

oneM2M logo

| oneM2M Technical Report | oneM2M Technical Report |
| --- | --- |
| Document Number | TR-0076-V-0.5.0 |
| Document Name: | Integrating NGSI-LD API in oneM2M\ |
| Date: | 2025-02-13 |
| Abstract: | < An abstract of the document and information that may be used in subsequent electronic searches> |
| Template Version: January 2020 (do not modify) |  |

The present document is provided for future development work within oneM2M only. The Partners accept no liability for any use of this report.

The present document has not been subject to any approval process by the oneM2M Partners Type 1. Published oneM2M specifications and reports for implementation should be obtained via the oneM2M Partners’ Publications Offices.

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

Copyright Notification

1. 2025, oneM2M Partners Type 1 (ARIB, ATIS, CCSA, ETSI, TIA, TSDSI, TTA, TTC).

All rights reserved.

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

Notice of Disclaimer & Limitation of Liability

The information provided in this document is directed solely to professionals who have the appropriate degree of experience to understand and interpret its contents in accordance with generally accepted engineering or other professional standards and applicable regulations. No recommendation as to products or vendors is made or should be implied.

NO REPRESENTATION OR WARRANTY IS MADE THAT THE INFORMATION IS TECHNICALLY ACCURATE OR SUFFICIENT OR CONFORMS TO ANY STATUTE, GOVERNMENTAL RULE 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. NO oneM2M PARTNER TYPE 1 SHALL BE LIABLE, BEYOND THE AMOUNT OF ANY SUM RECEIVED IN PAYMENT BY THAT PARTNER FOR THIS DOCUMENT, WITH RESPECT TO ANY CLAIM, AND IN NO EVENT SHALL oneM2M BE LIABLE FOR LOST PROFITS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES. oneM2M EXPRESSLY ADVISES ANY AND ALL USE OF OR RELIANCE UPON THIS INFORMATION PROVIDED IN THIS DOCUMENT IS AT THE RISK OF THE USER.

# Contents

[1 Scope](#scope)  
[2 References](#references)  
    [2.1 Normative references](#normative-references)  
    [2.2 Informative references](#informative-references)  
[3 Definition of terms, symbols and abbreviations](#Xe254da9c2f303321d6c9a4dcbd53386c9eebb59)  
    [3.1 Terms](#terms)  
    [3.2 Symbols](#symbols)  
    [3.3 Abbreviations](#abbreviations)  
[4 Conventions](#conventions)  
[5 Introduction to NGSI-LD API and NGSI-LD Information Model](#X4a32ec1ebfd2655c793f487d87ca403ad5a7e39)  
    [5.1 Motivation and key concepts](#motivation-and-key-concepts)  
    [5.2 NGSI-LD Information Model](#ngsi-ld-information-model)  
    [5.3 NGSI-LD API](#ngsi-ld-api)  
        [5.3.1 Overview](#overview)  
        [5.3.2 Retrieve and Query operations](#retrieve-and-query-operations)  
            [Retrieving an Entity](#retrieving-an-entity)  
            [Querying Entities with Geographic Scope](#querying-entities-with-geographic-scope)  
        [5.3.3 Subscription/notification operations](#subscriptionnotification-operations)  
        [5.3.4 Management operations](#management-operations)  
    [5.4 Architectural considerations](#architectural-considerations)  
[6 Assessment of additional functionality brought by NGSI-LD](#X0edb8036f0104c50499b47ce3683c36586e9ed0)  
    [6.1 Introduction](#introduction)  
    [6.2 Use Case: Query parking sites in the vicinity (UC1)](#X1ae41adec5b3480f0618fe636380e0d2cb66e29)  
    [6.3 Use Case: Query parking sites in the vicinity (UC2)](#Xf747ff9c924d3adc568bbfad2bb2c849974ac9c)  
    [6.4 Use Case: Retrieve or query agriculture parcel(s) modelled as composite digital twins (UC3)](#Xbcf4f5eb1cd4835351e931038d27d3794f0e2b1)  
    [6.5 Use Case: Subscribe to be notified when temperature is above/below threshold in building (UC4)](#X745e9c2cdc8a1c01f9fc5c409b105d6e8e1d207)  
    [6.6 Summary of advantageous NGSI-LD functionalities](#X840fd440e555a60c6f5704d30c7004ee0360dcb)  
[7 Architectural integration of NGSI-LD into oneM2M](#Xce18da1a1bf1406cd66c6b3bc30a207e2536624)  
    [7.1 Introduction](#introduction-1)  
    [7.2 Mapping Approach - Making oneM2M Information available via NGSI-LD](#X50234233d68d3da985a09cf051747b93dc3a855)  
        [7.2.1 Motivation](#motivation)  
        [7.2.2 Sketch of Mapping Approach](#sketch-of-mapping-approach)  
        [7.2.3 Integration in oneM2M architecture](#integration-in-onem2m-architecture)  
[8 Mapping between the information stored in oneM2M resources and the NGSI-LD information model](#Xd391e463d9622006361fb6093334bbec11e51b8)  
[9 Integration of NGSI-LD into oneM2M’s management and security frameworks](#Xaf0afc5ba0a7aeafeb7b8e8b5b384ef879ebaa8)  
[10 Overall impact assessment and recommendations](#Xb4e7c7b93f26a6f42791abb00c215883de7c391)  
[Proforma copyright release text block]  
    [Annexes]  
[Annex <A>:Title of annex]  
[Annex <B>:Title of annex]  
    [ First clause of the annex]  
        [B.1.1 First subdivided clause of the annex](#Xe3370b47c86688ba84b7fc63c3002b536b46f91)  
[Annex <y>:Bibliography]  
[History](#history)

# 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> &lt;Explanation>  
&lt;2nd symbol> &lt;2nd Explanation>  
&lt;3rd symbol> &lt;3rd Explanation>

## 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> &lt;Explanation>  
&lt;ABBREVIATION2> &lt;Explanation>  
&lt;ABBREVIATION3> &lt;Explanation>

# 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.

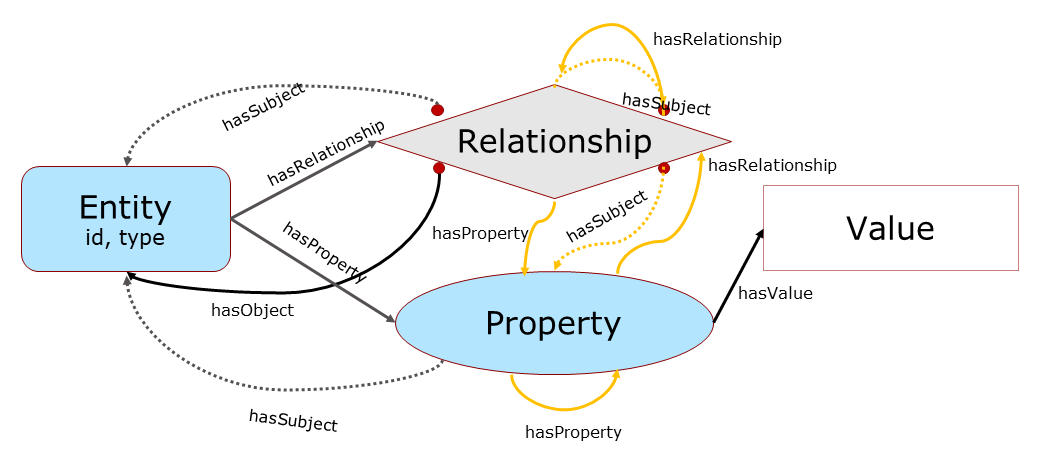


Figure 5.2-1: NGSI-LD Information Model

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.

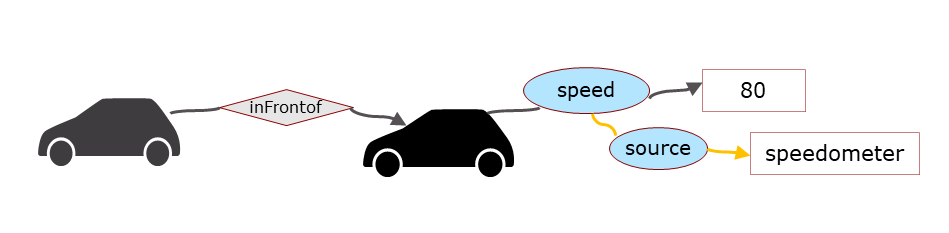


Figure 5.2-2: Simple NGSI-LD Entity Example

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.

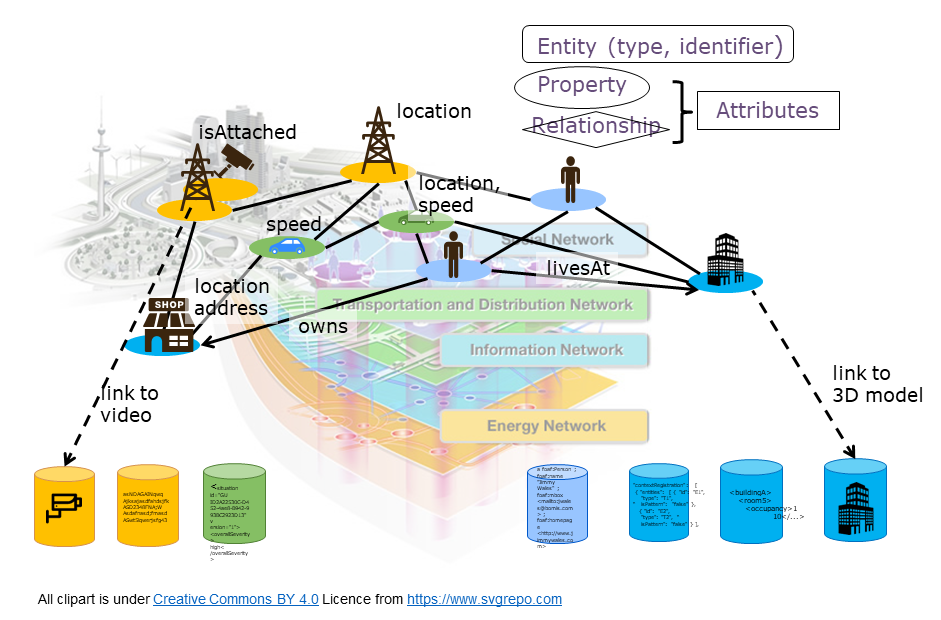


Figure 5.2-3: NGSI-LD Entity Graph Example Sketch

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.

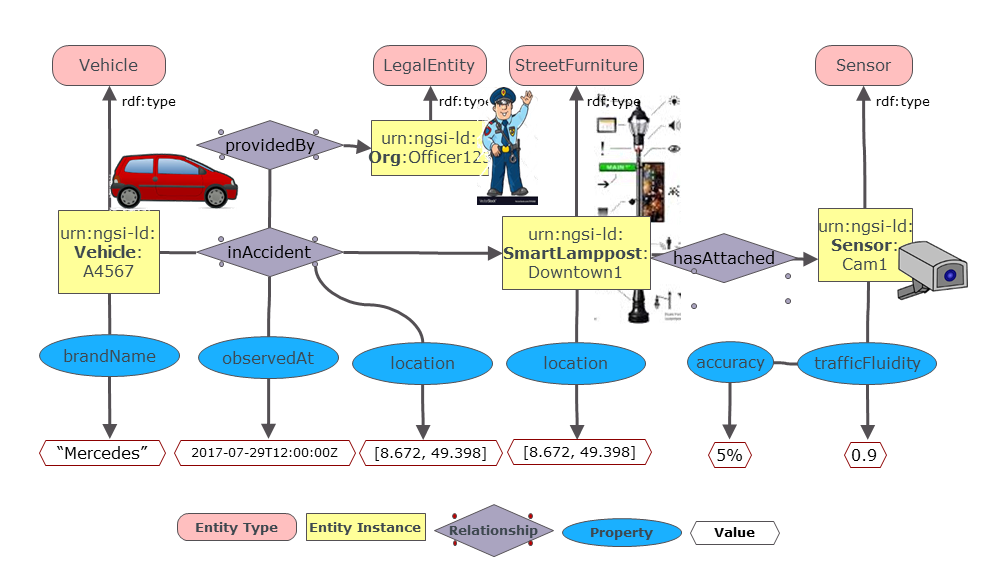


Figure 5.2-4: NGSI-LD Conceptual Property Graph Example

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.

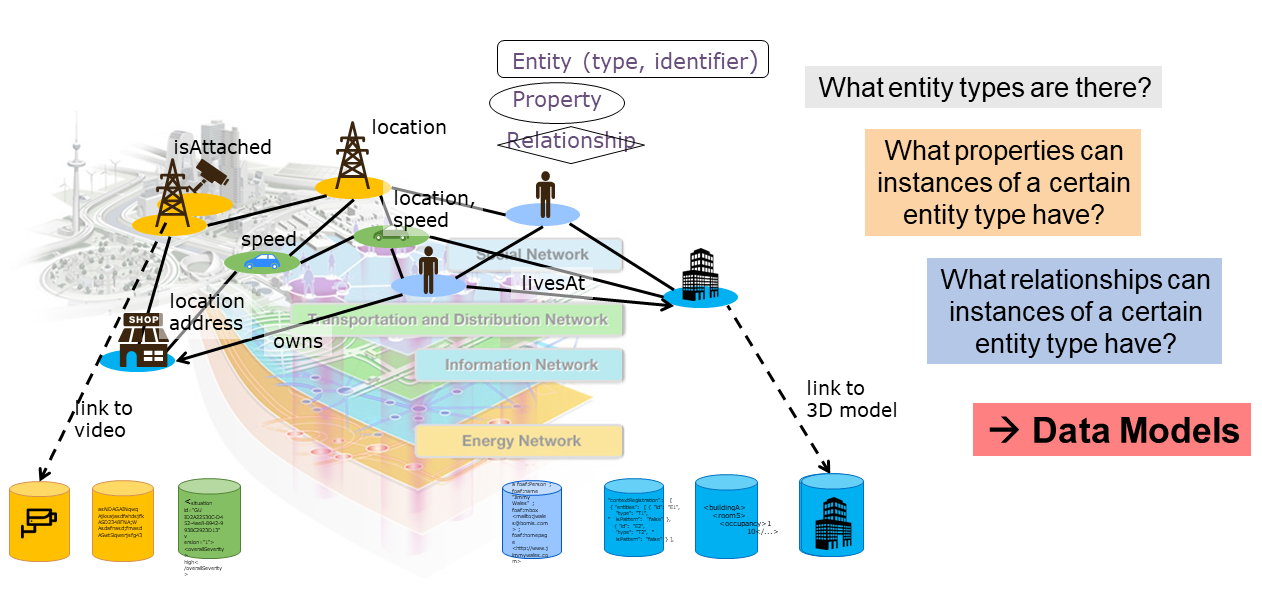


Figure 5.2-5: NGSI-LD Compatible Data Models

## 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.

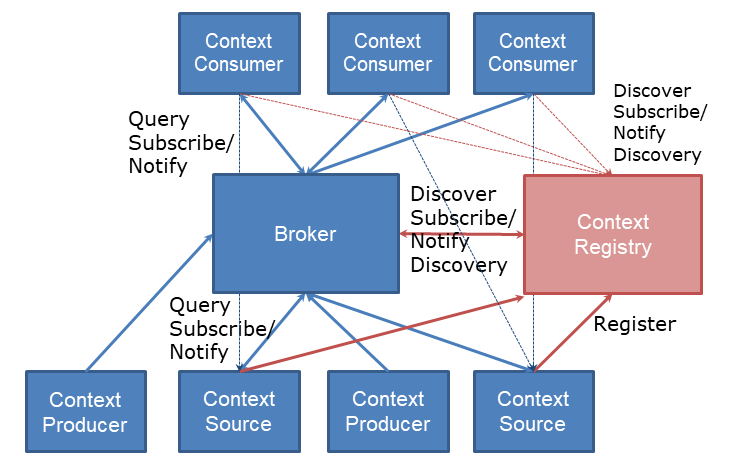


Figure 5.3.1-1: NGSI-LD Architectural Roles and Interactions

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 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.

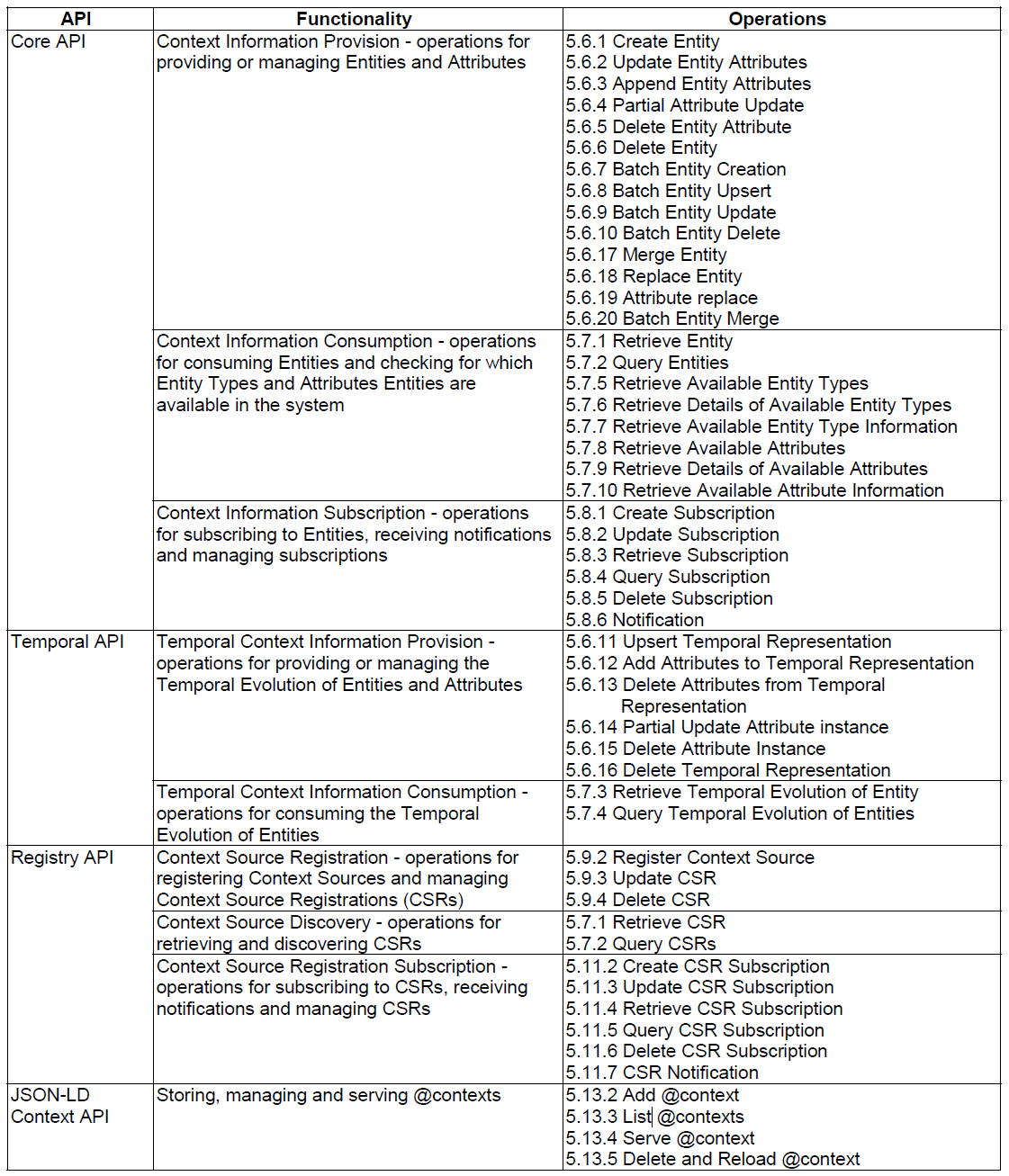


Figure 5.3.1-2: NGSI-LD Abstract API

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

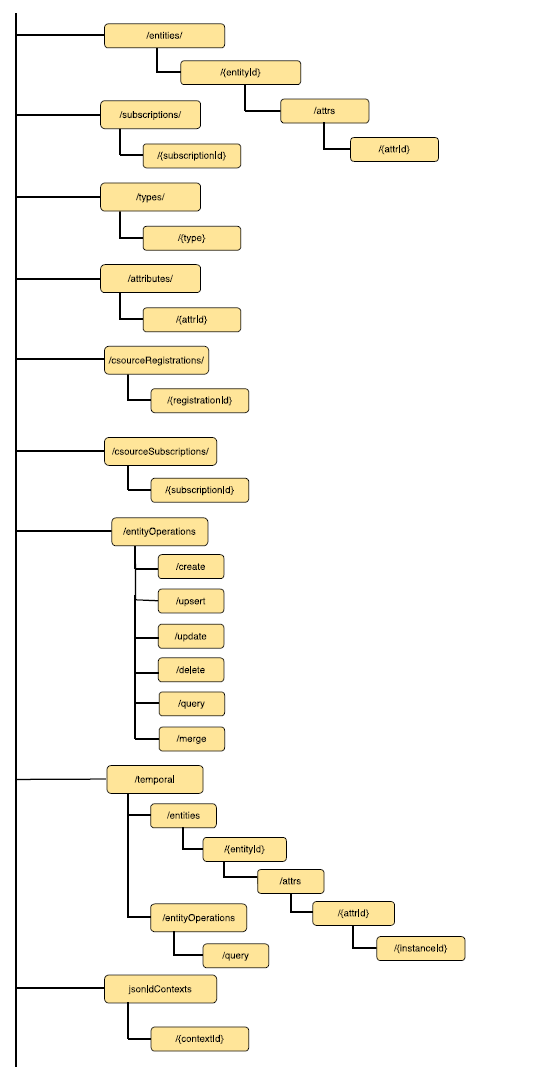


Figure 5.3.1-3: NGSI-LD Resource Structure

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

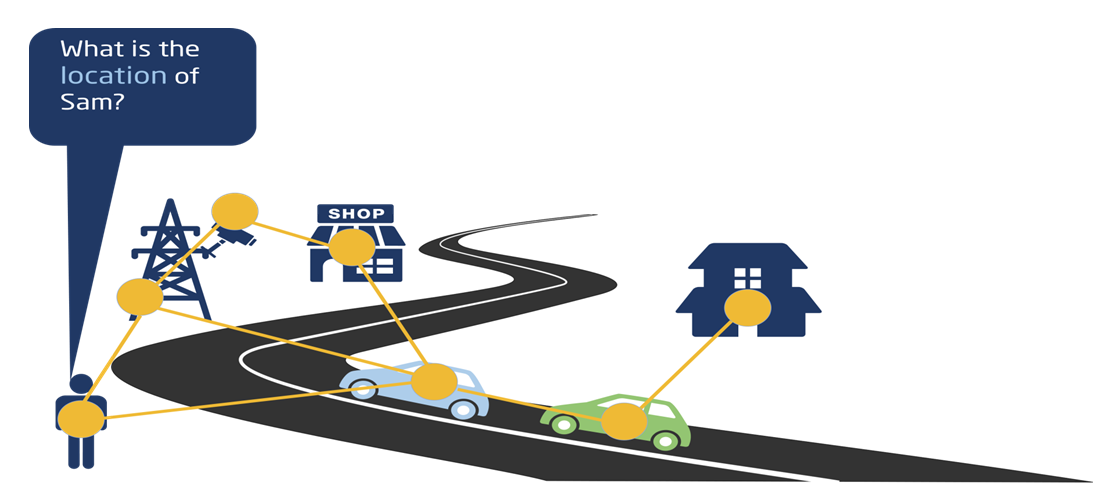


Figure 5.3.2-1: NGSI-LD API - Retrieve Entity

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.

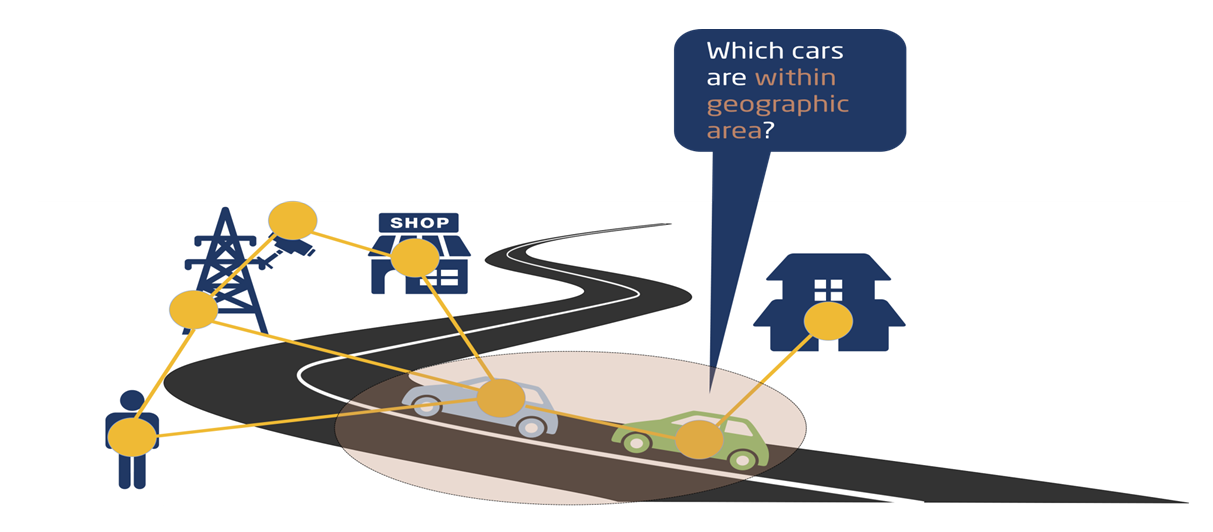


Figure 5.3.2-2: NGSI-LD API - Query Entities with Geographic Scope

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": “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.

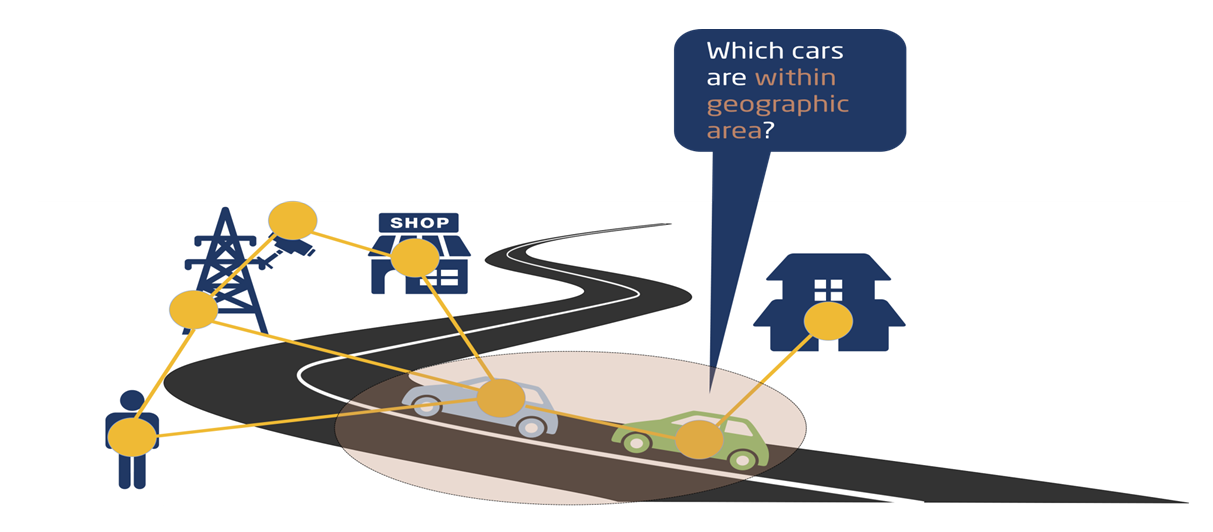


Figure 5.3.3-1: NGSI-LD API - Subscribe to Entities with Geographic Scope

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",  
    "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.

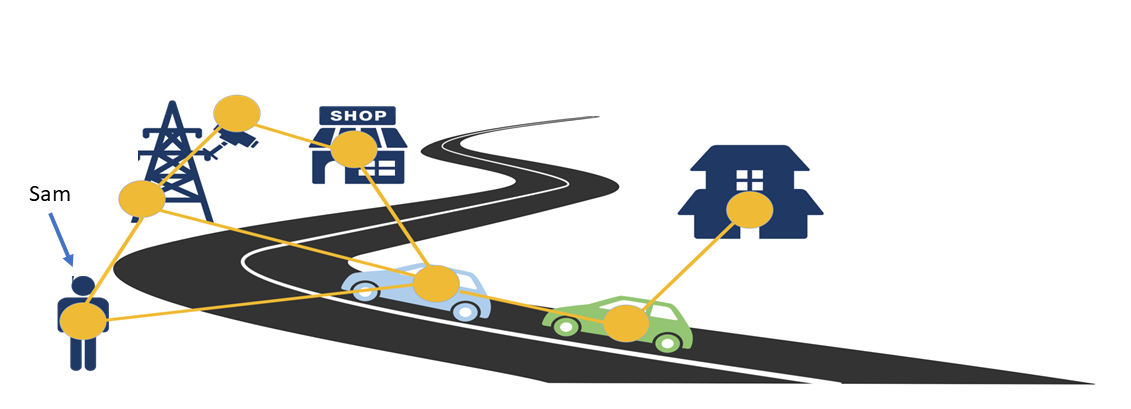


Figure 5.3.4-1: NGSI-LD API - Create Entity

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.

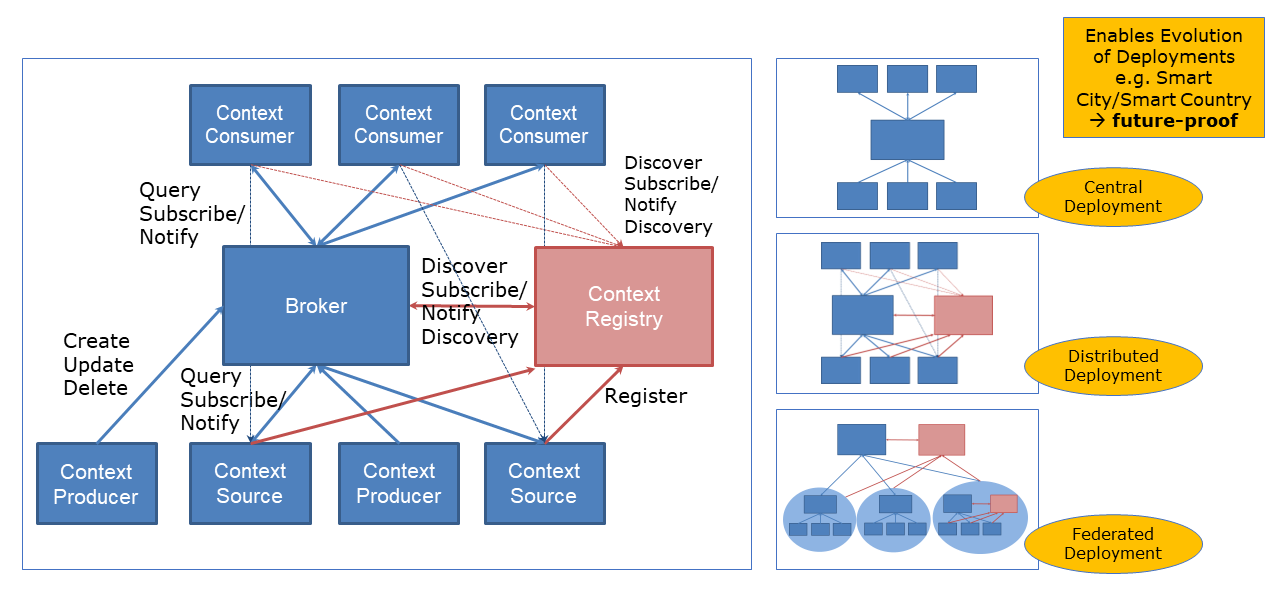


Figure 5.4-1: NGSI-LD Deployment Architectures

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

## 6.1 Introduction

Based on the general introduction of NGSI-LD in clause 5, clause 6 describes which advantages the integration of the NGSI-LD API into the oneM2M standard can bring to oneM2M applications. For this purpose, a number of use cases are analysed, highlighting the respective advantages for applications. Finally, the advantages are summarized with the goal of providing directions for further explorations of how the integration of NGSI-LD into oneM2M can be achieved, which is the focus of the following clauses.

## 6.2 Use Case: Query parking sites in the vicinity (UC1)

Use Case 1 (UC1) is about querying for parking sites within a given geographic area, e.g. around the current location of a car. Figure 6.2-1 visualizes this scenario.

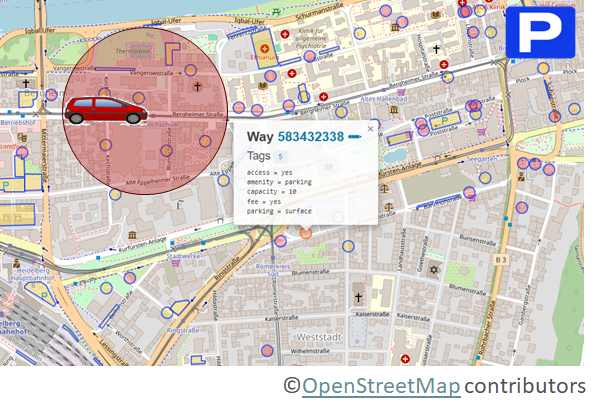


Figure 6.2-1: UC1 - Query parking sites in the vicinity

The basis for modelling the information required for this use case is the NGSI-LD compatible Smart Data Model[i.5] on Parking, which makes a distinction between OffStreetParking and OnStreetParking. As this distinction is not relevant for our use case, we restrict the example on a number of common NGSI-LD properties: - address - allowedVehicleType - availableSpotNumber - category - chargeType - location

The query for parking sites within a given geographic area, e.g. around the current location of a car:

GET /ngsi-ld/v1/entities/?type=OnStreetParking,OffStreetParking&geoproperty=location&georel=near%3BmaxDistance==1000&geometry=Point&coordinates[-3.8040616,43.4631649]  
  
Accept: application/json  
Host: localhost:9090  
Link: <http://example.org/myContext.jsonld>; rel="http://www.w3.org/ns/json-ld#context"; type="application/ld+json"

It queries for all NGSI-LD Entities of type OnStreetParking or OffStreetParking, and geographically scopes it with the circle around the geographic point with coordinates -3.8040616,43.4631649 and a radius of 1000m.

Excerpt of result:

[{  
 "id": "urn:ngsi-ld:OnStreetParking:santander:daoiz\_velarde\_1\_5",  
 "type": "OnStreetParking",  
 "allowedVehicleType": {  
 "type": "Property",  
 "value": [  
 "car"  
 ]  
 },  
 "availableSpotNumber": {  
 "type": "Property",  
 "value": 3,  
 "observedAt": "2018-09-12T12:00:00Z"  
 },  
 "category": {  
 "type": "Property",  
 "value": [  
 "blueZone",  
 "shortTerm",  
 "forDisabled"  
 ]  
 },  
 "chargeType": {  
 "type": "Property",  
 "value": [  
 "temporaryFee"  
 ]  
 ...

The advantages for the application are the following: - Application can reuqest information by specifying what it needs (“parking sites in the vicinity”) - Application gets result with a single request (unless paging is required due to number of results) - Application can gegographically scope the request

In order to do this, the application needs to know the following: - Data model: - type: OnStreetParking, OffStreetParking - GeoProperty name: location - its current location - Root URL: localhost:9090

## 6.3 Use Case: Query parking sites in the vicinity (UC2)

Use Case 2 (UC2) is about querying for a *free* parking spot within the proximity of a car. Figure 6.3-1 visualizes this scenario.

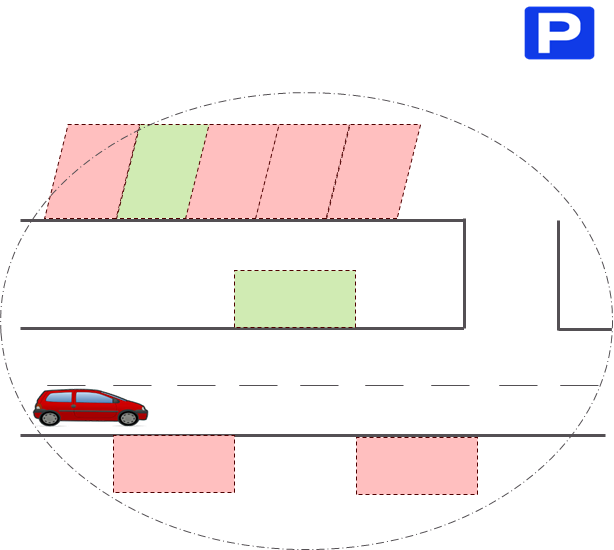


Figure 6.3-1: UC2 - Query for free parking spots in the proximity

The NGSI-LD Entity Type used is ParkingSpot, also from the NGSI-LD compatible Smart Data Model[i.5] on Parking. For this purpose, we are only interested in the following properties of each parking spot: - categrory - location - name - parkingSite (relation) - status

The query for parking spots within a given geographic area, e.g. within 100m around the current location of a car, filtered according to the status being *free* and restricting the result to the elments: id, type, category, location, name and status:

GET /ngsi-ld/v1/entities/?type=ParkingSpot&geoproperty=location&georel=near%3BmaxDistance==100&geometry=Point&coordinates=[-3.8040616,43.4631649]&q=status==“free”&pick=id,type,category,location,name,status  
  
Accept: application/json  
Host: localhost:9090  
Link: <http://example.org/myContext.jsonld>; rel="http://www.w3.org/ns/json-ld#context"; type="application/ld+json"

Excerpt of result:

[{  
 "id": "urn:ngsi-ld:ParkingSpot:santander:daoiz\_velarde\_1\_5:3",  
 "type": "ParkingSpot",  
 "category": {  
 "type": "Property",  
 "value": [  
 "onStreet"  
 ]  
 },  
 "location": {  
 "type": "GeoProperty",  
 "value": {  
 "type": "Point",  
 "coordinates": [  
 -3.80356167695194,  
 43.46296641666926  
 ]  
 }  
 },  
 "name": {  
 "type": "Property",  
 "value": "A-13"  
 "status": {  
 "type": "Property",  
 "value": "free",  
 "observedAt": "2018-09-21T12:00:00Z“  
 }  
},   
...

The advantages for the application are the following: - Application can reuqest information by specifying what it needs (“free parking spaces in the close vicinity”) - Application gets result with a single request (unless paging is required due to number of results) - Application can gegographically scope the request - Application can filter by status, i.e. only get “free” parking spaces - Application can project to only get the properties and relationships they are interested in

In order to do this, the application needs to know the following: - Data model: - type: ParkingSpot - GeoProperty name: location - Property names: category, location, name and status – and its possible values (“free”) - its current location - Root URL: localhost:9090

## 6.4 Use Case: Retrieve or query agriculture parcel(s) modelled as composite digital twins (UC3)

Use Case 3 (UC3) is retrieving (or querying for) agriculture parcel(s) modelled as composite digital twins. Figure 6.4-1 visualizes this scenario.

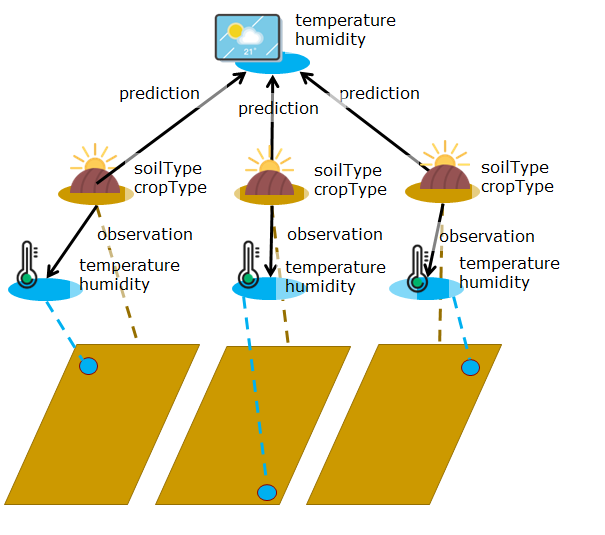


Figure 6.4-1: UC3 - Retrieve agricultural parcel

The NGSI-LD Entity Type used is AgriParcel, also from the NGSI-LD compatible Smart Data Model[i.5] on Agrifood. For this purpose, we are interested in the following properties and relationships of AgriParcel: - soil type - crop type - location - prediction (relationship to Weather Forecast) - observation (relationship to Weather Station)

Entities of type WeatherStation have at least the following properties: - temperature - humidity

Entities of type WeatherForecast have at least the following properties: - temperature - humidity - wind speed

The retrieve operation for a specific AgriParcel (AgriParcel:001) looks as follows:

GET /ngsi-ld/v1/entities/urn:ngsi-ld:AgriParcel:001? format=concise&join=inline&joinLevel=1  
Accept: application/json  
Host: localhost:9090  
Link: <http://example.org/myContext.jsonld>; rel="http://www.w3.org/ns/json-ld#context"; type="application/ld+json"

Introducing a *join* and *joinLevel* allows following the relationships and integrating the entities pointed to by the relation into the result, creating a hierarchy of entities.

Excerpt of result:

{  
 "id": "urn:ngsi-ld:AgriParcel:001",  
 "type": “AgriParcel",  
 "soilType": "Loamy",  
 "prediction": {  
 "object": "urn:ngsi-ld:WeatherForecast:XXX",  
 "objectType": “WeatherForecast",   
 "entity": {  
 "id": "urn:ngsi-ld:WeatherForecast:XXX ",  
 "type": “WeatherForecast",  
 "humidity": {"value": 98, "unitCode: "PCT"},  
 "temperature": {"value": 30, "unitCode: "CEL"},  
 "windSpeed": {"value": 3, "unitCode: "MPH"}  
 }},  
 “observation": {  
 "object": "urn:ngsi-ld:WeatherObserved:001",  
 "objectType": “WeatherObserved",  
 "entity": {  
 "id": "urn:ngsi-ld:WeatherObserved:001",  
 "type": “WeatherObserved",  
 ...  
}

If the specific AgriParcel is not a-priori known, a query for NGSI-LD Entities of type AgriParcel with a geographic scope can be used as in UC1 and UC2.

GET /ngsi-ld/v1/entities/?type=AgriParcel&geoproperty=location&georel=near%3BmaxDistance==1000&geometry=Point&coordinates=[57.5522023,-20.34840123]&format=concise&join=inline&joinLevel=1  
Accept: application/json  
Host: localhost:9090  
Link: <http://example.org/myContext.jsonld>; rel="http://www.w3.org/ns/json-ld#context"; type="application/ld+json"

The advantages for the application are the following: - Application can request information by specifying what it needs (“information about parcel(s) of land together with respective weather info”) - Application gets results with a single request (unless paging is required due to number of results) - Application can geographically scope the request (for the query) - Application can request that relationships are followed, and the related entities are embedded (up to a given depth) in the result

In order to do this, the application needs to know the following: - Data model: - type: AgriParcel - Relationships (worth following): observation, prediction - GeoProperty name: location - Root URL: localhost:9090

## 6.5 Use Case: Subscribe to be notified when temperature is above/below threshold in building (UC4)

Use Case 4 (UC4) is subscribing to changes in the indoor temperature of a building, filtering for an indoor temperature greater than 30°C, i.e. only once the threshold of 30°C has been crossed, notifications are going to be sent. Figure 6.5-1 visualizes this scenario.

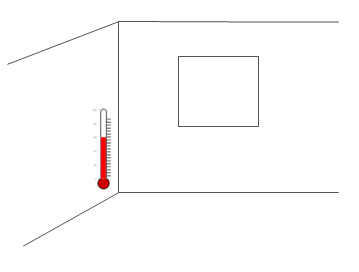


Figure 6.5-1: UC4 - Subscribe to be notified whenever temperature changes and is above threshold

The NGSI-LD Entity Type used is Building, also from the NGSI-LD compatible Smart Data Model[i.5] on Smart Cities. For this purpose, we are interested in the following properties of Building: - category - floorsAboveGround - floorsBelowGround - indoorTemperature - location - owner

The subscribe operation for the *indoor temperature* of the *Building* with the *id* “building-a85e3da145c1” and the filter on *indoorTemperature* greater that *30* looks as follows:

{  
    "id": "urn:ngsi-ld:Subscription:storeSubscription4",  
    "type": "Subscription",  
    "entities": [  
        {  
            "id": "urn:ngsi-ld:Building:building-a85e3da145c1",  
            "type": "Building"  
        }  
    ],  
    "watchedAttributes": ["indoorTemperature"],  
 "q": "indoorTemperature>30",  
    "notification": {  
        "format": "normalized",  
        "endpoint": {  
            "uri": "http://localhost:8080",  
            "accept": "application/json"  
        }  
    }  
}

Excerpt of resulting notification:

{  
 "id": "urn:ngsi-ld:Notification:515236543545",  
    "type": "Notification",  
 "subscriptionId": "urn:ngsi-ld:Subscription:storeSubscription4",   
 "data": {  
 "id": "urn:ngsi-ld:Building:building-a85e3da145c1",  
 "type": "Building",  
 "address": {  
 "type": "Property",  
 "value": {  
 "addressLocality": "London",  
 "postalCode": "EC4N 8AF",  
 "streetAddress": "25 Walbrook"  
 }  
 },  
 "category": {  
 "type": "Property",  
 "value": [  
 "office"  
 ]  
 },  
 “indoorTemperature": {  
 "type": "Property",  
 "value": 31.3,  
 }  
 ...  
 }  
}

The advantages for the application are the following: - Application can subscribe to be notified whenever the specified information changes(“indoor temperature of a building”) - Application gets notification on change - Application can request to only be notified in case a value is above a certain threshold (indoorTemperature>30)

In order to do this, the application needs to know the following: - Data model: - type: Building - Property: indoorTemperature (and its value, e.g. that Celsius scale is used) - Root URL: localhost:9090

## 6.6 Summary of advantageous NGSI-LD functionalities

The following bullet points provide a summary of the NGSI-LD functionalities that are advantageous for certain applications as identified in the use cases presented above.

* Applications can request information by specifying what they need. This is especially important for applications that on the one hand want to (re-)use information, but on the other hand do not want to deal with
  + first finding out where information is stored in the system
  + understanding different resource structures
  + handling various native data representations
* Applications can geographically scope queries
* Applications can filter results based on property values / relationship targets
* Application can project to only get the properties and relationships they are interested in
* Applications can get back whole entity graphs by following relationships. This can be especially relevant for (composite) Digital Twins
* With queries, applications can get back result with a single request, i.e. there is no separate discovery step followed by retrieval(s).
* Applications can subscribe to be notified of relevant changes in information (or periodically, for which no example has been shown)

# 7 Architectural integration of NGSI-LD into oneM2M

## 7.1 Introduction

Clause 7 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.

## 7.2 Mapping Approach - Making oneM2M Information available via NGSI-LD

### 7.2.1 Motivation

oneM2M resource structures already contain relevant information, however, applications often have to first discover and then individually retrieve this information to check whether it is actually relevant for them. The goal of this approach is to make the information stored in existing oneM2M resources also available to applications via the NGSI-LD API, enabling application to specify and efficiently retrieve the information they actually need.

In particular, applications can request the information by specifying the following: - Applications can filter and scope based on the information itself (as opposed to meta information attached to resources). Specific index structures can be used to make access efficient due to the common meta information model. - Application can project to only get the properties and relationships they are interested in (as opposed to getting the content on resource granularity) - Applications can get back whole entity graphs by following relationships (e.g. across multiple resources)

Applications have the following interaction options using NGSI-LD: - Retrieve specific information they can already identify, i.e. as an NGSI-LD Entity with a given identifier - Query relevant information as described above and getting back all results, in form of NGSI-LD Entities, with a single request. There is no separate discovery step followed by retrieval(s). - Subscribe to be notified of relevant changes in information, or periodically.

### 7.2.2 Sketch of Mapping Approach

The idea of the approach is to define a general mechanism based on a mapping language. With this approach, users can define a mapping, e.g., how a value that can be extracted from a oneM2M resource is the value of an NGSI-LD property, belonging to a specifc NGSI-LD Entity with a certain NGSI-LD type. These user-provided mappings could be stored in <semanticDescriptor> resources. Alternatively, a specific mapping resource type could be defined.

Figure 7.2.2-1 shows two oneM2M resource structures, e.g. two <container> resources with <contentInstance> resources and one <semanticDescriptor> resource each. The <contentInstance> resources encapsulate information as provided by the source, e.g. a device or IPE. The resources contain the mapping that describes how the value can be extracted, the value of which NGSI-LD property it represents, to which NGSI-LD Entity the property belongs and what type the Entity has. The resulting information is depicted on right side of Figure 7.2.2-1 and this information can be accessed through the NGSI-LD API.

The extraction of values is not limited to <contentInstance> resources, but includes all resource types that can contain information, e.g. <flexContainer> or even <semanticDescriptor>.

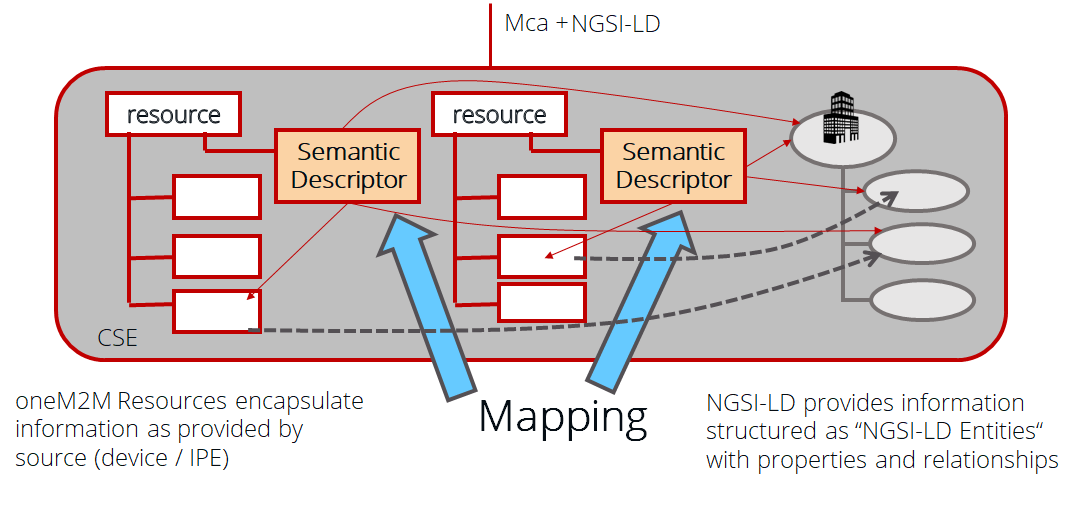


Figure 7.2.2-1: Core Mca and NGSI-LD provide complementary functionality

Figure 7.2.2-2 shows an example. The content instances contain values in different formats. On the left side, the content instances under Resource A contain integers, in the middle, under Resource B, the content instances each contain an XML structure. The semantic descriptor of Resource A contains mapping information, i.e. that the information is about an Entity with the identifer Room123, which is of type Room, has a property called indoorTemperature, whose value is to be extracted from the resource, and that has a unit as meta information that indicates that the temperature is given in Celsius. The mapping information in the semantic descriptor of Resource B indicates that the information belongs to the same entity, but in this case the property is called occupancy and the information has to be extracted from the XML. The extraction examples only serve illustration purposes here, the actual format specifying how to extract information still has to be specified, taking into account suitable existing standards. On the right of Figure 7.2.2-2 the resulting NGSI-LD entity is represented that can be accessed using the NGSI-LD API.

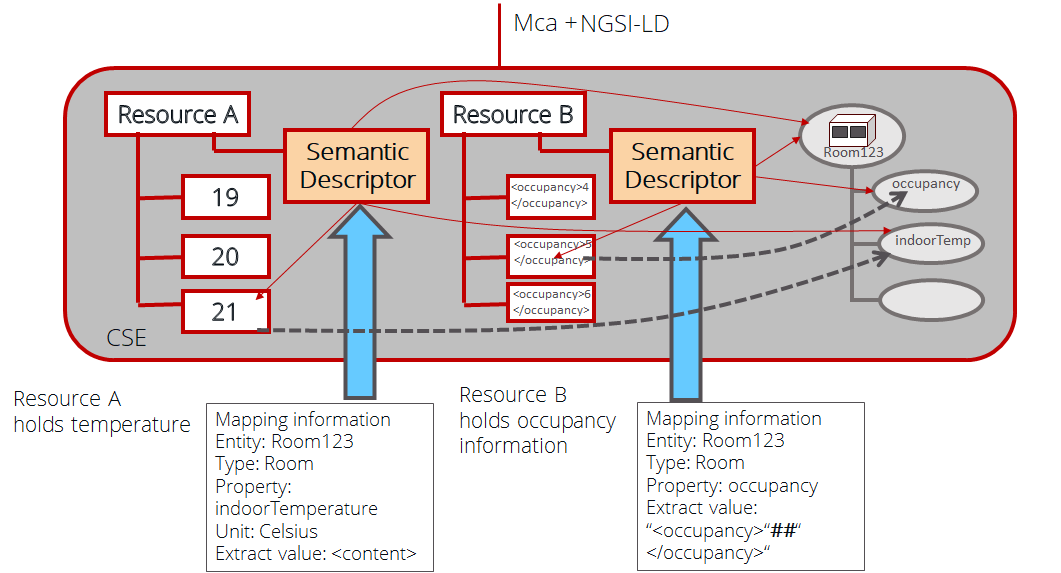


Figure 7.2.2-2: Mapping example - oneM2M information mapped to NGSI-LD Entity

### 7.2.3 Integration in oneM2M architecture

NGSI-LD provides a new set of functionalities to oneM2M, thus it should be its own CSF (logical grouping of functionalities), see Figure 7.2.3-1. The name of such a new CSF is up for discussion, e.g. NGSI-LD, Context Management or Context Information Access.

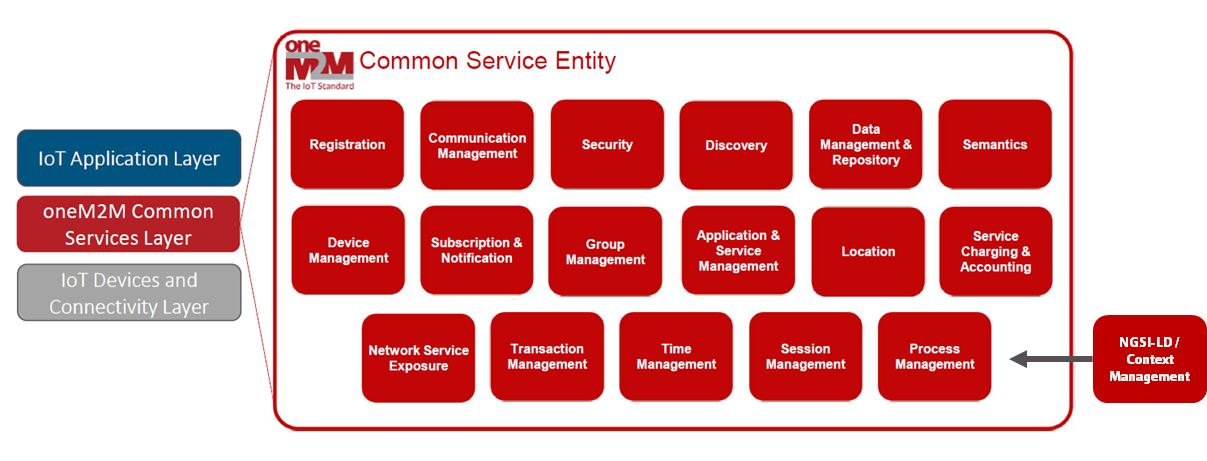


Figure 7.2.3-1: NGSI-LD as CSF in oneM2M

The NGSI-LD API also needs to be added to one or more oneM2M reference points. It can be added to one or more existing reference points. Alternatively, one or more additional reference points could be added. Figure 7.2.3-2 shows the options and Table 7.2.3-1 discusses pros and cons for adding NGSI-LD to existing reference point(s).

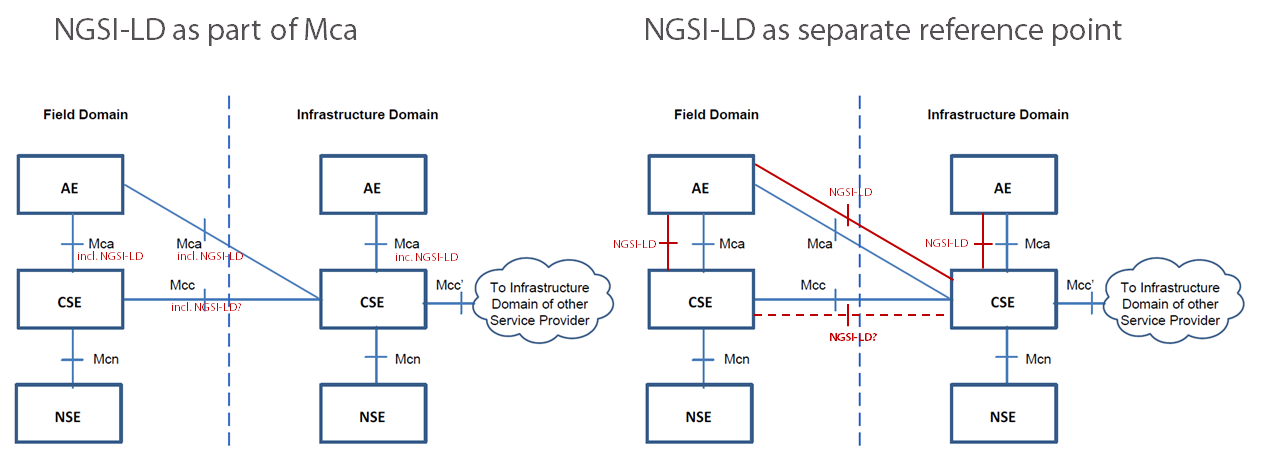


Figure 7.2.3-2: NGSI-LD in oneM2M reference points

Table 7.2.3-1: Pros and Cons for adding NGSI-LD to existing reference point(s)

| Pros | Cons |
| --- | --- |
| Interaction is between the same oneM2M Entities – AE and CSE (or CSE and CSE) | Core Mca and NGSI-LD make very different assumptions, resulting in a heterogeneous reference point. oneM2M is a resource-based architecture with a REST interface, whereas NGSI-LD is a functional API based on the NGSI-LD entity concept, with a REST binding (REST resources are either defined in the specification or are derived from NGSI-LD Entity structure – they cannot be freely created) |

NGSI-LD foresees distributed operations based on registrations as shown in Figure 7.2.3-3. If one CSEs registers with another CSE (e.g. parent CSE), information can be transparently accessed through the CSE to CSE connection.

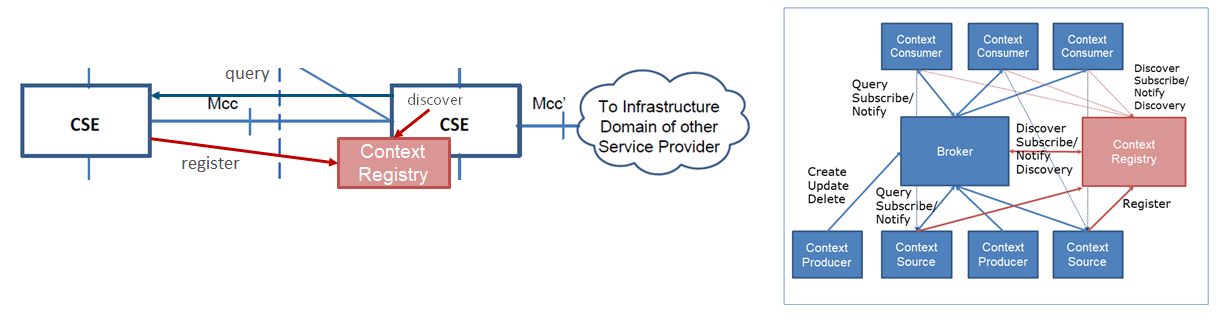


Figure 7.2.3-3: NGSI-LD between CSEs

Figure 7.2.3-4 shows the NGSI-LD REST/HTTP binding. NGSI-LD resources are either static, i.e. exist from the start as defined in the NGSI-LD specification, or are implicitly defined by the information. For example, an NGSI-LD Entity is created and as a result one or more resources exist, so information is created/updated/deleted/queried etc., not resources as such, and there can be multiple ways to create the same information – e.g. as individual entity or via batch operations.

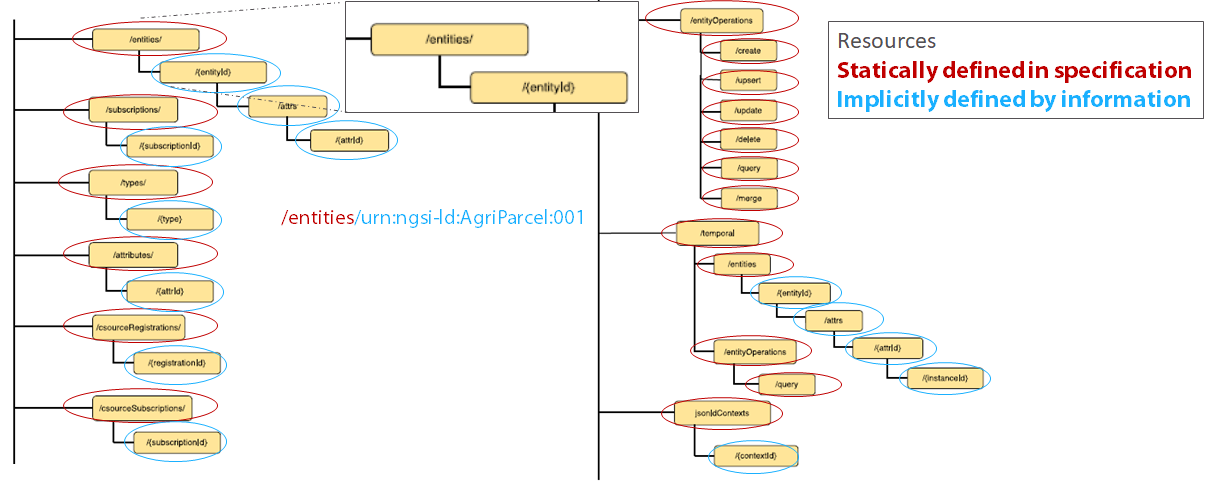


Figure 7.2.3-4: NGSI-LD REST/HTTP binding

In the proposed mapping approach, NGSI-LD resources would be implicitly created through the mapping information.

The following base path / resource is defined in the NGSI-LD Specification [i.2] and thus always exists:

<path>/ngsi-ld/v1/entities

Here <path> is used as a placeholder. In oneM2M the respective binding and addressing needs to be taken into account that enables targetting the ngsi-ld resource. This is going to be discussed separately, and the remainder of the path is used to target the relevant subresource.

By adding the mapping as shown in Figure 7.2.3-5, the following resources would implicitly become available:

<path>/ngsi-ld/v1/entities/urn:Room123  
<path>/ngsi-ld/v1/entities/urn:Room123/attrs/indoorTemperature

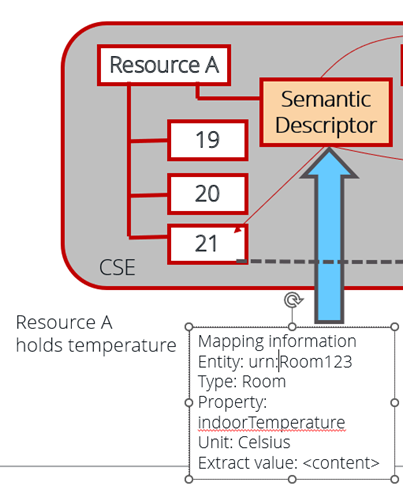


Figure 7.2.3-5: Implicit resource creation by adding mapping

The final question is how to anchor the NGSI-LD base resource ngsi-ld into the oneM2M resource structure. As shown in Figure 7.2.3-6, the proposal is to add it as a child resource of CSEBase.

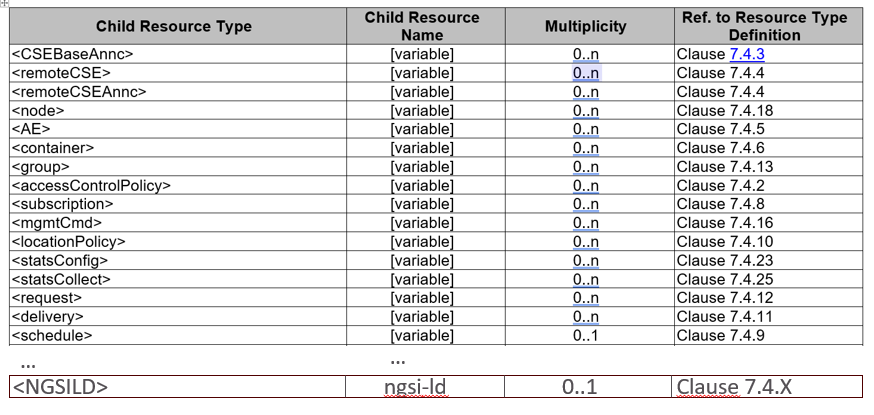


Figure 7.2.3-6: NGSI-LD resource as child resource of CSE Base

# 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> |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

| 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\_0075\_NGSI-LD\_API\_and\_Information\_Model |
| V0.3.0 | 2024-09-11 | Includes the following contribution agreed during SDS66: SDS-2024-0118R01-Value\_provided\_by\_NGSI-LD\_use\_cases |
| V0.4.0 | 2024-11-19 | Includes the following contribution agreed during SDS67: SDS-2024-0142R02-Mapping\_Approach-oneM2M\_Information\_via\_NGSI-LD |
| V0.5.0 | 2025-02-13 | Includes the following contribution agreed during SDS68: SDS-2025-0014R01-Mapping\_Approach\_for\_integrating\_NGSI-LD\_into\_oneM2M\_Architecture |
|  |  |  |
|  |  |  |