneM2M-Drafting-Rules-V1%202%202.doc)

- <a name="\_ref\_i.2">[i.2]</a> ETSI GS CIM 009: "Context Information Management (CIM); NGSI-LD API" [https://www.etsi.org/deliver/etsi\_gs/CIM/001\_099/009/01.08.01\_60/gs\_CIM009v010801p.pdf](https://www.etsi.org/deliver/etsi\_gs/CIM/001\_099/009/01.08.01\_60/gs\_CIM009v010801p.pdf)

- <a name="\_ref\_i.3">[i.3]</a> ETSI GS CIM 006: "Context Information Management (CIM); Information Model" [https://www.etsi.org/deliver/etsi\_gs/CIM/001\_099/006/01.03.01\_60/gs\_CIM006v010301p.pdf](https://www.etsi.org/deliver/etsi\_gs/CIM/001\_099/006/01.03.01\_60/gs\_CIM006v010301p.pdf)

- <a name="\_ref\_i.4">[i.4]</a> JSON-LD 1.1 - A JSON-based Serialization for Linked Data", W3C Recommendation 16 July 2020, [https://www.w3.org/TR/json-ld11/](https://www.w3.org/TR/json-ld11/)

- <a name="\_ref\_i.5">[i.5]</a> Smart Data Models [https://smartdatamodels.org/](https://smartdatamodels.org/)

# 3 Definition of terms, symbols and abbreviations

<mark>Delete from the above heading the word(s) which is/are not applicable.</mark>

@@ -93,7 +95,6 @@ The following referenced documents are not necessary for the application of the

For the purposes of the present document, the [following] terms and definitions [given in ... and the following] apply:

<mark>Definition format</mark>

<mark>&lt;defined term>: &lt;definition></mark>

<mark>If a definition is taken from an external source, use the format below where [N] identifies the external document which must be listed in Section 2 References.</mark>

@@ -139,22 +140,306 @@ The key words "Shall", "Shall not", "May", "Need not", "Should", "Should not" in

## 5.1 Motivation and key concepts

A key motivation behind the NGSI-LD API<a href="#\_ref\_i.2">[i.2]</a> and the underlying NGSI-LD Information Model<a href="#\_ref\_i.3">[i.3]</a> is to make it easy for applications to get the information they need. To achieve this, applications can specify what information they want to have. This requires a common view of the world, which is encoded in the NGSI-LD Information Model. According to the NGSI-LD Information Model, the world consist of entities. There are entities of different types that have properties and relationships to other entities. The idea is to mimic a high-level human view of the world where objects are classified by assigning names to them and putting them into relation to each other. If talking about either a Property or a Relationship, the term Attribute can be used. Both the NGSI-LD API and the NGSI-LD Information Model are specified by ETSI ISG CIM as Group Specifications.

The NGSI-LD Information Model is a meta model. There are no restrictions on what entities exist and what properties and relationships they may have. This can be specified through compatible data models, which will be further explained in clause 5.2. The NGSI-LD API itself only relies on the NGSI-LD Informtion (meta) Model and not on the specific data models, i.e. it can handle entities speficied according to any compatible data model.

Known entities can be retrieved using an identifier, whereas entities can also be discovered and retrieved in a single step using queries. Queries can be geographically scoped, i.e. entities have to be in the specified area, and filtered according to propertoes or relationships, e.g. their value has to be larger than a certain value. Furthermore, applications can subscribe to be notified regarding changes to entities or simply periodically.

## 5.2 NGSI-LD Information Model

Figure 5.2-1 shows the NGSI-LD Information Model<a href="#\_ref\_i.3">[i.3]</a>. The key concept is the NGSI-LD Entity. An NGSI-LD Entity can represent an actual physical object, like a room or a table, or an abstract concept like a company. NGSI-LD Entities can have the following elements:

- NGSI-LD Entities have an identifier \_id\_, which is always a URI, following the linked data principles. "id" maps to "@id", which is defined by JSON-LD<a href="#\_ref\_i.4">[i.4]</a>. which is used for syntactically representing NGSI-LD information.

- NGSI-LD Entities have one or more \_type\_s. "type" maps to "@type", which is defined by JSON-LD<a href="#\_ref\_i.4">[i.4]</a>. which is used for syntactically representing NGSI-LD information.

- NGSI-LD Entities have zero or more Properties. A Property defines an aspect of an Entity. Each Property has a Value, which can have a simple datatype like a string or integer or be a complex JSON object.

- NGSI-LD Entities have zero or more Relationship. A Relationship points to another NGSI-LD Entity.

![NGSI-LD Information Model](media/NGSI-LD\_Information\_Model.png)

\*\*Figure 5.2-1: NGSI-LD Information Model\*\*

To enable meta data, both Properties and Relationships can themselves have Properties and Relationships, e.g. to encode a unit, an accuracy or the originator of the information, which may itself be modelled as an Entity.

Figure 5.2-2 shows a simple example of NGSI-LD Entity Instances. There are two cars modelled as Entities of type car. The car on the left has an "in front of" Relationship to the car on the right. The car on the right has a Property "speed", which in turn has the value "80", and the Property speed itself has another Property "source", which identifies the speedometer. If the speedometer had been modelled as an Entity, the "speed" Property would have a Relationship to the speedometer Entity instead.

![Simple NGSI-LD Entity Example](media/NGSI-LD\_Information\_Model\_Simple\_Example.png)

\*\*Figure 5.2-2: Simple NGSI-LD Entity Example\*\*

Figure 5.2-3 shows the sketch of an Entity graph. The Entities and the Relationships between Entities form a graph with the Entities as nodes and the Relationships as edges. Not all information is suitable to be directly represented in NGSI-LD, e.g. a video stream or a complex 3D model would not be suitable. In such cases, there can be Properties pointing to the respective information in external systems and meta information can be added that allows application to access this information.

![NGSI-LD Entity Graph Example Sketch](media/NGSI-LD\_Instance\_Graph\_Example.png)

\*\*Figure 5.2-3: NGSI-LD Entity Graph Example Sketch\*\*

Figure 5.2-4 shows a detailed Entity graph example. It shows that all Entities have a type and that both Relationships and Properties can again have Relationships and Properties providing meta information regarding the original Property or Relationship.

![NGSI-LD Detailed Entity Graph Example](media/NGSI-LD\_Conceptual\_Property\_Graph\_Example.png)

\*\*Figure 5.2-4: NGSI-LD Conceptual Property Graph Example\*\*

As the NGSI-LD Information Model is a meta model, it only defines what kind of elements exist, i.e. Entities, Properties, Relationships etc. but not what Entity types exist and what Relationships and Properties instances of such an Entity Type have, see Figure 5.2-5.

This information is specified by Data Models. To be used with NGSI-LD, they have to be compatible with the NGSI-LD Information Model, and specify what types of Entities exist and what Properties and Relationships instances of the respective Entity types can have. An example of a collection of such data models are the Smart Data Models<a href="#\_ref\_i.5">[i.5]</a>, which are supported by FIWARE, IUDX, OASC and tmforum. The specification of Data Models is considered out-of-scope of ETSI ISG CIM as it does not have the domain experts that would be required to create such models.

![NGSI-LD Compatible Data Models](media/NGSI-LD\_Compatible\_Data\_Models.png)

\*\*Figure 5.2-5: NGSI-LD Compatible Data Models\*\*

## 5.3 NGSI-LD API

### 5.3.1 Overview

<mark>The NGSI-LD resource structure will be introduced here or in an additional subclause. </mark>

Figure 5.3.1-1 shows the archtectural roles in an NGSI-LD system and the interactions between them. The NGSI-LD API provides support for all these roles and interactions.

### 5.3.2 Query operations

![NGSI-LD Roles and Interactions](media/NGSI-LD\_Roles.png)

\*\*Figure 5.3.1-1: NGSI-LD Architectural Roles and Interactions\*\*

The following architectural roles exist:

- The \*\*Context Broker\*\* typically has the key role in an NGSI-LD system and implements major parts of the NGSI-LD API. It can store information and transparently provides access to information stored elsewhere in case of a distributed deployment, in which case it interacts with the Context Registry.

- \*\*Context Consumers\*\* typically interact only with a single Context Broker, i.e. they only need to know its URL to request or subscribe for information.

- \*\*Context Producers\*\* produce information and the create, update and delete the resepective representation in the Context Broker.

- \*\*Context Sources\*\* store information themselves and make it accessible through requests and subscriptions. To enable Context Broker to find and access their information they register the information they have with the \*\*Context Registry\*\*.

- The \*\*Context Registry\*\* stores the registration of the Context Sources and, when requested, provides the list of Context Sources that may have relevant information for the given request.

The NGSI-LD specification consists of two parts. An abstract API is defined in clause 5 of the specification<a href="#\_ref\_i.3">[i.3]</a>, whereas a REST-style HTTP binding is defined in clause 6.

![Abstract NGSI-LD API](media/NGSI-LD\_Abstract\_API.png)

\*\*Figure 5.3.1-2: NGSI-LD Abstract API\*\*

All operations of the NGSI-LD Abstract API are shown in Figure 5.3.1-2, including the respective clauses in the NGSI-LD specfication<a href="#\_ref\_i.3">[i.3]</a>, in which they are defined.

![NGSI-LD Resource Structure](media/NGSI-LD\_Resource\_Structure.png)

\*\*Figure 5.3.1-3: NGSI-LD Resource Structure\*\*

The NGSI-LD resource structure of the HTTP Binding of NGSI-LD as defined in clause 6 of the NGSI-LD specification<a href="#\_ref\_i.3">[i.3]</a> is shown in Figure 5.3.1-3.

### 5.3.2 Retrieve and Query operations

This section shows a number of examples for retrieving and querying Entities using the NGSI-LD API<a href="#\_ref\_i.3">[i.3]</a>.

#### Retrieving an Entity

In the example the entity representing the person Sam is to be retrieved, see Figure 5.3.2-1.

![NGSI-LD API - Retrieve Entity](media/NGSI-LD\_API-Retrieve\_Entity.png)

\*\*Figure 5.3.2-1: NGSI-LD API - Retrieve Entity\*\*

What do applications need to know:

|Element | Value |

|---| --- |

|Base URL | http://localhost:9090/ngsi-ld/v1/entities/ |

|Entity Id | urn:ngsi-ld:Person:Sam |

|Data Model | location Property |

|Security credentials | [orthogonal aspect, not covered here] |

|Not needed | where actual information is stored |

Retrieve request

```

GET /ngsi-ld/v1/entities/urn:ngsi-ld:Person:Sam?attrs=location HTTP/1.1

Host: localhost:9090

Accept: application/ld+json

```

Response: (NGSI-LD Entity)

```

{

 "@context": [

 {

 "Person":"https://forge.etsi.org/gitlab/exampleOntology/Person",

 "location":"https://forge.etsi.org/gitlab/exampleOntology/location"

 },

 "http://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"

 ],

 "id": "urn:ngsi-ld:Person:Sam",

 "type": "Person",

 "location": {

 "type": "GeoProperty",

 "value": {

 "type": "Point",

 "coordinates": [-8.5, 41.2]

 }

 }

}

```

#### Querying Entities with Geographic Scope

In the example all cars within a given geographic scope are to be queried, see Figure 5.3.2-2.

![NGSI-LD API - Query Entities with Geographic Scope](media/NGSI-LD\_API-Query\_Entities\_Geographic\_Filter.png)

\*\*Figure 5.3.2-2: NGSI-LD API - Query Entities with Geographic Scope\*\*

What do applications need to know:

|Element | Value |

|---| --- |

|Base URL | http://localhost:9090/ngsi-ld/v1/entities/ |

|Data Model | car type |

|Geographic location | coordinates |

|Security credentials | [orthogonal aspect, not covered here] |

|Not needed | where actual information is stored |

Query request

```

GET /ngsi-ld/v1/entities?type=https://forge.etsi.org/gitlab/primerContext/StoreOntology/Car&geoproperty=location&georel=near;minDistance==1500&geometry=Point& coordinates=%5B57.4874120%2C20.2845608%5D&q=speed>50

HTTP/1.1

Host: localhost:9090

Accept: application/ld+json

```

Excerpt of result:

```

[

 {

 "id": "urn:ngsi-ld:Car:HDB1234",

 "type": âCar",

 "location {

 "type": "GeoProperty",

 "value": {

 "type": "Point",

 "coordinates": [57.48765, 20.284567]

 }

 }

...

```

### 5.3.3 Subscription/notification operations

In the given example, the subscriber wants to be notified whenever a car is detected in the specified geographic area, see Figure 5.3.3-1.

![NGSI-LD API - Subscribe to Entities with Geographic Scope](media/NGSI-LD\_API-Query\_Entities\_Geographic\_Filter.png)

\*\*Figure 5.3.3-1: NGSI-LD API - Subscribe to Entities with Geographic Scope\*\*

Here, two cases need to be monitored at the same time:

- new car added to the system with location in the area

- location of existing car has changed and is now within specified area.

What do applications need to know:

|Element | Value |

|---| --- |

|Base URL | http://localhost:9090/ngsi-ld/v1/entities/ |

|Data Model | car type, location Property |

|Security credentials | [orthogonal aspect, not covered here] |

|Own notification endpoint| http://localhost:9123 |

|Not needed | where actual information is stored |

Subscription

```

POST /ngsi-ld/v1/subscriptions HTTP/1.1

Host: localhost: 9090

Content-Type: application/json

{

 "id": "urn:ngsi-ld:Subscription:subscription123",

 "type": "Subscription",

 "entities": [

 {

 "type": "Car"

 }

 ],

 "geoQ": {"geoproperty":"location", "georel":"near;maxDistance==1500","geometry":"Point","coordinates":[57.48765,20.284567]},

 "notification": {

 "format": "normalized",

 "endpoint": {

 "uri": "http://localhost:9123",

 "accept": "application/json"

 }

 }

}

```

Example Notification:

```

{

 "id": "urn:ngsi-ld:Notification:515236541235",

Â  Â  "type": "Notification",

 "subscriptionId": "urn:ngsi-ld:Subscription:subscription123",

 "data": {

 "id": "urn:ngsi-ld:Car:Car12345",

 "type": "Car",

 "location": {

 "type": "GeoProperty",

 "value": {

 "type": "Point",

 "coordinates": [57.48765, 20.284567]

 },

 },

 "speed": {

 "type": "Property",

 "value": 35

 }

 }

}

```

### 5.3.4 Management operations

In the example the entity representing the person Sam is to be created, see Figure 5.3.4-1.

![NGSI-LD API - Create Entity](media/NGSI-LD\_API-Create\_Entity.png)

\*\*Figure 5.3.4-1: NGSI-LD API - Create Entity\*\*

What do applications need to know:

|Element | Value |

|---| --- |

|Base URL | http://localhost:9090/ngsi-ld/v1/entities/ |

|Data Model | person type |

|Entity | (Sam, see below) |

|Security credentials | [orthogonal aspect, not covered here] |

|Own notification endpoint| http://localhost:9123 |

|Not needed | where actual information is stored |

```

POST /ngsi-ld/v1/entities/

HTTP/1.1

Host: localhost:9090

Content-Type: application/ld+json

{

 "@context": [

 {

 "Person": "https://forge.etsi.org/gitlab/exampleOntology/Person",

 "location":"https://forge.etsi.org/gitlab/exampleOntology/location"

 },

 "http://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"

 ],

 "id": "urn:ngsi-ld:Person:Sam",

 "type": "Person",

 "location {

 "type": "GeoProperty",

 "value": {

 "type": "Point",

 "coordinates": [-8.5, 41.2]

 }

 }

}

```

## 5.4 Architectural considerations

In Figure 5.4.1, different supported deployment architectures for NGSI-LD systems are shown.

![NGSI-LD Architecture](media/NGSI-LD\_Architecture.png)

\*\*Figure 5.4-1: NGSI-LD Deployment Architectures\*\*

The central deployment has a central Context Broker storing all information in the system. Context Producers create, update and delete the information stored in the Context Broker. Context Consumers retrieve, query and subscribe to information stored in the Context Broker.

In the distributed deployment, there are Context Sources, possibly in addition to Context Producers. Context Sources store their own information and implement the NGSI-LD operations for retrieving, querying and subscribing to information. In order to be found by the Context Broker, the Context Sources register what kind of information they have with the Context Registry. On a request from a Context Consumer, the Context Broker checks the Context Registry for relevant Context Sources in addition to its own storage. It aggregates the infomration from the Context Sources and its own storage before returning it to the Context Consumer, i.e the distribution is transparent to the Context Consumer.

As these are architectural roles, an application can implement multiple roles at the same time, e.g. act as both a Context Consumer and a Context Producer.

Since Context Brokers also implement all operations of Context Sources, they can act as Context Sources themselves, and thus hierarchical architectures can be built as shown in the case of the Federated Deployment. However, the difference between distributed and federated deployments is more that in the case of a distributed deployment it is assumed that the whole deployment is set up and controlled by a single stakeholder, i.e. the distribution is intentional, whereas in a federated deployment, the assumption is that multiple stakeholder want to (partially) share their information.

As mentioned above, the underlying distribution is largely transparent to the Context Consumers, thus deployments can evolve from centralized to distributed or federated without having to change the Context Consumer.

# 6 Assessment of additional functionality brought by NGSI-LD

<mark>Based on the introduction in clause 5, description of the additional functionality that the integration of NGSI-LD API and its related functionality can bring to the oneM2M standard, including the resulting integrated use cases. </mark>

----------------------- End of change 1 -----------------------