Figure 1: oneM2M logo

oneM2M Technical Report	oneM2M Technical Report
Document Number	TR-0076-V-0.2.0
Document Name:	Integrating NGSI-LD API in oneM2M\
Date:	<20yy-mm-dd>
Abstract:	< An abstract of the document and information that may be used in subsequent electronic searches>
Template Version: January 2020 (do not modify)	

About oneM2M

The purpose and goal of oneM2M is to develop technical specifications which

address the need for a common M2M Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the myriad of devices in the field with M2M application servers worldwide.

More information about oneM2M may be found at: http//www.oneM2M.org

Contents

1 Scope

The present document discusses how key features of the NGSI-LD API can be integrated in oneM2M and studies the impacts and necessary changes to oneM2M Specifications in particular in regard to the following.

The present document - describes 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. - studies solutions for the architectural integration of NGSI-LD and its related functionalities into oneM2M, in particular with respect to oneM2M reference points and the existing oneM2M Common Service Functions. - studies the mapping of the information stored in oneM2M resources to the NGSI-LD information model. This includes, but is not limited to the current oneM2M semantic models (in particular SDT and the oneM2M base ontology, including SAREF integration) to the NGSI-LD information model, with the goal of making it available through an integration of NGSI-LD API and the Mca reference point. This may suggest changes to the current NGSI-LD and Mca, and the related information models. - studies the integration of NGSI-LD into oneM2M's management and security frameworks, in particular for registration, authentication, access control and device management.

2 References

2.1 Normative references

As a Technical Report (TR) is entirely informative it shall not list normative references.

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

The following referenced documents are necessary for the application of the present document.

Not applicable.

2.2 Informative references

Clause 2.2 shall only contain informative references which are cited in the document itself.

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

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] oneM2M Drafting Rules https://member.onem2m.org/static_Pages/others/Rules_Pages/oneM2M-Drafting-Rules-V1%202%202.doc
- [i.2] 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
- [i.3] 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
- [i.4] JSON-LD 1.1 - A JSON-based Serialization for Linked Data”, W3C Recommendation 16 July 2020, https://www.w3.org/TR/json-ld11/
- [i.5] Smart Data Models https://smartdatamodels.org/

3 Definition of terms, symbols and abbreviations

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

3.1 Terms

Clause numbering depends on applicability.

- A definition shall not take the form of, or contain, a requirement.
- The form of a definition shall be such that it can replace the term in context. Additional information shall be given only in the form of examples or notes (see below).
- The terms and definitions shall be presented in alphabetical order.

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

Definition format <defined term>: <definition>

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.

<defined term>[N]: <definition>

example 1: text used to clarify abstract rules by applying them literally

NOTE: This may contain additional information.

3.2 Symbols

Clause numbering depends on applicability.

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

Symbol format

```
&lt;symbol&gt;    &lt;Explanation&gt;
&lt;2nd symbol&gt;    &lt;2nd Explanation&gt;
&lt;3rd symbol&gt;    &lt;3rd Explanation&gt;
```

3.3 Abbreviations

Abbreviations should be ordered alphabetically.

Clause numbering depends on applicability.

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

Abbreviation format

```
&lt;ABBREVIATION1&gt;    &lt;Explanation&gt;
&lt;ABBREVIATION2&gt;    &lt;Explanation&gt;
&lt;ABBREVIATION3&gt;    &lt;Explanation&gt;
```

4 Conventions

The key words “Shall”, “Shall not”, “May”, “Need not”, “Should”, “Should not” in this document are to be interpreted as described in the oneM2M Drafting Rules [i.1]

5 Introduction to NGSI-LD API and NGSI-LD Information Model

5.1 Motivation and key concepts

A key motivation behind the NGSI-LD API[i.2] and the underlying NGSI-LD Information Model[i.3] 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[i.3]. 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[i.4]. 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[i.4]. 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.

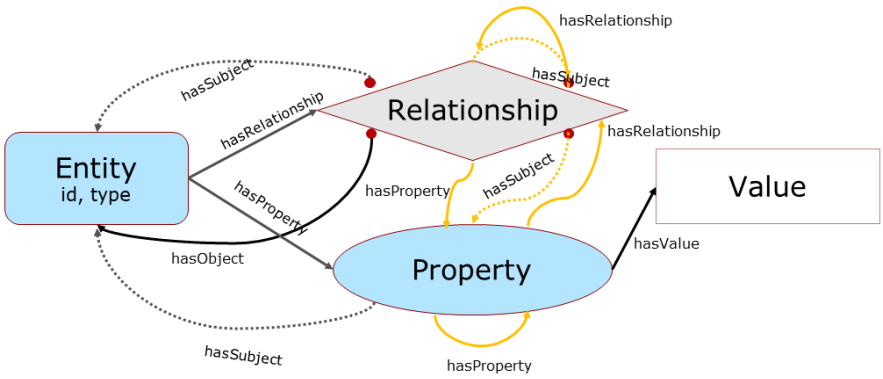

Figure 2: Figure 5.2-1: NGSI-LD Information Model

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.

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.

Figure 5.2-4 shows a detailed Entity graph example. It shows that all Entities

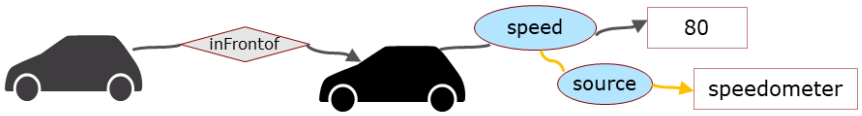

Figure 3: Figure 5.2-2: Simple NGSI-LD Entity Example

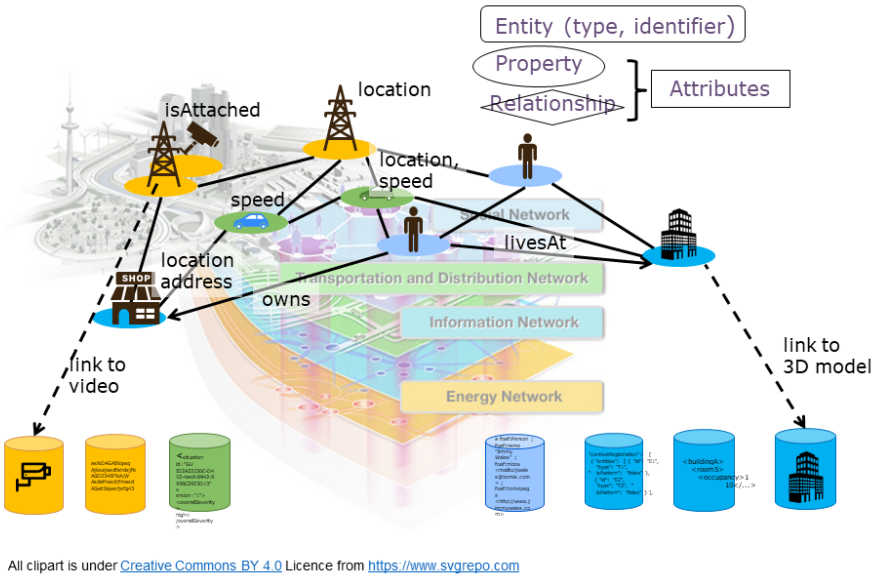

Figure 4: Figure 5.2-3: NGSI-LD Entity Graph Example Sketch

have a type and that both Relationships and Properties can again have Relationships and Properties providing meta information regarding the original Property or Relationship.

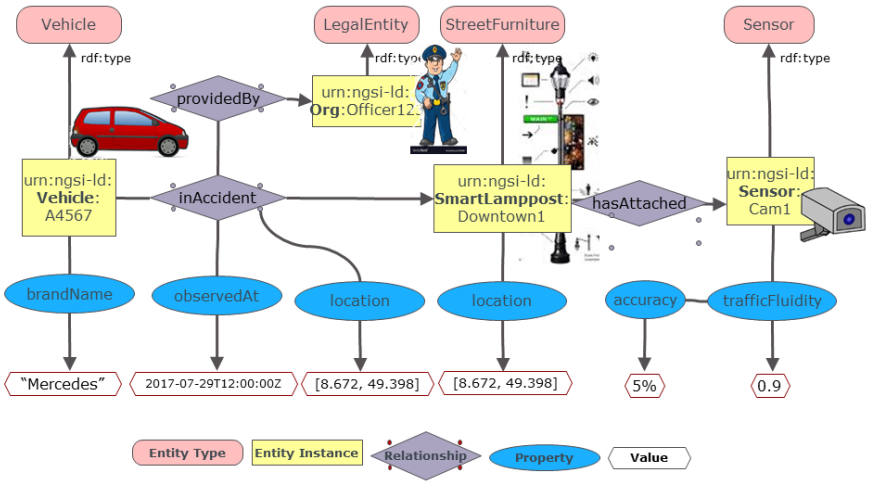

Figure 5: Figure 5.2-4: NGSI-LD Conceptual Property Graph Example

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[i.5], 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.

5.3 NGSI-LD API

5.3.1 Overview

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.

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

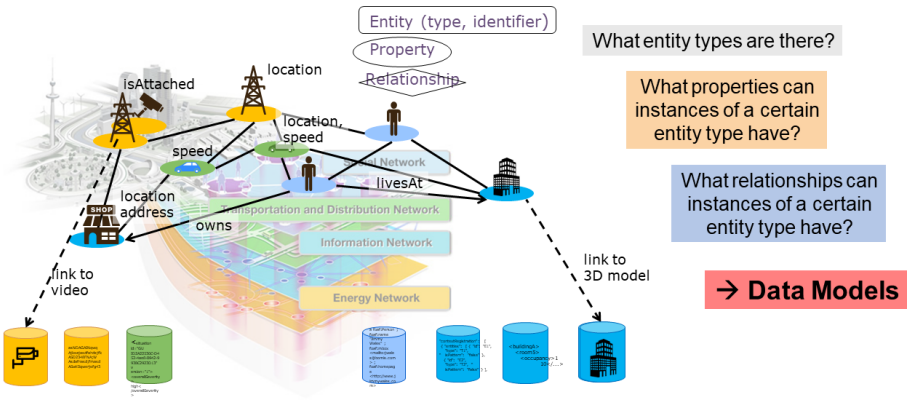

Figure 6: Figure 5.2-5: NGSI-LD Compatible Data Models

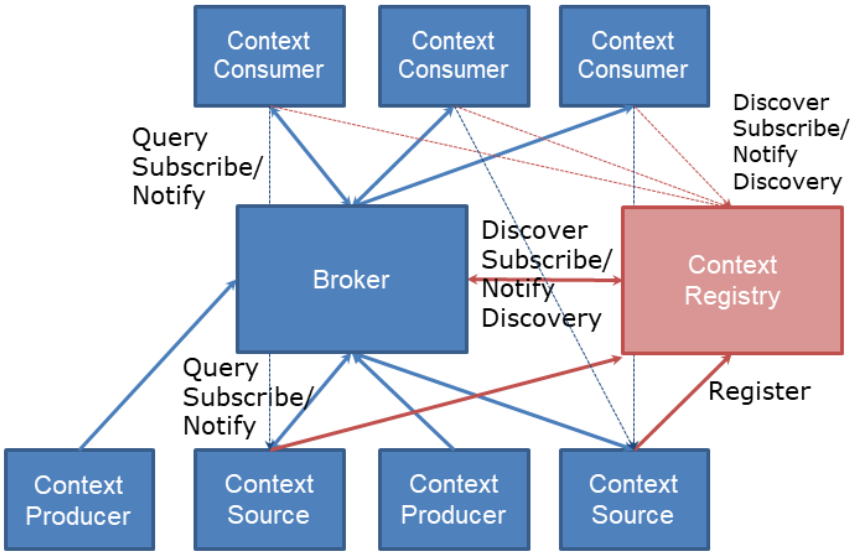

Figure 7: Figure 5.3.1-1: NGSI-LD Architectural Roles and Interactions

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 Brokers 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[i.3], whereas a REST-style HTTP binding is defined in clause 6.

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[i.3], in which they are defined.

The NGSI-LD resource structure of the HTTP Binding of NGSI-LD as defined in clause 6 of the NGSI-LD specification[i.3] 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[i.3].

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

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"
```

API	Functionality	Operations
Core API	Context Information Provision - operations for providing or managing Entities and Attributes	5.6.1 Create Entity 5.6.2 Update Entity Attributes 5.6.3 Append Entity Attributes 5.6.4 Partial Attribute Update 5.6.5 Delete Entity Attribute 5.6.6 Delete Entity 5.6.7 Batch Entity Creation 5.6.8 Batch Entity Upsert 5.6.9 Batch Entity Update 5.6.10 Batch Entity Delete 5.6.17 Merge Entity 5.6.18 Replace Entity 5.6.19 Attribute replace 5.6.20 Batch Entity Merge
	Context Information Consumption - operations for consuming Entities and checking for which Entity Types and Attributes Entities are available in the system	5.7.1 Retrieve Entity 5.7.2 Query Entities 5.7.5 Retrieve Available Entity Types 5.7.6 Retrieve Details of Available Entity Types 5.7.7 Retrieve Available Entity Type Information 5.7.8 Retrieve Available Attributes 5.7.9 Retrieve Details of Available Attributes 5.7.10 Retrieve Available Attribute Information
	Context Information Subscription - operations for subscribing to Entities, receiving notifications and managing subscriptions	5.8.1 Create Subscription 5.8.2 Update Subscription 5.8.3 Retrieve Subscription 5.8.4 Query Subscription 5.8.5 Delete Subscription 5.8.6 Notification
Temporal API	Temporal Context Information Provision - operations for providing or managing the Temporal Evolution of Entities and Attributes	5.6.11 Upsert Temporal Representation 5.6.12 Add Attributes to Temporal Representation 5.6.13 Delete Attributes from Temporal Representation 5.6.14 Partial Update Attribute instance 5.6.15 Delete Attribute Instance 5.6.16 Delete Temporal Representation
	Temporal Context Information Consumption - operations for consuming the Temporal Evolution of Entities	5.7.3 Retrieve Temporal Evolution of Entity 5.7.4 Query Temporal Evolution of Entities
Registry API	Context Source Registration - operations for registering Context Sources and managing Context Source Registrations (CSRs)	5.9.2 Register Context Source 5.9.3 Update CSR 5.9.4 Delete CSR
	Context Source Discovery - operations for retrieving and discovering CSRs	5.7.1 Retrieve CSR 5.7.2 Query CSRs
	Context Source Registration Subscription - operations for subscribing to CSRs, receiving notifications and managing CSRs	5.11.2 Create CSR Subscription 5.11.3 Update CSR Subscription 5.11.4 Retrieve CSR Subscription 5.11.5 Query CSR Subscription 5.11.6 Delete CSR Subscription 5.11.7 CSR Notification
JSON-LD Context API	Storing, managing and serving @contexts	5.13.2 Add @context 5.13.3 List @contexts 5.13.4 Serve @context 5.13.5 Delete and Reload @context

Figure 8: Figure 5.3.1-2: NGSI-LD Abstract API

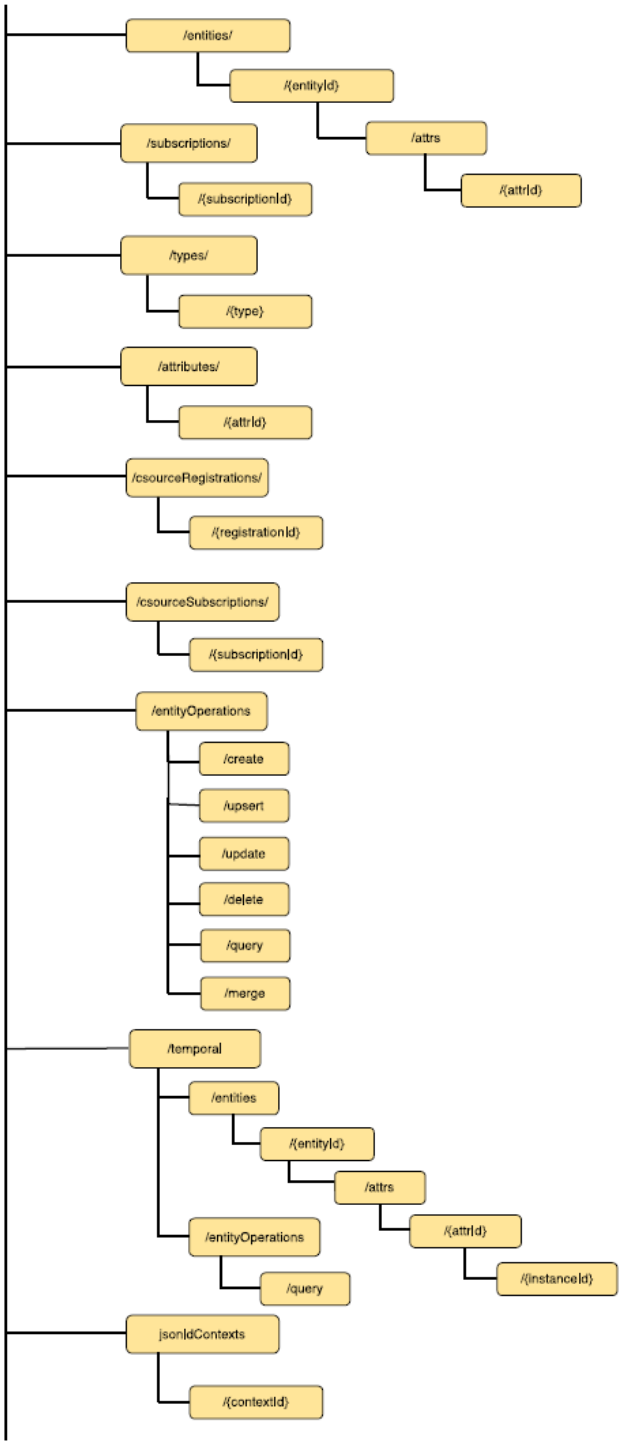

Figure 9: Figure 5.3.1-3: NGSI-LD Resource Structure

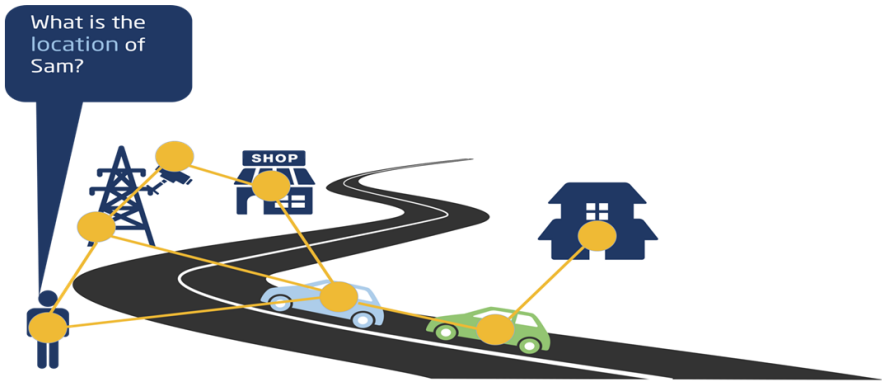

Figure 10: Figure 5.3.2-1: NGSI-LD API - Retrieve Entity

```
        },
        "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.

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&
HTTP/1.1
Host: localhost:9090
Accept: application/ld+json
```

Excerpt of result:

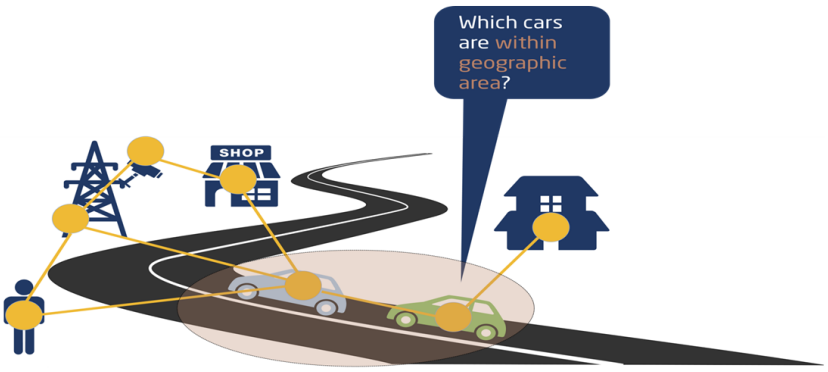

Figure 11: Figure 5.3.2-2: NGSI-LD API - Query Entities with Geographic Scope

```
[
    {
        "id": "urn:ngsi-ld:Car:HDB1234",
        "type": "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.

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
```

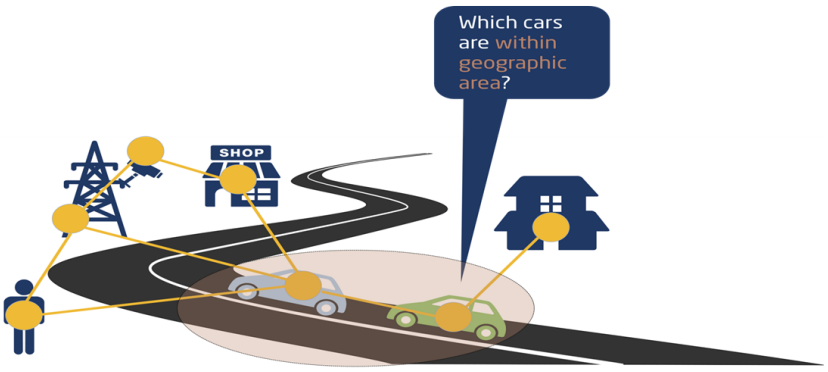

Figure 12: Figure 5.3.3-1: NGSI-LD API - Subscribe to Entities with Geographic Scope

```
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",
    "notification": {
        "format": "normalized",
        "endpoint": {
            "uri": "http://localhost:9123",
            "accept": "application/json"
        }
    }
}
```

Example Notification:

```
{
    "id": "urn:ngsi-ld:Notification:515236541235",
    "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.

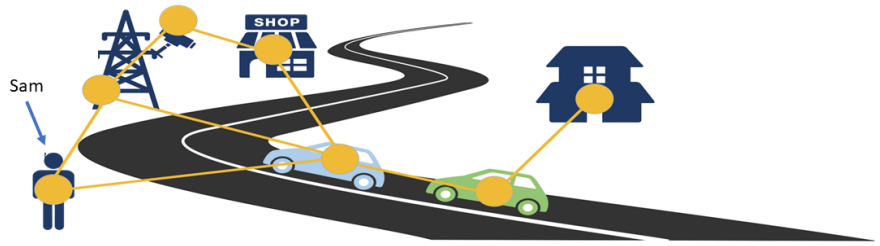

Figure 13: Figure 5.3.4-1: NGSI-LD API - Create Entity

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.

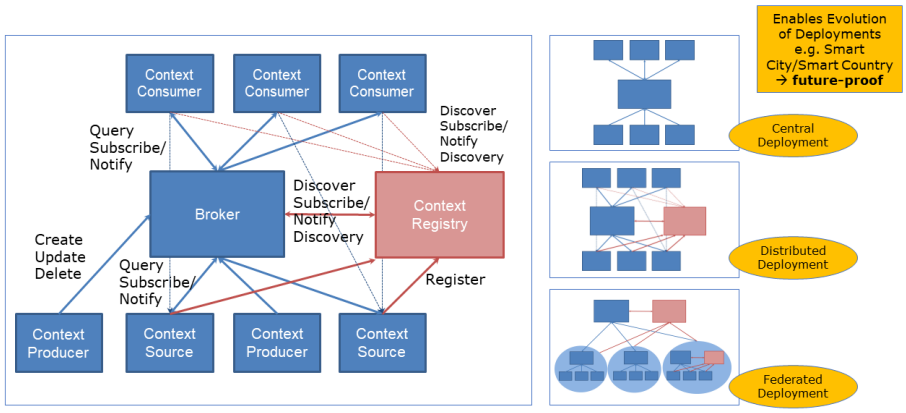

Figure 14: Figure 5.4-1: NGSI-LD Deployment Architectures

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

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.

7 Architectural integration of NGSI-LD into oneM2M

Study solutions for the architectural integration of NGSI-LD and its related functionalities into oneM2M, in particular with respect to oneM2M reference points and the existing oneM2M Common Service Functions.

8 Mapping between the information stored in oneM2M resources and the NGSI-LD information model

Study the mapping between the information stored in oneM2M resources and the NGSI-LD information model. This includes, but is not limited to the current

oneM2M semantic models (in particular SDT and the oneM2M base ontology, including SAREF integration) to the NGSI-LD information model, with the goal of making it available through an integration of NGSI-LD API and the Mca reference point. This may lead to an evolution of the current NGSI-LD and Mca, and the related information models.

9 Integration of NGSI-LD into oneM2M's management and security frameworks

Study the integration of NGSI-LD into oneM2M's management and security frameworks, in particular for registration, authentication, access control and device management.

10 Overall impact assessment and recommendations

Study the impacts and necessary changes to oneM2M Specifications

The following text is to be used when appropriate:

Proforma copyright release text block

This text box shall immediately follow after the heading of an element (i.e. clause or annex) containing a proforma or template which is intended to be copied by the user. Such an element shall always start on a new page.

Notwithstanding the provisions of the copyright clause related to the text of the present document, oneM2M grants that users of the present document may freely reproduce the <proformatype> proforma in this {clause|annex} so that it can be used for its intended purposes and may further publish the completed <proformatype>.

<PAGE BREAK>

Annexes

Each annex shall start on a new page (insert a page break between annexes A and B, annexes B and C, etc.).

Annex <A>:

Title of annex

<Text>

<PAGE BREAK>

Annex <B>:

Title of annex

<Text>

First clause of the annex

<Text>

B.1.1 First subdivided clause of the annex

<Text>

<PAGE BREAK>

Annex <y>:

Bibliography The annex entitled “Bibliography” is optional.

It shall contain a list of standards, books, articles, or other sources on a particular subject which are not mentioned in the document itself.

It shall not include references mentioned in the document.

- <Publication>: “<Title>”.

OR

<Publication>: “<Title>”.

<PAGE BREAK>

History

This clause shall be the last one in the document and list the main phases (all additional information will be removed at the publication stage).

Publication history	Publication history	Publication history
V1.1.1	<yyyy-mm-dd>	<Milestone>

Publication history	Publication history	Publication history

Draft history (to be removed on publication)	Draft history (to be removed on publication)	Draft history (to be removed on publication)
V0.1.0	2024-04-24	Includes the following contribution agreed during SDS64: SDS-2024-0045R01-TR-0076_Table_of_Contents
V0.2.0	2024-06-27	Includes the following contribution agreed during SDS65: SDS_2024_0045_TR_0076_Table_of_Contents