ETSI GR CIM 008 V1.3.1 (2025-08)

GROUP REPORT

Context Information Management (CIM); NGSI-LD Primer

Reference

RGR/CIM-008NGSI-LD-PriV131

Keywords

API, interoperability, IoT, smart city

Contents

Intellectual Property Rights

Foreword

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

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Executive summary

The present document (this "Primer") is intended to give developers an introduction on how the NGSI-LD API is used. The aim is to give developers, especially to those building applications and services on top of the NGSI-LD API, an easy start by explaining the NGSI-LD API based on typical examples. For illustration purposes a scenario is introduced, for which the information is modelled according to the NGSI-LD information model. Examples for providing information, i.e. creating, updating and deleting information, and for requesting information, i.e. synchronous queries as well as asynchronous subscribe/notify interactions, are given. The focus is on typical usage rather than on completeness of all features.

Introduction

While ETSI GS CIM 009 [i.1] provides the complete specification of the NGSI-LD API, the present document, called "Primer", is intended to give users an introduction to the use of the NGSI-LD API. The idea is to take a simple scenario, i.e. a store that sells products to customers, for illustration purposes and show typical NGSI-LD API operation examples. The examples for information provision show how Entities, Properties and Relationships can be created, updated, appended, replaced and deleted. The examples for information consumption show how Entities can be synchronously queried, filtered according to Property values or filtered according to geographical location using geographic queries. Finally change-based and time-based subscriptions are introduced and how these create asynchronous subscriptions depending on a change-based or time-based trigger.

1 Scope

The present document provides an introduction, in particular for developers, on how the NGSI-LD API, defined in ETSI GS CIM 009 [i.1], is used. The focus is on typical use and is based on a small NGSI-LD data model example, where the data model conforms to the NGSI-LD information model. More information about the NGSI-LD information model can be found in ETSI GR CIM 002 [i.2].

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

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

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents may be useful in implementing an ETSI deliverable or add to the reader's understanding, but are not required for conformance to the present document.

[i.1] ETSI GS CIM 009: "Context Information Management (CIM); NGSI-LD API".

[i.2] ETSI GR CIM 002: "Context Information Management (CIM); Use Cases (UC)".

[i.3] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

[i.4] IEEE POSIX 1003.2™-1992: "IEEE Standard for Information Technology - Portable Operating System Interfaces (POSIX®) - Part 2: Shell and Utilities".

[i.5] IETF RFC 7946: "The GeoJSON Format".

[i.6] ISG CIM Forge: "Postman Scripts for NGSI-LD Primer".

[i.7] IETF RFC 5646: "Tags for Identifying Languages".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GS CIM 009 [i.1] and the following apply:

NOTE 1: The letters "NGSI-LD" were added to most terms to confirm that they are distinct from other terms of similar/same name in use in other organizations, however, in the present document the letters "NGSI-LD" are generally omitted for brevity.

NOTE 2: The terms defined in this clause are capitalized throughout the present document.

NGSI-LD Attribute: reference to both an NGSI-LD Property and to an NGSI-LD Relationship

NGSI-LD Attribute Instance (in case of temporal representation of NGSI-LD Entities): reference to an NGSI-LD Attribute, at a specific moment in time of its temporal evolution, usually identified by its instanceId

NGSI-LD Central Broker: NGSI-LD Context Broker that only uses a local storage when serving NGSI-LD requests, without involving any external Context Sources

NGSI-LD Context Broker: architectural component that implements all the NGSI-LD interfaces

NGSI-LD Context Consumer: agent that uses the query and subscription functionality of NGSI-LD to retrieve context information

NGSI-LD Context Producer: agent that uses the NGSI-LD context provision and/or registration functionality to provide or announce the availability of its context information to an NGSI-LD Context Broker

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

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

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

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

NGSI-LD Context Source Registrations: description of the information that can be provided by a Context Source, which is used when registering the Context Source with the Context Registry

NGSI-LD Core API: core part of the NGSI-LD API that has to be implemented by all Brokers, including operations for providing or managing Entities and Attributes, operations for consuming Entities and checking which Entity Types and Attributes Entities are available in the system and operations for subscribing to Entities, receiving notifications and managing subscriptions

NGSI-LD Distribution Broker: NGSI-LD Context Broker that uses both local context information and registration information from an NGSI-LD Context Registry, to access matching context information from a set of distributed Context Sources

NGSI-LD Entity: informational representative of something that is supposed to exist in the real world, physically or conceptually

NOTE: In the NGSI-LD API, any instance of such an Entity is **uniquely identified by a URI**, and characterized by reference to one or more **NGSI-LD Entity Type(s)**.

NGSI-LD Entity Type: categorization of an NGSI-LD Entity as belonging to a class of similar Entities, or sharing a set of characteristic Properties

NOTE: In the NGSI-LD API, an NGSI-LD Entity Type is **uniquely identified by a URI**.

EXAMPLE 1: "Vehicle" is an NGSI-LD Entity Type and is identified with a proper URI.

EXAMPLE 2: Bob's private car whose plate number is "ABCD1234" is an NGSI-LD Entity whose NGSI-LD Entity Type Name is "Vehicle".

EXAMPLE 3: Alice's motorhome has a unique URI as id, but can be assigned multiple NGSI-LD Entity Types, e.g. "Vehicle" and "Home".

NGSI-LD Federation Broker: Distribution Broker that federates information from multiple underlying NGSI-LD Context Brokers and across domains

NGSI-LD GeoProperty: subclass of NGSI-LD Property which is a description instance which associates a main characteristic, i.e. an **NGSI-LD Value**, to either an NGSI-LD Entity, an NGSI-LD Relationship or another NGSI-LD Property, that uses the special *hasValue* Property to define its target value and holds a geographic location in GeoJSON Format

NGSI-LD JsonProperty: subclass of NGSI-LD Property which is a description instance which associates a raw JSON literal value as a defined main characteristic to an NGSI-LD Entity, an NGSI-LD Relationship or another NGSI-LD Property and that uses the special *hasJson* (a subproperty of *hasValue*) property to define its target value

NOTE: The target value contains data which is not available for interpretation.

NGSI-LD Language Map: JSON-LD language map in the form of key-value pairs holding the string representation of a main characteristic in a series of natural languages

EXAMPLE: "Bob's vehicle is currently parked on a street which is known as 'Grand Place' in French and 'Grote Markt' in Dutch" can be represented by an NGSI-LD LanguageProperty whose Name is "street" which holds an NGSI-LD Language Map of two key-value pairs containing both the French ("fr") and Dutch ("nl") exonyms of the street name.

NGSI-LD LanguageProperty: subclass of NGSI-LD Property which is a description instance which associates a set of strings in different natural languages as a defined main characteristic, i.e. an **NGSI-LD Map**, to an NGSI-LD Entity, an NGSI-LD Relationship or another NGSI-LD Property and that uses the special *hasLanguageMap* (a subproperty of *hasValue*) Property to define its target value

NGSI-LD Linked Entity: NGSI-LD Entity referenced from another NGSI-LD Entity (the linking NGSI-LD Entity) via an NGSI-LD Relationship

NGSI-LD Linking Entity: NGSI-LD Entity which is the subject of a Relationship to another NGSI-LD Entity (the linked NGSI-LD Entity) or an external resource (identified by a URI)

NGSI-LD ListProperty: description instance which associates an ordered array of main characteristics, i.e. **NGSI-LD Values**, to either an NGSI-LD Entity, an NGSI-LD Relationship or another NGSI-LD Property and that uses the special *hasValueList* property to define its target value

NGSI-LD ListRelationship: description of an ordered array of directed links between a subject which is either an NGSI-LD Entity, an NGSI-LD Property or another NGSI-LD Relationship on one hand, and a series of objects, which are NGSI-LD Entities, on the other hand, and which uses the special *hasObjectList* property to define its target objects

EXAMPLE: "A bus route services the following bus stops" can be represented by an NGSI-LD *ListRelationship* whose name is "route" which holds an array of directed links towards a series of NGSI-LD Entities of type (Type name) `"BusStop"`.

NGSI-LD Name: short-hand string (term) that locally identifies an NGSI-LD Entity Type, Property Type or Relationship Type and which can be mapped to a URI which serves as a fully qualified identifier

EXAMPLE: The sentence "Bob's vehicle's speed is 40 km/h" can be represented by an NGSI-LD Property, whose Name is "speed", and which characterizes an NGSI-LD Entity, which NGSI-LD Type Name is "Vehicle". Such a name can be expanded to a fully qualified name in the form of a URI, for instance "http://example.org/Vehicle" or "http://example.org/speed".

NGSI-LD Null: *"urn:ngsi-ld:null"* or *{"@none": "urn:ngsi-ld:null"}* used as an encoding for *null* values

NGSI-LD Property: description instance which associates a main characteristic, i.e. an **NGSI-LD Value**, to either an NGSI-LD Entity, an NGSI-LD Relationship or another NGSI-LD Property and that uses the special *hasValue* Property to define its target value

NGSI-LD Query: collection of criteria used to select a sub-set of NGSI-LD Entities, matching the criteria

NGSI-LD Registry API: part of the NGSI-LD API that is implemented by the Context Registry, including operations for registering Context Sources and managing Context Source Registrations (CSRs), operations for retrieving and discovering CSRs, and operations for subscribing to CSRs and receiving notifications

NGSI-LD Relationship: description of a directed link between a subject which is either an NGSI-LD Entity, an NGSI-LD Property, or another NGSI-LD Relationship on one hand, and an object, which is an NGSI-LD Entity, on the other hand, and which uses the special *hasObject* Property to define its target object

EXAMPLE: An NGSI-LD Entity of type (Type Name) "Vehicle" (when parked) can be the subject of an NGSI-LD Relationship which object is an NGSI-LD Entity of type "Parking".

NGSI-LD Scope: enables putting Entities into a hierarchical structure and scoping queries and subscriptions according to it

NGSI-LD Temporal API: part of the NGSI-LD API pertaining to the Temporal Evolution of Entities, including operations for providing and managing the Temporal Evolution of Entities and Attributes, and operations for consuming the Temporal Evolution of Entities

NGSI-LD Temporal Evolution of Entities: sequence of values attributed to them over time, i.e. their history or future predictions

NGSI-LD Tenant: user or a group of users that utilize a single instance of a system implementing the NGSI-LD API (NGSI-LD Context Source or NGSI-LD Broker) in isolation from other users or groups of users of the same instance

NOTE: Any information related to one Tenant (e.g. Entities, Subscriptions, Context Source Registrations) are only visible to users of the same Tenant, but not to users of a different Tenant.

NGSI-LD Value: JSON value (i.e. a string, a number, true or false, an object, an array), or a JSON-LD typed value (i.e. a string as the lexical form of the value together with a type, defined by an XSD base type or more generally an IRI), or a JSON-LD structured value (i.e. a set, a list, a language-tagged string)

EXAMPLE: Bob's private car 'speed' NGSI-LD Value is the number 100 (kilometres per hour).

NGSI-LD VocabProperty: subclass of NGSI-LD Property which is a description instance which associates a string value which can be coerced to a URI as a defined main characteristic, i.e. an NGSI-LD Vocabulary, to an NGSI-LD Entity, an NGSI-LD Relationship or another NGSI-LD Property and that uses the special *hasVocab* (a subproperty of *hasValue*) Property to define its target value

NGSI-LD Vocabulary: string representation of a main characteristic which is explicitly defined to undergo JSON-LD type coercion to a URI

EXAMPLE: "Bob's car is a non-commercial vehicle" can be represented by an NGSI-LD VocabProperty whose Name is "category" which holds an NGSI-LD Vocabulary with the string value *"non-commercial"*. If the associated JSON-LD context defines the term *"non-commercial"* as *"http://example.com/ non-commercial"*, then the returned value will be the expanded using type coercion into the IRI the *http://example.com/ non-commercial*.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS CIM 009 [i.1] and the following apply:

API	Application Programming Interface
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
IoT	Internet of Things
IRI	Internationalized Resource Identifier
ISG	Industry Specification Group
JSON	JavaScript Object Notation
JSON-LD	JSON Linked Data
NGSI	Next Generation Service Interfaces
NGSI-LD	NGSI Linked Data
POSIX	Portable Operating System Interface
RFC	Request For Comments
URI	Uniform Resource Identifier
URL	Universal Resource Locator

4 Motivation and an Example Use Case

The concept of Entity is at the core of the NGSI-LD model. Entities represent physical or conceptual objects existing in the real world. Entities can have Properties describing aspects of the object the Entity stands for and Relationships to other Entities. What kind of Properties and Relationships Entities can have, is determined by the Entity Type, which in turn can be defined as part of a data model.

The NGSI-LD API and the Context Brokers implementing it are only based on the abstract NGSI-LD information model, which defines the Entity concept, and that Entities can have Properties and Relationships. They are agnostic to the data model, i.e. what Entity Types exist and what concrete Properties and Relationship the Entity instances can have. Thus, Context Brokers cannot enforce the conformance to a specific data model, making this the responsibility of the users and their applications.

As example use case, the present document is using a system for managing context information related to grocery stores as depicted in Figure 4-1. It shows two Entity instances of Entity Type (grocery) store "6-Stars" and "Checker Market" with its location depicted on a map, a product "Wine" and a customer "Paul".

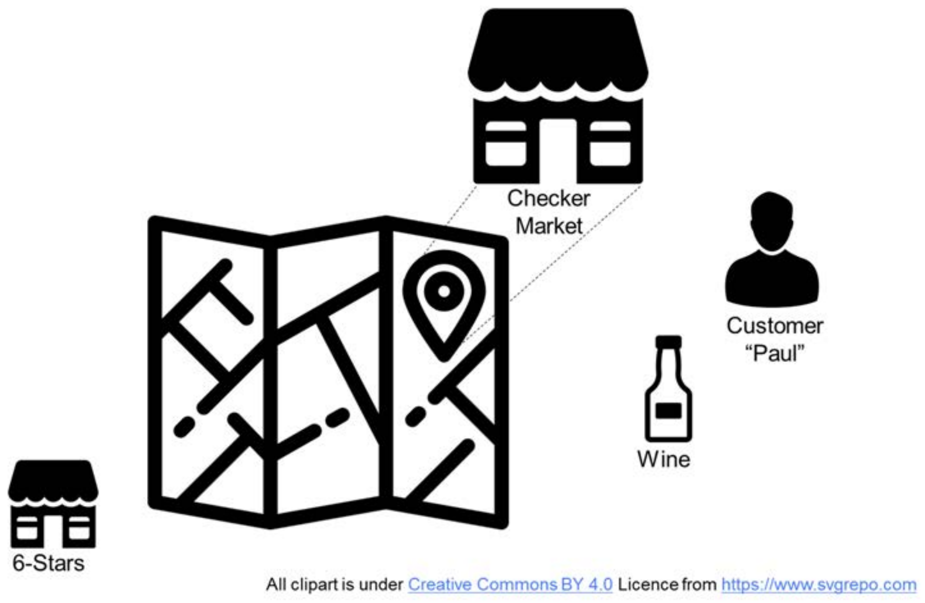

Checker Market
Customer "Paul"
Wine
6-Stars
All clipart is under Creative Commons BY 4.0 Licence from https://www.svgrepo.com

Figure 4-1: Grocery store use case example

The Entity Types used in the example are Store, Customer, Shelf, Inventory item and Product. Figure 4-2 shows the Entity Types together with the Properties and Relationships that Entity Instances of the respective Entity Type can have. As a convention for this example, Properties are defined as nouns, whereas Relationships are defined as verbs. (The use of this convention is not a requirement of NGSI-LD.)

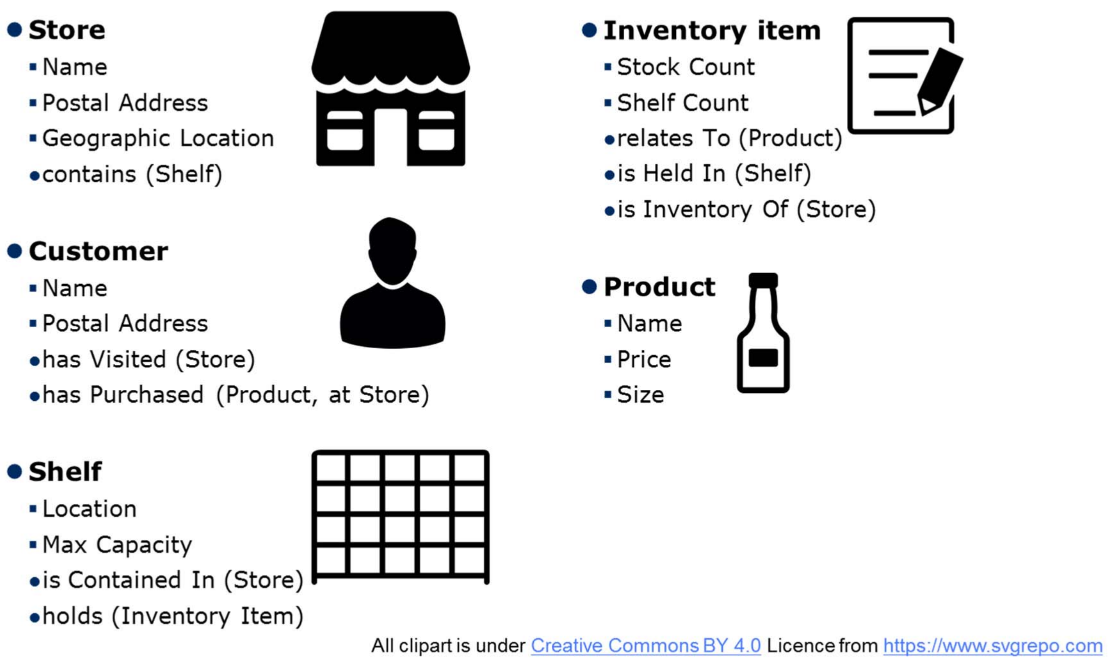


Figure 4-2: Entity Types, Relationships and Properties of use case

5 Getting Started

5.1 Introduction

The purpose of clause 5 is to give a first introduction to the NGSI-LD representation and API operations using the HTTP binding. The examples can be used in an HTTP client (e.g. Postman or curl), targeting an NGSI-LD implementation, i.e. a Context Broker.

To make it easier to try out NGSI-LD, all examples in the present document are available online as Postman scripts [i.6]. Using one of the available (open-source) implementations of NGSI-LD Brokers, the interested reader can play around with the examples and get a hands-on experience of NGSI-LD.

5.2 Architectural Assumptions

NGSI-LD defines an API together with an underlying information model. It does not define a specific system architecture, but instead it is envisioned that the NGSI-LD API can be used in different architectural settings and the architectural assumptions of the API are kept to a minimum. For the following examples, the present document is using the simplest architectural setup, i.e. a Central Context Broker that stores all information. The resulting architecture is depicted in Figure 5.2-1 and requires a certain subset of NGSI-LD operations that are introduced in clauses 5, 6, 7, 8, 9 and 10.

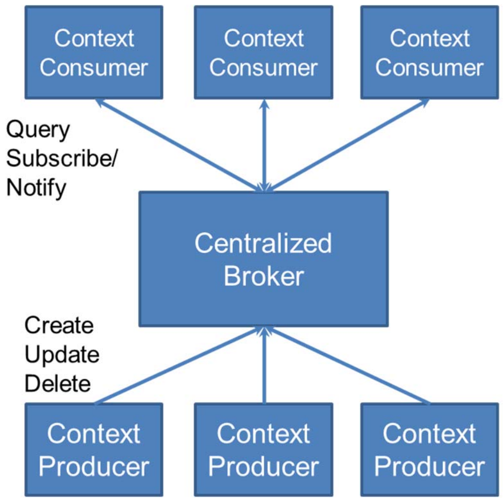


Figure 5.2-1: Basic architectural assumptions

The roles in this setup are Context Producers, Context Consumers and a Central Broker. The assumption is that the Central Broker stores all information. Context Producers manage information, i.e. create, update and delete it, whereas Context Consumers synchronously query information or subscribe to information to be asynchronously notified. In the following, the assumption is that the Central Broker exposes the NGSI-LD API on `localhost:9090`. Context Producers and Context Consumers are roles. The same software program can have both roles at the same time, i.e. manage and request information.

It is planned to introduce more advanced architectural options in a future version of this Primer.

5.3 Creating Entities and Properties

As Entities are at the core of NGSI-LD, the following HTTP request creates a store Entity with the id `urn:ngsi-ld:Store:001` of type `Store` (mapped to `https://uri.etsi.org/ngsi-ld/primer/Store`) in `@context`, the Properties `address` and `storeName` and the GeoProperty `location` (all mapped to the respective URI concepts in `@context`) as shown in Figure 5.3-1.

```
POST /ngsi-ld/v1/entities/ HTTP/1.1
Host: localhost:9090
Content-Type: application/ld+json

{
    "@context": [
        {
            "Store": "https://uri.etsi.org/ngsi-ld/primer/Store",
            "address": "https://uri.etsi.org/ngsi-ld/primer/address",
            "storeName": "https://uri.etsi.org/ngsi-ld/primer/storeName",
            "streetAddress": "https://uri.etsi.org/ngsi-ld/primer/streetAddress",
            "addressRegion": "https://uri.etsi.org/ngsi-ld/primer/addressRegion",
            "addressLocality": "https://uri.etsi.org/ngsi-ld/primer/addressLocality",
            "postalCode": "https://uri.etsi.org/ngsi-ld/primer/postalCode"
        },
            "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.9.jsonld"
    ],
    "id": "urn:ngsi-ld:Store:001",
    "type": "Store",
    "address": {
        "type": "Property",
        "value": {
            "streetAddress": "Main Street 65",
            "addressRegion": "Metropolis",
            "addressLocality": "Duckburg",
            "postalCode": "42000"
        }
    },
    "location": {
        "type": "GeoProperty",
        "value": {
            "type": "Point",
            "coordinates": [57.4874121, -20.2845607]
        }
    },
    "storeName": {
        "type": "Property",
        "value": "Checker Market"
    }
}
```

Figure 5.3-1: Entity creation

If the creation was successful, the response in Figure 5.3-2 with HTTP return code 201 Created is returned.

```
HTTP/1.1 201 Created
Date: Wed, 03 Apr 2019 15:08:33 GMT
location: /ngsi-ld/v1/entities/urn:ngsi-ld:Store:001
```

Figure 5.3-2: Entity creation result

NGSI-LD defines the three special Properties `location`, `observationSpace` and `operationSpace` as `GeoProperty`. A GeoProperty encodes a geographical location in GeoJSON format. A GeoProperty can be used for the scope of geographic queries, whereby the API specification requires that such queries will only return results based on that scope.

5.4 Retrieving Entities and Properties

Now that the store Entity with the id `urn:ngsi-ld:Store:001` has been created, it can be retrieved. The request is shown in Figure 5.4-1 and the successful result in Figure 5.4-2. Note that since no @context was provided in the request, only the core context is returned, whereas all user-defined aspects are returned as URIs [i.3]. The URIs prefixed with "`ngsi-ld`" appear instead of the full URIs, because the NGSI-LD Core context (`https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.9.jsonld`), which is considered the default JSON-LD @context, if no other @context is specified, defines the `ngsi-ld` namespace for `https://uri.etsi.org/ngsi-ld`, i.e. on doing the JSON-LD compaction of the results by the Broker, this namespace is used. How to provide the @context in requests without body is presented in clause 8.

```
GET /ngsi-ld/v1/entities/urn:ngsi-ld:Store:001 HTTP/1.1
Host: localhost:9090
Accept: application/ld+json
```

Figure 5.4-1: Entity retrieval

```
HTTP/1.1 200 OK
Date: Wed, 03 Apr 2019 15:50:09 GMT
Content-Type: application/ld+json

{
  "id": "urn:ngsi-ld:Store:001",
  "type": "ngsi-ld:primer/Store",
  "location": {
    "type": "GeoProperty",
    "value": {
      "type": "Point",
      "coordinates": [ 57.4874121, -20.2845607 ]
    }
  },
  "ngsi-ld:primer/address": {
    "type": "Property",
    "value": {
      "ngsi-ld:primer/addressLocality": "Duck Village",
      "ngsi-ld:primer/addressRegion": "Metropolis",
      "ngsi-ld:primer/postalCode": "42000",
      "ngsi-ld:primer/streetAddress": "Main Street 65"
    }
  },
  "ngsi-ld:primer/storeName": {
    "type": "Property",
    "value": "Checker Market"
  },
  "@context": [ "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.9.jsonld" ]
}
```

Figure 5.4-2: Entity retrieval result

5.5 Overview

Clause 6 explains the NGSI-LD Information Model in more detail and clause 7 focuses on the NGSI-LD representation in JSON-LD and how it is used in the specified HTTP binding. Clause 8 gives typical examples of how NGSI-LD information is managed, i.e. created, appended, updated, replaced and deleted. Clause 9 describes how to synchronously retrieve and query NGSI-LD information and clause 10 gives examples for subscribing to NGSI-LD information and asynchronously retrieve notifications. Clause 11 provides examples for Batch Operations, Clause 12 shows how to find out what types of entities and what attributes are actually available. Clause 13 gives examples of how the Temporal API of NGSI-LD can be used.

Developers eager to experiment with NGSI-LD operations can also decide to immediately jump to clause 8 and clause 9, and only go back to clause 6 and clause 7 in case there are open questions.

6 Information Model

As already introduced, the core underlying concepts of the NGSI-LD information model, also referred to as the NGSI-LD meta-model, are Entity, Relationship and Property. Logically, the Entity Types, Relationships and Properties are modelled as subclasses of the core concepts Entity, Relationship and Property respectively. For our example use case, the Entity Types, Relationships and Properties are shown in Figure 6-1. Note that the concepts defined in the NGSI-LD specification are shown in bold. This includes the special Property `location`, which is defined as a GeoProperty in the specification ETSI GS CIM 009 [i.1].

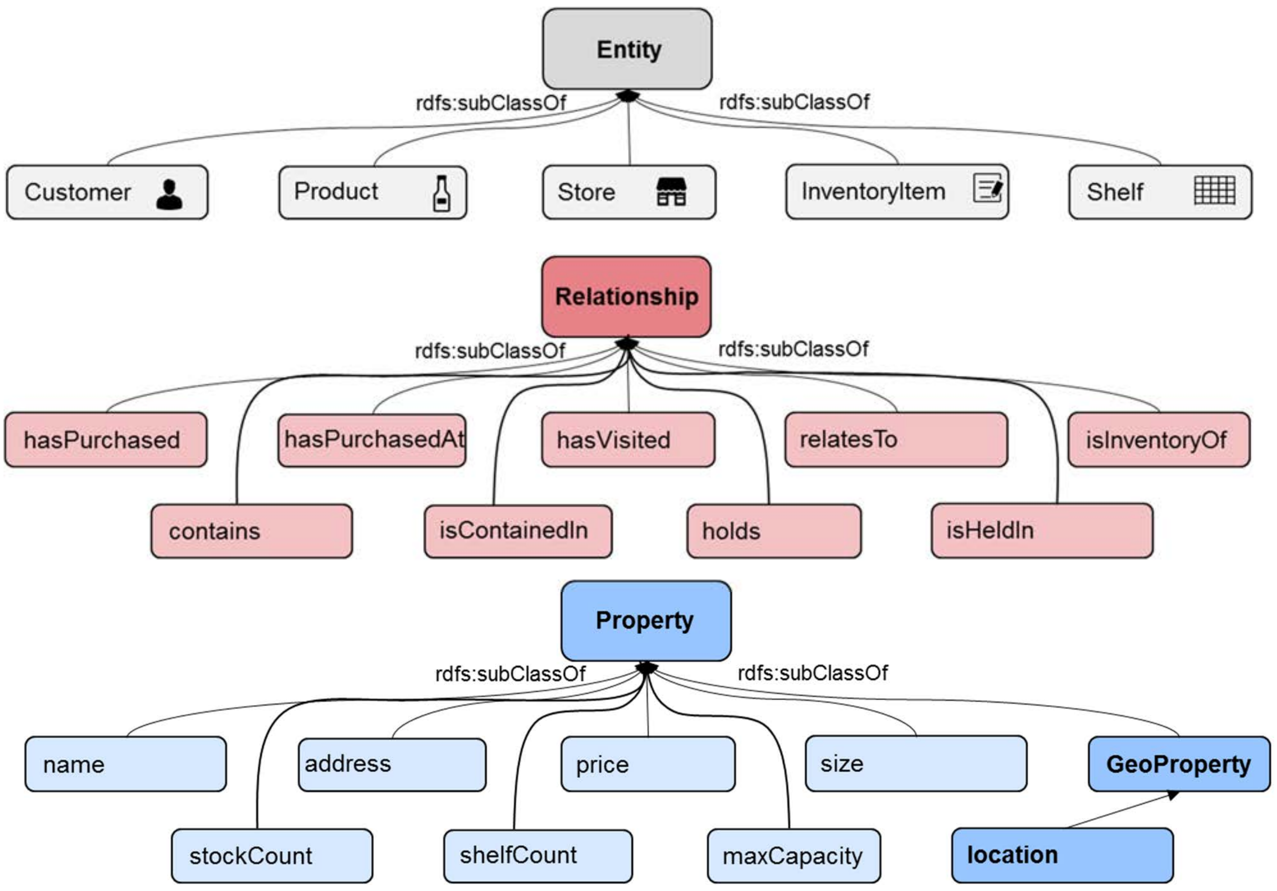
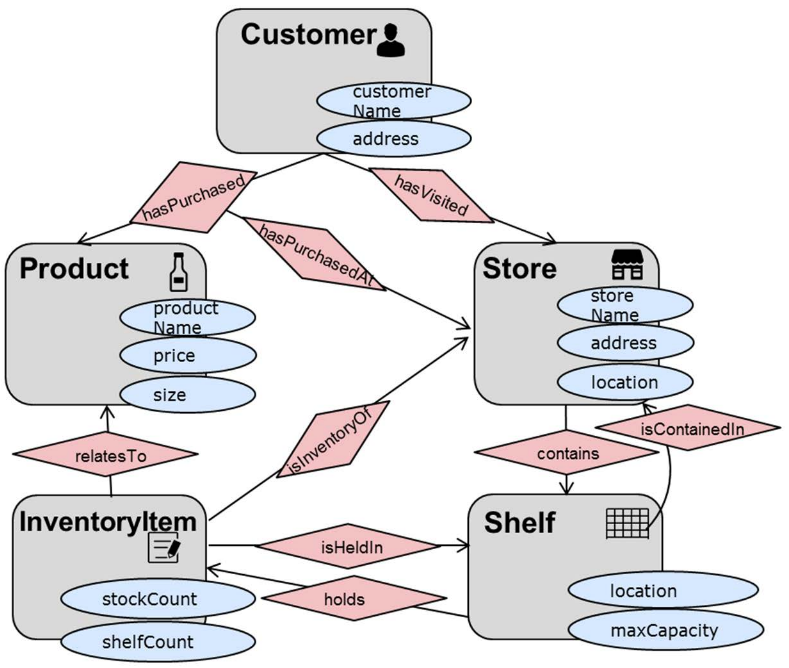


Figure 6-1: Entity Types, Properties and Relationships of the use case

The subclass Relationships can be explicitly modelled in an ontology, but for the purpose of using the NGSI-LD API, they can also be implicitly assumed based on how they are used, i.e. as Entity Types or Relationship and Property names. The important aspect for the use in the NGSI-LD API is that the respective concepts are explicitly defined as URIs in the @context.



All clipart is under Creative Commons BY 4.0 Licence from https://www.svgrepo.com

Figure 6-2: Visualization of Entity Types, Properties and Relationships of the use case

Figure 6-2 visualizes the Entity Types as rounded rectangles with the respective Properties inside, whereas the Relationships are shown as red coloured diamonds on directed arrows.

Properties have an Entity of a certain type or another Property or Relationship as their domain and have a Value as their range. The domain defines of what type the respective Entities can be for which the Property appears, e.g. a Property `price` makes sense for a Product, but not for a Customer. The range indicates what type of Value they can have, e.g. a `stockCount` for example would need to be of type integer. In Figure 6-2 the Properties are shown together with the Entity Type that can be their domain. Domain and range of a certain Property can be defined in an ontology, but this is not required for use in the NGSI-LD API.

Relationships in NGSI-LD are directed. Again, their domain and range indicate what types of source and target Entities are valid for the Relationship. For example, the `hasPurchased` Relationship is only valid for Customer Entities as domain and Product Entities as range. In Figure 6-2 possible directed Relationships are visualized as red coloured diamonds on directed arrows. In order to have inverse Relationships, these need to be explicitly modelled, e.g. as the `contains` and `isContainedIn` Relationships between Store and Shelf/Shelf and Store respectively.

For the JSON-LD based representation of NGSI-LD, it is important to have the mapping of the short terms used in the information representation to the URIs uniquely identifying the Entity Types, Relationships and Properties mapped in the `@context`. Figure 6-3 shows the complete `@context` for all the Entities, Relationships and Properties introduced in our example use case. Note that the GeoProperty "location" is not defined here as it is already part of the NGSI-LD core context.

The NGSI-LD core context is stored in file `ngsi-ld-core-contextv1.9.jsonld`, which can be retrieved from `https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.9.jsonld`. As the NGSI-LD terms take precedence over any user-defined terms, the reference to the core @context is required by the API specification to always be the last element of any NGSI-LD compatible `@context`. In case it is not provided in a request, NGSI-LD systems will implicitly add it as the last element.

```
    "@context": [{
            "Customer": "https://uri.etsi.org/ngsi-ld/primer/Customer",
            "InventoryItem": "https://uri.etsi.org/ngsi-ld/primer/InventoryItem",
            "Product": "https://uri.etsi.org/ngsi-ld/primer/Product",
            "Shelf": "https://uri.etsi.org/ngsi-ld/primer/Shelf",
            "Store": "https://uri.etsi.org/ngsi-ld/primer/Store",
            "Wine": "https://uri.etsi.org/ngsi-ld/primer/Wine",
            "address": "https://uri.etsi.org/ngsi-ld/primer/address",
            "addressLocality": "https://uri.etsi.org/ngsi-ld/primer/addressLocality",
            "addressRegion": "https://uri.etsi.org/ngsi-ld/primer/addressRegion",
            "brand": "https://uri.etsi.org/ngsi-ld/primer/brand",
            "contains": "https://uri.etsi.org/ngsi-ld/primer/contains",
            "customerName": "https://uri.etsi.org/ngsi-ld/primer/customerName",
            "description": "https://uri.etsi.org/ngsi-ld/primer/description",
            "hasPurchased": "https://uri.etsi.org/ngsi-ld/primer/hasPurchased",
            "hasPurchasedAt": "https://uri.etsi.org/ngsi-ld/primer/hasPurchasedAt",
            "hasVisited": "https://uri.etsi.org/ngsi-ld/primer/hasVisited",
            "holds": "https://uri.etsi.org/ngsi-ld/primer/holds",
            "isContainedIn": "https://uri.etsi.org/ngsi-ld/primer/isContainedIn",
            "isHeldIn": "https://uri.etsi.org/ngsi-ld/primer/isHeldIn",
            "isInventoryOf": "https://uri.etsi.org/ngsi-ld/primer/isInventoryOf",
            "isMeasuredBy": "https://uri.etsi.org/ngsi-ld/primer/isMeasuredBy",
            "jsonType": "https://uri.etsi.org/ngsi-ld/primer/jsonType",
            "lastOrder": "https://uri.etsi.org/ngsi-ld/primer/lastOrder",
            "maxCapacity": "https://uri.etsi.org/ngsi-ld/primer/maxCapacity",
            "minCapacity": "https://uri.etsi.org/ngsi-ld/primer/minCapacity",
            "owner": "https://uri.etsi.org/ngsi-ld/primer/owner",
            "postalCode": "https://uri.etsi.org/ngsi-ld/primer/postalCode",
            "price": "https://uri.etsi.org/ngsi-ld/primer/price",
            "productName": "https://uri.etsi.org/ngsi-ld/primer/productName",
            "putBackOnShelf": "https://uri.etsi.org/ngsi-ld/primer/putBackOnShelf",
            "quantity": "https://uri.etsi.org/ngsi-ld/primer/quantity",
            "relatesTo": "https://uri.etsi.org/ngsi-ld/primer/relatesTo",
            "shelfCount": "https://uri.etsi.org/ngsi-ld/primer/shelfCount",
            "size": "https://uri.etsi.org/ngsi-ld/primer/size",
            "stockCount": "https://uri.etsi.org/ngsi-ld/primer/stockCount",
            "storeName": "https://uri.etsi.org/ngsi-ld/primer/storeName",
            "streetAddress": "https://uri.etsi.org/ngsi-ld/primer/streetAddress"
            },
            "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.9.jsonld"
   ]
```

Figure 6-3: @context of example use case

The @context shown in Figure 6-3 is available at https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld.

NGSI-LD enables providing meta information to Properties and Relationships. This is done in the form of Properties of Properties, Properties of Relationships, Relationships of Properties and Relationships of Relationships. An example of a Property of a Property could be the timestamp when the Property was created or modified, or quality information. A Relationship of a property could refer to the Entity representing the source of information. The same applies for Relationships.

```
    "storeName": {
         "type": "Property",
         "value": "6-Stars",
         "createdAt": "2019-08-09T18:08:02.669000Z",
         "providedBy": {
               "type": "Relationship",
               "object": "urn: ngsi-ld:Person:123"
    }
```

Figure 6-4: Example of a Relationship of Property

An example is shown for the Property `storeName` in Figure 6-4. A Relationship to the Entity representing the person who provided the `storeName` Property is given. In general Properties and Relationships of Properties are modelled as regular Properties or Relationships, i.e. they are JSON objects with the respective type. An exception are specific timestamps that are modelled as temporal Properties, i.e. `createdAt`, `modifiedAt` and `observedAt`, which are all defined in NGSI-LD's core @context. These, and only these special timestamps, are represented directly by their value, not as a JSON object and thus they cannot have any further Properties or Relationships.

As already mentioned in clause 5.3, a further special type of Properties are GeoProperties. They are provided with type `GeoProperty` instead of `Property` and the API specification requires that their value be provided in GeoJSON [i.5]. Figure 6-5 shows an example of the `location GeoProperty`, which is a special GeoProperty defined by NGSI-LD and is part of the core @context. It is also possible to have user-defined GeoProperties.

```
    "location": {
        "type": "GeoProperty",
        "value": {
            "type": "Point",
            "coordinates": [57.5522, -20.3484]
         }
    }
```

Figure 6-5: Example of a GeoProperty

Only GeoProperties can be used to filter entities according to their geographic location as is explained in clause 9.4.

7 Information Representation

NGSI-LD is represented in JSON-LD, which stands for JavaScript Object Notation for Linked Data. JSON-LD is an ordinary JSON document, but it contains specific reserved names that only JSON-LD readers can interpret (like @id @type and @context). NGSI-LD maps id to @id and type to @type, so only @context may appear directly, i.e. outside value of @context in an NGSI-LD document.

As presented in clause 6, the most important aspect of JSON-LD is @context that maps short terms to unique URIs, identifying unique concepts for Entity Types, Properties and Relationships. Using the short terms in the core of the documents leads to a more readable and succinct JSON representation of the information itself.

In addition to this mapping, types can be specified in cases the data type required for a value is not a simple JSON type. Figure 7-1 shows an excerpt of the NGSI-LD core @context that defined the terms required in the NGSI-LD API specification itself - and can be retrieved through the URL `https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.9.jsonld`. For example, the API specification requires that the value of the `observedAt` temporal Property be of type `DateTime`.

```
"@context": {
      "ngsi-ld": "https://uri.etsi.org/ngsi-ld/",
      "id": "@id",
      "type": "@type",
      "value": "https://uri.etsi.org/ngsi-ld/hasValue",
      "object": {
        "@id": "https://uri.etsi.org/ngsi-ld/hasObject",
        "@type":"@id"
      },
   ...
   "observedAt": {
      "@id": "https://uri.etsi.org/ngsi-ld/observedAt",
      "@type": "DateTime"
   },
   ...
```

Figure 7-1: Excerpt of NGSI-LD core @context

In an HTTP binding, there are two general approaches for providing the @context of JSON-LD. Either the @context is provided as part of the JSON document in the HTTP body, e.g. as shown in the example in Figure 5.3-1, or a URL to the @context is provided as an HTTP Link header. Of course, the former way of providing an @context is only feasible if there is an HTTP body as in POST, PUT or PATCH requests.

8 Information Provision

8.1 Overview Information Provision

Clause 8 describes the NGSI-LD API operations for information provision, i.e. the operations that can be used to provide and manage Entity information. Figure 8.1-1 gives an overview of the available information provision operations.

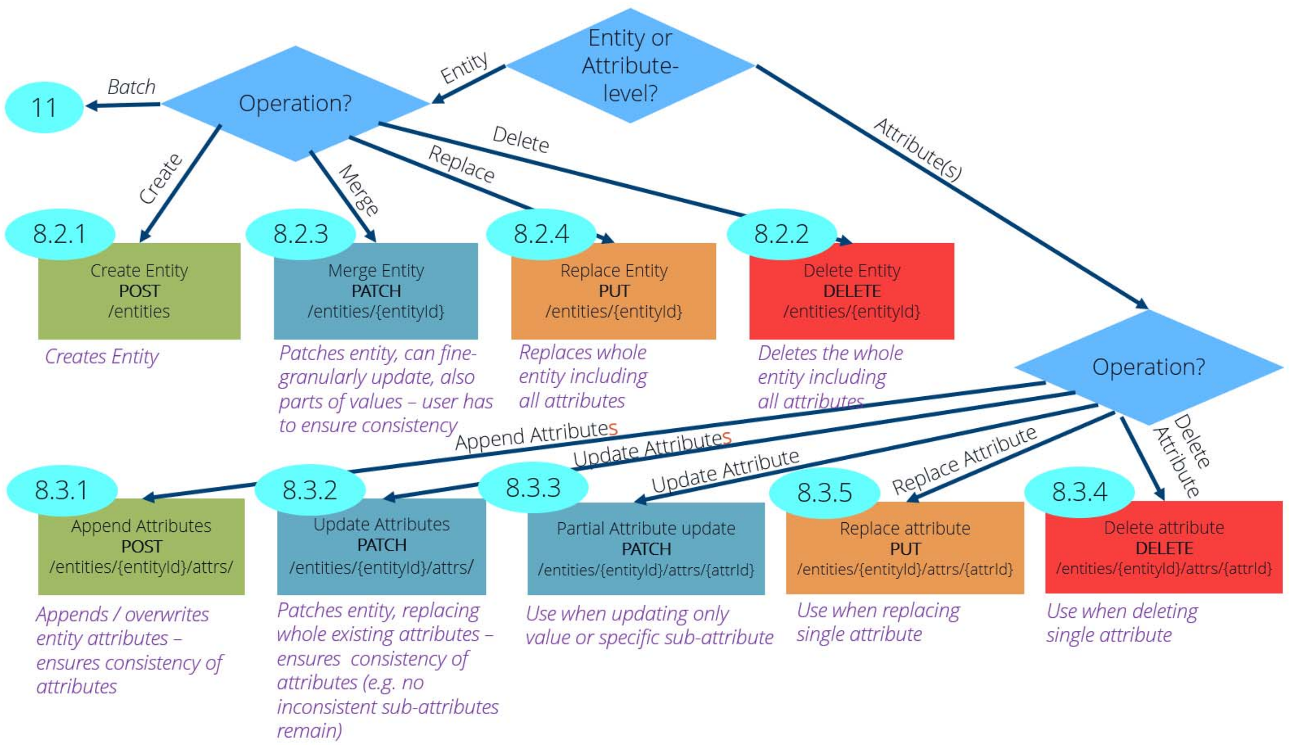

Entity or Attribute-level?
Entity
Attribute(s)
Operation?
Batch
11
Create
Merge
Replace
Delete
8.2.1
Create Entity
POST
/entities
Creates Entity
8.2.3
Merge Entity
PATCH
/entities/{entityId}
Patches entity, can fine-granularly update, also parts of values – user has to ensure consistency
8.2.4
Replace Entity
PUT
/entities/{entityId}
Replaces whole entity including all attributes
8.2.2
Delete Entity
DELETE
/entities/{entityId}
Deletes the whole entity including all attributes
Operation?
Append Attributes
Update Attributes
Update Attribute
Replace Attribute
Delete Attribute
8.3.1
Append Attributes
POST
/entities/{entityId}/attrs/
Appends / overwrites entity attributes – ensures consistency of attributes
8.3.2
Update Attributes
PATCH
/entities/{entityId}/attrs/
Patches entity, replacing whole existing attributes – ensures consistency of attributes (e.g. no inconsistent sub-attributes remain)
8.3.3
Partial Attribute update
PATCH
/entities/{entityId}/attrs/{attrId}
Use when updating only value or specific sub-attribute
8.3.5
Replace attribute
PUT
/entities/{entityId}/attrs/{attrId}
Use when replacing single attribute
8.3.4
Delete attribute
DELETE
/entities/{entityId}/attrs/{attrId}
Use when deleting single attribute

Figure 8.1-1: Overview of information provision operations

Clause 8.2 describes the operations for creating, merging, replacing and deleting complete Entities. Clause 8.3 describes the operations for appending and updating single or multiple Attributes and updating, replacing and deleting individual Attributes (Properties and Relationships). When updating information, only a fragment with the relevant aspects of the Attributes to be updated needs to be provided. For Entities, there are also batch operations, which are described in clause 11.

8.2 Creating/Deleting/Merging and Replacing Entities

8.2.1 Create Entity Using Link Header

Figure 8.2.1-1 shows the NGSI-LD request to create another `Store` Entity with its Attributes. In this case, the @context is provided as a Link in the HTTP header pointing to the URL `https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld`. This means that the `@context` needed to expand the JSON-LD information in the body can be retrieved from this URL. This is reflected by the Content-Type `application/json`. The alternative is shown in Figure 5.3-1, where the `@context` is part of the JSON in the HTTP body. The Content-Type in that case is `application/ld+json`. For brevity, all further operations presented in clause 8 use the Link header for referring to the `@context` and thus the `application/json` content type, which in this case contains the complete `@context` of the example use case. Alternatively, the `@context` could be provided in the HTTP body using the content type `application/ld+json`, given that the HTTP operation does have a body.

```
POST /ngsi-ld/v1/entities/ HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

{
   "id": "urn:ngsi-ld:Store:002",
   "type": "Store",
   "address": {
        "type": "Property",
        "value": {
            "streetAddress": "Tiger Street 4",
            "addressRegion": "Metropolis",
            "addressLocality": "Cat City",
            "postalCode": "42420"
        }
    },
    "location": {
        "type": "GeoProperty",
        "value": {
            "type": "Point",
            "coordinates": [57.5522, -20.3484]
         }
    },
    "storeName": {
         "type": "Property",
         "value": "6-Stars"
    }
}
```

Figure 8.2.1-1: Store Entity creation using link header

If the creation was successful, the response in Figure 8.2.1-2 with HTTP return code `201` Created is returned.

```
HTTP/1.1 201 Created
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/entities/urn:ngsi-ld:Store:002
```

Figure 8.2.1-2: Store Entity creation result

For the example use case, further Entities are needed. Figure 8.2.1-3 gives an example for creating a `Shelf` Entity and Figure 8.2.1-4 shows the successful result.

```
POST /ngsi-ld/v1/entities/ HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

{
      "id": "urn:ngsi-ld:Shelf:123",
      "type": "Shelf",
      "maxCapacity": {
         "type": "Property",
         "value": 100
      }
}
```

Figure 8.2.1-3: Shelf Entity creation using link header

```
HTTP/1.1 201 Created
Date: Mon, 09 Sep 2019 13:20:23 GMT
location: /ngsi-ld/v1/entities/"urn:ngsi-ld:Shelf:123"
```

Figure 8.2.1-4: Shelf Entity creation result

8.2.2 Delete Entity

```
DELETE /ngsi-ld/v1/entities/urn:ngsi-ld:Store:002 HTTP/1.1
Host: localhost:9090
```

Figure 8.2.2-1: Store Entity deletion

Figure 8.2.2-1 shows the HTTP request for deleting the `Store` Entity with the id `urn:ngsi-ld:Store:002` and Figure 8.2.2-2 the result in the case of successful deletion.

```
HTTP/1.1 204 No Content
Date: Wed, 04 Apr 2019 11:47:18 GMT
```

Figure 8.2.2-2: Store Entity deletion result

NOTE: In the following, the present document assumes that the Entity with identifier `urn:ngsi-ld:Store:002` exists, so if it is deleted, it should be re-created again as shown in Figure 8.2.1-1.

8.2.3 Merge Entity

Merge Entity allows the fine-grained patching of an Entity. This allows detailed changes, even the change of JSON elements inside a value. Also, it is possible to delete JSON elements by setting them to NGSI-LD null (`"urn:ngsi-ld:null"` for regular Properties and Relationships, or `{"@none": "urn:ngsi-ld:null"}` for Language Maps). The use of regular JSON `null` values is not possible due to the special use of the value `null` in JSON-LD (which removes the element on JSON-LD expansion).

As the user can independently change aspects with this operation, the user is also responsible for ensuring consistency of the Entity. For example, changing a value, but not changing the related meta information, like an `observedAt` timestamp, may lead to a semantically inconsistent Attribute.

Figure 8.2.3-1 shows a merge Entity request that changes the `streetAddress` to `"Superman Street 4"` and removes the `addressRegion` element from the `value` of the `address` Property of the `Store:002` Entity. As shown in the example, for the merge Entity request, the Content-Type `application/merge-patch+json` can be used, in addition the `application/json` and `application/ld+json` content types.

```
PATCH /ngsi-ld/v1/entities/urn:ngsi-ld:Store:002 HTTP/1.1
Host: localhost:9090
Content-Type: application/merge-patch+json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"

{
   "id": "urn:ngsi-ld:Store:002",
   "type": "Store",
   "address": {
        "type": "Property",
        "value": {
            "streetAddress": "Superman Street 4",
            "addressRegion": "urn:ngsi-ld:null",
            "addressLocality": "Cat City",
            "postalCode": "42420"
        }
    }
}
```

Figure 8.2.3-1: Change street address of Store:002 and remove addressRegion element

Figure 8.2.3-2 shows the result of the successful execution of the merge Entity request and Figure 8.2.3-3 shows the `Store:022` Entity after the execution of the merge Entity request.

```
HTTP/1.1 204 No Content
Date: Thu, 03 Aug 2023 09:41:37 GMT
```

Figure 8.2.3-2: Change street address of Store:002 and remove addressRegion element result

```
{
     "id": "urn:ngsi-ld:Store:002",
     "type": "Store",
     "location": {
          "type": "GeoProperty",
          "isMeasuredBy": {
               "type": "Relationship",
               "object": "urn:ngsi-ld:GPSTracker:001"
          },
          "value": {
               "type": "Point",
               "coordinates": [
                    57.5522,
                    -20.3484
               ]
          }
     },
     "address": {
          "type": "Property",
          "value": {
               "postalCode": "42420",
               "streetAddress": "Superman Street 4",
               "addressLocality": "Cat City"
          }
     },
     "storeName": {
          "type": "Property",
          "value": "6-Stars"
     },
     "@context": [
          "https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld"
     ]
}
```

Figure 8.2.3-3: Store:002 after executing the request

8.2.4 Replace Entity

Replace Entity allows replacing the complete Entity, i.e. it deletes the previous Entity information, including all Attributes and Entity Types and replaces them with the provided Entity information. Only the system-generated `createdAt` timestamps are kept, which is the only difference from first deleting the Entity and the creating it again.

In Figure 8.2.4-1, `Store:002` is taken over by a new owner, who wants to make sure only the newly provided information is kept, changing the `streetAddress` back to `"Tiger Street 4"` and the `storeName` to `"Luxury Store"`.

```
PUT /ngsi-ld/v1/entities/urn:ngsi-ld:Store:002 HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
Content-Length: 359

{
   "id": "urn:ngsi-ld:Store:002",
   "type": "Store",
   "address": {
        "type": "Property",
        "value": {
            "streetAddress": "Tiger Street 4",
            "addressLocality": "Cat City",
            "postalCode": "42420"
        }
    },
    "location": {
        "type": "GeoProperty",
        "value": {
            "type": "Point",
            "coordinates": [
                57.5522,
                -20.3484
            ]
        }
    },
    "storeName": {
        "type": "Property",
        "value": "Luxury Store"
    }
}
```

Figure 8.2.4-1: Replace Store:002 Entity

Figure 8.2.4-2 shows the result of the successful execution of the replace Entity request.

```
HTTP/1.1 204 No Content
Date: Thu, 03 Aug 2023 15:49:13 GMT
```

Figure 8.2.4-2: Replace Store:002 Entity result

Figure 8.2.4-3 shows how to request the `urn:ngsi-ld:Store:002` Entity with system Attributes by adding the parameter `options=sysAttrs`, and Figure 8.2.4-4 shows the Entity with system Attributes, where `createdAt` shows the time of the original creation of the Entity, whereas `modifiedAt` shows the time when the Entity was replaced.

```
GET /ngsi-ld/v1/entities/urn:ngsi-ld:Store:002?options=sysAttrs HTTP/1.1
Host: localhost:9090
Accept: application/ld+json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
```

Figure 8.2.4-3: Requestion Store:002 Entity, with sysAttrs included

```
{
     "id": "urn:ngsi-ld:Store:002",
     "type": "Store",
     "createdAt": "2019-04-04T11:42:15Z",
     "modifiedAt": "2023-08-03T15:49:13Z",
     "address": {
          "type": "Property",
          "value": {
               "postalCode": "42420",
               "streetAddress": "Tiger Street 4",
               "addressLocality": "Cat City"
          },
          "createdAt": "2019-04-04T11:42:15Z",
          "modifiedAt": "2023-08-03T15:49:13Z"
     },
     "storeName": {
          "type": "Property",
          "value": "Luxury Store",
          "createdAt": "2019-04-04T11:42:15Z",
          "modifiedAt": "2023-08-03T15:49:13Z"
     },
     "@context": [
          "https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld"
     ]
}
```

Figure 8.2.4-4: Store:002 Entity after executing the request, with sysAttrs included

8.3 Appending/Updating/Replacing/Deleting Attributes

8.3.1 Appending Attributes

Entities that have been created are not static but change over time. This includes adding new Properties and Relationships as opposed to updating the value or object of existing Properties and Relationships respectively. In the use case, a shelf may be put up in a store and this fact can be modelled as a `contains` Relationship. Figure 8.3.1-1 shows the append operation that adds the `contains` Relationship to the store Entity with id `urn:ngsi-ld:Store:001` that points to the shelf Entity `urn:ngsi-ld:Shelf:123`. The successful result is shown in Figure 8.3.1-2.

```
POST /ngsi-ld/v1/entities/urn:ngsi-ld:Store:001/attrs HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
   "contains":{
       "type":"Relationship",
       "object":"urn:ngsi-ld:Shelf:123",
       "observedAt":"2019-09-09T12:09:07Z"
   }
}
```

Figure 8.3.1-1: Append Relationship to Store Entity

```
HTTP/1.1 204 No Content
Date: Mon, 09 Sep 2019 14:08:15 GMT
```

Figure 8.3.1-2: Append Relationship to Store Entity result

Relationships in NGSI-LD are unidirectional, i.e. if the inverse Relationship is required, then the API specification requires that it has to be explicitly created. In Figure 8.3.1-3, the `isContainedIn` Relationship is appended to the shelf with id `urn:ngsi-ld:Shelf:123` that points to the store Entity with id `urn:ngsi-ld:Store:001`, thus it represents the inverse of the contains Relationship added to the store Entity as shown in Figure 8.3.1-1. As request in Figure 8.3.1-3 is the same operation, the result generally looks the same as in Figure 8.3.1-2.

```
POST /ngsi-ld/v1/entities/urn:ngsi-ld:Shelf:123/attrs HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
   "isContainedIn":{
       "type":"Relationship",
       "object":"urn:ngsi-ld:Store:001",
       "observedAt":"2019-09-09T14:09:07Z"
   }
}
```

Figure 8.3.1-3: Append Shelf Entity Relationship using link header

8.3.2 Updating Attributes

If an existing Property or Relationship is to be updated, there is a corresponding NGSI-LD operation. An example is shown in Figure 8.3.2-1. In the example the name of the store has changed from "Checker Market" to "Mega Market", thus the Property `storeName` of the store Entity with id `urn:ngsi-ld:Store:001` is updated accordingly. Figure 8.3.2-2 shows the successful result.

```
PATCH /ngsi-ld/v1/entities/urn:ngsi-ld:Store:001/attrs HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
   "storeName":{
       "type":"Property",
       "value":"Mega Market"
   }
}
```

Figure 8.3.2-1: Property update using link header

```
HTTP/1.1 204 No Content
Date: Fri, 02 Aug 2019 15:20:13 GMT
```

Figure 8.3.2-2: Property update result

8.3.3 Partial Update of a single Attribute

If only parts of a Property or Relationship are to be updated, there is also the possibility of a partial update, where only a fragment of the Attribute with the elements to be updated is provided. Figure 8.3.3-1 shows the update of the name on the storeName resource representing the Attribute, where only the changed value and not the complete Property information is provided, unlike the example in Figure 8.3.2-1 where the complete Attribute is updated. Figure 8.3.3-2 shows the successful result.

```
PATCH /ngsi-ld/v1/entities/urn:ngsi-ld:Store:001/attrs/storeName HTTP/1.1
Host: localhost:9090
Content-Type:application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
       "value": "Mega Store"
}
```

Figure 8.3.3-1: Partial update of storeName Entity Property using link header

```
HTTP/1.1 204 No Content
Date: Mon, 09 Sep 2019 12:45:41 GMT
```

Figure 8.3.3-2: Partial update of storeName Entity update result

In Figure 8.3.3-3, an example is shown that not only the value of a Property or the object of a Relationship can be updated with a partial update, but also meta information, e.g. represented as a Property of a Property. In the example, the `observedAt` Temporal Property of the `contains` Relationship that was appended in Figure 8.3.1-1 is updated, e.g. to make it consistent with the `observedAt` Property of the inverse Relationship added in Figure 8.3.1-3. Figure 8.3.3-4 shows the successful result.

```
PATCH /ngsi-ld/v1/entities/urn:ngsi-ld:Store:001/attrs/contains HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
       "observedAt":"2019-09-09T14:09:07Z"
}
```

Figure 8.3.3-3: Partial update of Property of contains Relationship using link header

```
HTTP/1.1 204 No Content
Date: Mon, 09 Sep 2019 14:57:27 GMT
```

Figure 8.3.3-4: Partial update of Property of contains Relationship result

8.3.4 Deleting a single Attribute

If a Property or Relationship is no longer valid, it can be deleted. In the example in Figure 8.3.4-1, the shelf has been removed from the store and thus the `contains` Relationship of the store Entity with id `urn:ngsi-ld:Store:001` is deleted. Figure 8.3.4-2 shows the successful result. In case there is an inverse Relationship, it has to be deleted separately.

```
DELETE /ngsi-ld/v1/entities/urn:ngsi-ld:Store:001/attrs/contains HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
```

Figure 8.3.4-1: Delete Relationship contains using link header

```
HTTP/1.1 204 No Content
Date: Mon, 09 Sep 2019 15:10:35 GMT
```

Figure 8.3.4-2: Delete Relationship contains result

8.3.5 Replacing a single Attribute

A single Attribute can also be replaced completely. Figure 8.3.5-1 shows how the `storeName` Property is replaced, adding a Relationship to the owner of the name. Figure 8.3.5-2 shows the successful result after executing the operation.

```
PUT /ngsi-ld/v1/entities/urn:ngsi-ld:Store:001/attrs/storeName HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
Content-Length: 239

{
     "type": "Property",
     "value": "Sustainable Shopping Market",
     "owner": {
          "type": "Relationship",
          "object": "urn:ngsi-ld:SustainableShopping:456"
     },
     "observedAt": "2023-08-03T16:00:05Z"
}
```

Figure 8.3.5-1: Replace Property storeName

```
HTTP/1.1 204 No Content
Date: Thu, 03 Aug 2023 16:08:48 GMT
```

Figure 8.3.5-2: Replace Property storeName result

9 Information Consumption

9.1 Overview Information Consumption

Clause 9 describes the NGSI-LD API operations for information consumption, i.e. the operations that can be used to synchronously request Entity information. Clause 9.2 describes the operations for retrieving information about a known Entity, i.e. based on the Entity identifier. Clause 9.3 describes the operations for querying Entity information based on Entity Type, i.e. discovering new Entities, filtering according to Property values or Relationship objects. Clause 9.4 introduces geographical queries, i.e. discovering Entities of a certain type whose (current) location is within the specified geographic area. Clause 9.6 describes an alternate way to query Entities with HTTP POST requests, suitable option to overcome known limitations of URL parameters (e.g. the maximum length of a URL).

9.2 Retrieving Information

9.2.1 Retrieving Entity Using Simplified Representation

In clause 5.4 an example is given how the full representation of an Entity can be retrieved. Such a representation can include additional information about Properties and Relationships, e.g. Properties of Properties, typically used for meta information like a timestamp or the source of the information. If such meta information is not required, NGSI-LD also offers a simplified representation that only provides the Property with its value or the Relationship with its object in a key value presentation. In order to retrieve an Entity in this representation, `keyValues` needs to be specified as an option. Figure 9.2.1-1 shows an example of the request for retrieving the Entity with the Entity identifier `urn:ngsi-ld:Store:001` with `options=keyValues,` using link header for providing the `@context`.

```
GET /ngsi-ld/v1/entities/urn:ngsi-ld:Store:001?options=keyValues HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
```

Figure 9.2.1-1: Retrieval of simplified representation of Store:001 Entity

```
HTTP/1.1 200 OK
Date: Wed, 03 Apr 2019 15:50:09 GMT
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

{
    "id": "urn:ngsi-ld:Store:001",
    "location": {
        "type": "Point",
        "coordinates": [
            57.4874121,
            -20.2845607
        ]
    },
    "address": {
        "address:Locality": "Duckburg",
        "addressRegion": "Metropolis",
        "postalCode": "42000",
        "streetAddress": "Main Street 65"
    },
    "storeName": "Sustainable Shopping Market"
}
```

Figure 9.2.1-2: Retrieved result with simplified representation of Store:001 Entity

Figure 9.2.1-2 shows the simplified representation of NGSI-LD Store:001 Entity with only the values for the address, location and storeName Properties.

9.3 Query Language

9.3.1 Querying Entities by Type

This example queries all NGSI-LD Entities of type `Store` (mapped to `https://uri.etsi.org/ngsi-ld/primer/Store` in the `@context`) including all available Properties and Relationships. The request is shown in Figure 9.3.1-1 and the result with two Store Entities in Figure 9.3.1-2. A link header is used for providing the `@context`.

```
GET /ngsi-ld/v1/entities?type=Store HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Accept: application/json
```

Figure 9.3.1-1: Query for Entities by type

```
HTTP/1.1 200 OK
Date: Wed, 05 Apr 2019 16:20:11 GMT
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

[
   {
        "id": "urn:ngsi-ld:Store:001",
        "type": "Store",
        "location": {
            "type": "GeoProperty",
            "value": {
                "type": "Point",
                "coordinates": [
                    57.4874121,
                    -20.2845607
                ]
            }
        },
        "address": {
            "type": "Property",
            "value": {
                "address:Locality": "Duckburg",
                "addressRegion": "Metropolis",
                "postalCode": "42000",
                "streetAddress": "Main Street 65"
            }
        },
        "storeName": {
            "type": "Property",
            "owner": {
                "type": "Relationship",
                "object": "urn:ngsi-ld:SustainableShopping:456"
            },
            "value": "Sustainable Shopping Market",
            "observedAt": "2023-08-03T16:00:05Z"
        }
   },
   {
        "id": "urn:ngsi-ld:Store:002",
        "type": "Store",
        "address": {
            "type": "Property",
            "value": {
                "postalCode": "42420",
                "streetAddress": "Tiger Street 4",
                "addressLocality": "Cat City"
            }
        },
        "storeName": {
            "type": "Property",
            "value": "Luxury Store"
        }
   }
]
```

Figure 9.3.1-2: Query for Entities by type results

9.3.2 Querying Entities by Type, Filtering by Property Value

Instead of retrieving all Entities of a certain type, the results can be filtered using the `q` parameter. The filter can be on a simple Property value as shown in Figure 9.3.2-1, where all Entities of type `Store` (mapped to `https://uri.etsi.org/ngsi-ld/primer/Store` in the `@context`) that have a Property `storeName` (mapped to `https://uri.etsi.org/ngsi-ld/primer/storeName` in the `@context`) whose value is `"6-Stars"`.

```
GET /ngsi-ld/v1/entities?type=Store&q=storeName=="Luxury Store" HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
```

Figure 9.3.2-1: Query for Entities by type, filtered by storeName

As only a single store with the given storeName was created, only this store with the Entity identifier `urn:ngsi-ld:Store:002` is returned as shown in Figure 9.3.2-2.

```
HTTP/1.1 200 OK
Date: Mon, 24 Feb 2020 10:44:30 GMT
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

[
    {
        "id": "urn:ngsi-ld:Store:002",
        "type": "Store",
        "address": {
            "type": "Property",
            "value": {
                "postalCode": "42420",
                "streetAddress": "Tiger Street 4",
                "addressLocality": "Cat City"
            }
        },
        "storeName": {
            "type": "Property",
            "value": "Luxury Store"
        }
    }
   ]
```

Figure 9.3.2-2: Query for Entities by type, filtered by storeName result

It is also possible to filter according to elements of complex values. In Figure 9.3.2-3 the filter is according to the addressRegion element of address. Since both example stores have an address with `addressRegion=="Metropolis"`, the result contains only the Entity with `"id": "urn:ngsi-ld:Store:001"`, as the addressRegion has previously been removed from the `address` of `urn:ngsi-ld:Store:002`.

```
GET /ngsi-ld/v1/entities?type=Store&q=address[addressRegion]=="Metropolis" HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
```

Figure 9.3.2-3: Query for Entities by type, filtered by addressRegion element of address

NOTE: It is generally recommended, and in some cases required, to URL-encode (also referred to as percent encode) any parameter values in HTTP requests [i.3]. For readability reasons, this has not been done here, but the URL-encoded version of the filter would be:
`address%5BaddressRegion%5D%3D%3D%22Metropolis%22`

The NGSI-LD query language used to specify filters not only supports the 'equal' operator used in Figure 9.3.2-3, but all operators specified in Table 9.3.2-1.

Table 9.3.2-1: Operators supported by the NGSI-LD query language

Operator	Name	Comment
==	Equal	Comparison of numbers, quoted strings, dateTime, date and time, boolean, list of values (comma separated), range (dots, e.g. 1..5), URIs
!=	Unequal	Comparison of numbers, quoted strings, dateTime, date and time, boolean, list of values (comma separated), range (dots, e.g. 1..5), URIs
>	Greater	Comparison of numbers, quoted strings, dateTime, date and time
>=	Greater equal	Comparison of numbers, quoted strings, dateTime, date and time
<	Less	Comparison of numbers, quoted strings, dateTime, date and time
<=	Less equal	Comparison of numbers, quoted strings, dateTime, date and time
~=	Pattern	Comparison to a regular expression as described in IEEE POSIX 1003.2™ [i.4]
!~=	Not pattern	Comparison to a regular expression as described in IEEE POSIX 1003.2™ [i.4]

```
GET /ngsi-ld/v1/entities?type=https://w3id.org/ngsi-
ld/store/Store&q=address[addressRegion]=="Metropolis";address[addressLocality]=="Cat City" HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
```

Figure 9.3.2-4: Query for Entities by type with logical AND of two filter elements

It is also possible to combine multiple filters with the logical operators AND (;) or OR (|). Figure 9.3.2-4 shows an example where both `address[addressRegion]=="Metropolis"` and (;) `address[addressLocality]==" Cat City"` have to be true. This is currently not the case for any Entity as `urn:ngsi-ld:Store:002` does not have `addressRegion` set.

9.3.3 Querying Entities by Type, Filtering by Relationship Object

Not only Properties, but also Relationships can be used for filtering. Figure 9.3.3-1 shows an example, where all shelves are requested that have an `isContainedIn` Relationship to the store with URI `urn:ngsi-ld:Store:001`. The result is one shelf with the URI `urn:ngsi-ld:Shelf:123` as Entity identifier as shown in Figure 9.3.3-2.

```
GET /ngsi-ld/v1/entities?type=Shelf&q=isContainedIn==urn:ngsi-ld:Store:001 HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"'
```

Figure 9.3.3-1: Query for Entities by type, filtered by Relationship

```
HTTP/1.1 200 OK
Mon, 24 Feb 2020 10:49:58 GMT
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

[
  {
    "id": "urn:ngsi-ld:Shelf:123",
    "type": "Shelf",
    "isContainedIn": {
      "type": "Relationship",
      "object": "urn:ngsi-ld:Store:001",
      "observedAt": "2019-09-09T14:09:07Z"
    },
    "maxCapacity": {
      "type": "Property",
      "value": 100
    }
  }
]
```

Figure 9.3.3-2: Query for Entities by type, filtered by Relationship result

9.3.4 Querying Entities by Type, Filtering by Meta Information

It is also possible to filter by meta information, i.e. Properties of Properties, Relationships of Properties, Properties of Relationships and Relationships of Relationships. In Figure 9.3.4-1 an example is given, where only shelf Entities with `isContainedIn` Relationships are considered that themselves have a temporal Property `observedAt` that is greater or equal to `2019-09-09T14:09:07Z`. As observedAt is equal to `2019-09-09T14:09:07Z` and there are no other shelf Entities, the result is the same as in Figure 9.3.3-2. If only values for observedAt greater than `2019-09-09T14:09:07Z` are considered as in Figure 9.3.4-2, the result will be the empty list.

```
GET /ngsi-ld/v1/entities?type=Shelf&q=isContainedIn.observedAt>=2019-09-09T14:09:07Z HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"'
```

Figure 9.3.4-1: Query for Entities by type, filtered by meta information

```
GET /ngsi-ld/v1/entities?type=Shelf&q=isContainedIn.observedAt>2019-09-09T14:09:07Z HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"'
```

Figure 9.3.4-2: Query for Entities by type, filtered by meta information leading to empty result

9.4 Geographical Queries

In addition to filtering by Property values, geographic queries allow the filtering of Entities based on their geographic location. NGSI-LD supports special Properties of type GeoProperty that are required by the specification ETSI GS CIM 009 [i.1] to be specified in GeoJson [i.5]. With the geographic query language, a geographic area is specified that is matched against the specified GeoProperty (if no GeoProperty is specified, the default is `location`).
Figure 9.4-1 gives an example, where the geographic query specified a point location given as a GPS coordinate and a radius of 1 000 metres around this coordinate. The result in Figure 9.4-2 contains the Store Entity with Entity identifier `urn:ngsi-ld:Store:002`, whose location is within the specified geographic area.

```
GET /ngsi-
ld/v1/entities?type=Store&geoproperty=location&georel=near;maxDistance==1000&geometry=Point&coordina
tes=[57.5522023,-20.34840123] HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
```

Figure 9.4-1: Query for Entities by type, filtered by geographical query

```
HTTP/1.1 200 OK
Mon, 24 Feb 2020 10:53:41 GMT
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

[
    {
        "id": "urn:ngsi-ld:Store:002",
        "type": "Store",
        "location": {
            "type": "GeoProperty",
            "value": {
                "type": "Point",
                "coordinates": [57.5522,-20.3484]
            }"
        },
        "address": {
            "type": "Property",
            "value": {
                "addressRegion": "Metropolis",
                "postalCode": "42420",
                "streetAddress": "Tiger Street 4",
                "addressLocality": "Cat City"
            }
        },
        "storeName": {
            "type": "Property",
            "value": "6-Stars"
        }
    }
]
```

Figure 9.4-2: Query for Entities by type, filtered by geographic query result

9.5 Count of Results

This example in Figure 9.5-1 queries all NGSI-LD Entities of type `Store` (mapped to `https://uri.etsi.org/ngsi-ld/primer/Store` in the `@context`) including all available Properties and Relationships, as in Figure 9.3.1-1, but in addition sets the query parameter `count` to `true` and `limit` to `0`.

```
GET /ngsi-ld/v1/entities?type=Store&count=true&limit=0 HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Accept: application/json
```

Figure 9.5-1: Query for Entities by type, requesting count

As can be seen in Figure 9.5-2, the header `NGSILD-Results-Count` has the value `2`, as there are two Entities of type `Store` available, but the result in the Body is the empty list, because `limit` was set to `0`. Setting `limit` to a value of `2` or larger would result in both available Entities of type `Store` being returned.

```
HTTP/1.1 200 OK
Date: Wed, 05 Apr 2019 16:20:11 GMT
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json
NGSILD-Results-Count: 2

[]
```

Figure 9.5-2: Result with NGSILD-Results-Count header set and a limit of 0

9.6 POST Query Operation

9.6.1 Overview of POST Query Operation

There are a few cases where some of the queries described in the previous clauses cannot be performed. For instance:

- The client may end up assembling very long URLs that could be cut if they exceed the maximum allowed length.
- In some situations, it may be not desired/possible to URL-encode the resulting URL.

To overcome situations like the ones listed above, it is possible to utilize the POST Query Operation. The substantial difference with normal queries is that the query parameters are stored in the body of the POST HTTP Request.

This clause describes how to perform the same queries in the previous clauses using the POST Query Operation.

9.6.2 Translation of previous queries

9.6.2.1 Querying Entities by Type

Figure 9.6.2.1-1 shows how to perform the query described in Figure 9.3.1-1 using a POST Query Operation.

```
POST /ngsi-ld/v1/entityOperations/query HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Accept: application/json

{
    "type": "Query",
    "entities": [
        {
            "type": "Store"
        }
    ]
}
```

Figure 9.6.2.1-1: POST query for Entities by type

9.6.2.2 Querying Entities by Type, Filtering by Property Value

Figure 9.6.2.2-1 shows how to perform the query described in Figure 9.3.2-1 with POST Query Operations.

```
POST /ngsi-ld/v1/entityOperations/query HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Accept: application/json

{
    "type": "Query",
    "entities": [
        {
            "type": "Store"
        }
    ],
    "q": "storeName==\"Luxury Store\""
}
```

Figure 9.6.2.2-1: POST query for Entities by type, filtered by storeName

Figure 9.6.2.2-2 shows how to perform the query described in Figure 9.3.2-3 with POST Query Operations.

```
POST /ngsi-ld/v1/entityOperations/query HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Accept: application/json

{
    "type": "Query",
    "entities": [
        {
            "type": "Store"
        }
    ],
    "q": "address[addressRegion]==\"Metropolis\""
}
```

Figure 9.6.2.2-2: POST query for Entities by type, filtered by addressRegion element of address

Figure 9.6.2.2-3 shows how to perform the query described in Figure 9.3.2-4 with POST Query Operations.

```
POST /ngsi-ld/v1/entityOperations/query HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Accept: application/json

{
    "type": "Query",
    "entities": [
        {
            "type": "Store"
        }
    ],
    "q": "address[addressRegion]==\"Metropolis\";address[addressLocality]==\"Cat City\""
}
```

Figure 9.6.2.2-3: POST query for Entities by type with logical AND of two filter elements

9.6.2.3 Querying Entities by Type, Filtering by Relationship Object

Figure 9.6.2.3-1 shows how to perform the query described in Figure 9.3.3-1 with POST Query Operations.

```
POST /ngsi-ld/v1/entityOperations/query HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Accept: application/json

{
    "type": "Query",
    "entities": [
        {
            "type": "Shelf"
        }
    ],
    "q": "isContainedIn==urn:ngsi-ld:Store:001"
}
```

Figure 9.6.2.3-1: POST query for Entities by type, filtered by Relationship

9.6.2.4 Querying Entities by Type, Filtering by Meta Information

Figure 9.6.2.4-1 shows how to perform the query described in Figure 9.3.4-1 with POST Query Operations.

```
POST /ngsi-ld/v1/entityOperations/query HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Accept: application/json

{
    "type": "Query",
    "entities": [
        {
            "type": "Shelf"
        }
    ],
    "q": "isContainedIn.observedAt>=2019-09-09T14:09:07Z"
}
```

Figure 9.6.2.4-1: POST query for Entities by type, filtered by meta information

Figure 9.6.2.4-2 shows how to perform the query described in Figure 9.3.4-2 with POST Query Operations.

```
POST /ngsi-ld/v1/entityOperations/query HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Accept: application/json

{
    "type": "Query",
    "entities": [
        {
            "type": "Shelf"
        }
    ],
    "q": "isContainedIn.observedAt>2019-09-09T14:09:07Z"
}
```

Figure 9.6.2.4-2: POST query for Entities by type, filtered by meta information leading to empty result

9.6.2.5 Geographical Queries

Figure 9.6.2.5-1 shows how to perform the query described in Figure 9.4-1 using the POST Query Operation.

```
POST /ngsi-ld/v1/entityOperations/query HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Accept: application/json

{
    "type": "Query",
    "entities": [
        {
            "type": "Store"
        }
    ],
    "geoQ": {
        "geoproperty": "location",
        "georel": "near;maxDistance==1000",
        "geometry": "Point",
        "coordinates": [
            57.5522023,
            -20.34840123
        ]
    }
}
```

Figure 9.6.2.5-1: POST query for Entities by type, filtered by geographical query

9.6.3 Optional URL parameters allowed

As for the Query operation, the POST Query Operation also supports the usage of the optional URL parameters that can transform the result (e.g. Simplified representation, clause 9.1) or request more information (e.g. Count of Results, clause 9.5).

For instance, Figure 9.6.3-1 shows how to request the NGSILD-Results-Count header using the same URL parameter seen in clause 9.5 when filtering Entities by type.

```
POST /ngsi-ld/v1/entityOperations/query?count=true&limit=0 HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Accept: application/json

{
    "type": "Query",
    "entities": [
        {
            "type": "Store"
        }
    ]
}
```

Figure 9.6.3-1: POST query for Entities by type, requesting count

10 Information Subscriptions

10.1 Overview Information Subscriptions

Clause 10 describes the NGSI-LD API operations for subscriptions and the resulting notifications, i.e. the operations that can be used to request Entity information and asynchronously receive the information in form of notifications. Clause 10.2.1 describes change-based subscriptions and how updates and the creation of new Entities trigger notifications. Clause 10.2.2 describes time-based subscriptions and notifications that are triggered whenever the specified time interval has passed, independent of any changes to the Entity information.

10.2 Creating Subscriptions

10.2.1 Change-based Subscriptions, Updates and Resulting Notifications

Change-based subscription are based on the change of an Attribute, i.e. the value of a Property or the object of a Relationship. The Attributes whose changes trigger the notification are specified as a list in the `watchedAttributes` element of a subscription. Only in case this Attribute changes in an existing Entity or a new Entity with this Attribute is created, a notification will be sent.

In Figure 10.2.1-1, a subscription for Entities of type `Store` (mapped to `https://uri.etsi.org/ngsi-ld/primer/Store` in the `@context`) and `watchedAttributes` with the Property `storeName` (mapped to `https://uri.etsi.org/ngsi-ld/primer/storeName` in the `@context`) is shown.

The notification to be sent on a change is further specified in the `notification` element. The `attributes` specified are those to be included for the Entities to be returned, if available. If no Attribute is available for an Entity, it will not be returned. If no Attributes are specified, all available Attributes will be returned.

For the avoidance of any doubt, for the Attributes to be returned, only those specified in `attributes` will be considered. These can include the triggering Attribute from `watchedAttributes` or not - the triggering Attribute is not always returned, only in case it is specified. In the example in Figure 10.2.1-1, the Properties `storeName` and `address` (but not `location`), will be returned, if available.

```
POST /ngsi-ld/v1/subscriptions HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"


{
   "id": "urn:ngsi-ld:Subscription:storeSubscription",
      "type": "Subscription",
      "entities": [
      {
         "type": "Store"
         }
      ],
      "watchedAttributes": ["storeName"],
      "notification": {
         "attributes": ["storeName","address"],
         "format": "normalized",
         "endpoint": {
            "uri": "http://localhost:8000/notify",
            "accept": "application/json"
         }
     }
}
```

Figure 10.2.1-1: Subscription to Entities of type Store, based on changes

For the format, either `normalized` (which is the default) or `keyValue` can be specified. Just as in the case of queries, `keyValue` will only return pairs of Attribute names and values without any additional information, otherwise the complete information will be returned.

In the `endpoint` element, the `uri` specifies HTTP URL to which the notification is to be sent. This requires that the URL is reachable with an HTTP request, i.e. cannot be behind a firewall. The `accept` specifies the JSON-LD format, i.e. valid mime-types are `application/json` (i.e. the `@context` will be provided in a Link header) and `application/ld+json` (i.e. the `@context` will be provided as an element in the JSON body).

```
PATCH /ngsi-ld/v1/entities/urn:ngsi-ld:Store:001/attrs/storeName HTTP/1.1
Host: localhost:9090
Content-Type:application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"

{
       "value": "Mega Store"
}
```

Figure 10.2.1-2: Partial update of storeName Property in urn:ngsi-ld:Store:001 Entity

When updating the name of the Store entity with Entity identifier `urn:ngsi-ld:Store:001` from "Checker Market" to "Mega Store" as shown in Figure 10.2.1-2, the subscription `urn:ngsi-ld:Subscription:storeSubscription` matches as the type is `Store` and the updated Property `storeName` matches an element in the `watchedAttributes` list. Thus, the notification in Figure 10.2.1-3 is triggered.

```
POST /notify HTTP/1.1
Host: localhost:8000
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
Content-Type: application/json
{
   "id": "ngsildbroker:notification:-4999875641436467960",
   "type": "Notification",
   "data": [
      {
         "id": "urn:ngsi-ld:Store:001",
         "type": "Store",
         "address": {
            "type": "Property",
            "createdAt": "2020-02-05T09:04:40.987000Z",
            "value": {
               "address:Locality": "Duck Village",
               "addressRegion": "Metropolis",
               "postalCode": "42000",
               "streetAddress": "Main Street 65"
            },
            "modifiedAt": "2020-02-05T09:04:40.987000Z"
         },
         "storeName": {
            "type": "Property",
            "createdAt": "2020-02-05T09:04:40.987000Z",
            "value": "Mega Store",
            "modifiedAt": "2020-02-05T10:36:22.691000Z"
         }
      }
   ],
   "notifiedAt": "2020-02-05T10:36:23.709000Z",
   "subscriptionId": "urn:ngsi-ld:Subscription:storeSubscription"
}
```

Figure 10.2.1-3: Notification to subscription triggered by update of storeName Property

The notification contains the `subscriptionId`, so it is always clear which subscription triggered the notification and a `notifiedAt` timestamp.

The Entity is contained in the data `element` of the notification and, as specified, includes that Properties address and storeName, including `modifiedAt` timestamp of the `storeName`, i.e. when the Property was updated.

```
POST /ngsi-ld/v1/entities/ HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"

{
   "id": "urn:ngsi-ld:Store:003",
   "type": "Store",
   "address": {
        "type": "Property",
        "value": {
            "streetAddress": "Lion Street 25",
            "addressRegion": "Metropolis",
            "addressLocality": "Cat City",
            "postalCode": "42420"
        }
    },
   "location": {
        "type": "GeoProperty",
            "value": {
               "type": "Point",
               "coordinates": [
                  57.5722,
                  -20.3584
               ]
            }
   },
   "storeName": {
         "type": "Property",
         "value": "Giga Market"
   }
}
```

Figure 10.2.1-4: Creation of new Store Entity with storeName Giga Market

A notification for the subscription can also be triggered by the creation of a new Entity as shown in Figure 10.2.1-4. The creation results in the notification in Figure 10.2.1-5, containing the Entity with the Entity identifier `urn:ngsi-ld:Store:003` and Properties `address` and `storeName` as specified in the subscription.

```
POST /notify HTTP/1.1
Host: localhost:8000
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
Content-Type: application/json
{
   "id": "ngsildbroker:notification:-8311420531544870425",
   "type": "Notification",
   "data": [
      {
         "id": "urn:ngsi-ld:Store:003",
         "type": "Store",
         "address": {
            "type": "Property",
            "createdAt": "2020-02-05T15:54:56.813000Z",
            "value": {
               "addressRegion": "Metropolis",
               "postalCode": "42420",
               "streetAddress": "Lion Street 25",
               "addressLocality": "Cat City"
            },
            "modifiedAt": "2020-02-05T15:54:56.813000Z"
         },
         "storeName": {
            "type": "Property",
            "createdAt": "2020-02-05T15:54:56.813000Z",
            "value": "Giga Market",
            "modifiedAt": "2020-02-05T15:54:56.813000Z"
         }
      }
   ],
   "notifiedAt": "2020-02-05T15:54:57.024000Z",
   "subscriptionId": "urn:ngsi-ld:Subscription:storeSubscription"
}
```

Figure 10.2.1-5: Notification to subscription triggered by creation of new Store with storeName Property

10.2.2 Time-based Subscriptions and Resulting Notifications

Time-based subscriptions are sent after each notification interval, i.e. independent of whether there have been any changes in the Entities that fit the subscription or not.

In Figure 10.2.2-1, a subscription for Entities of type `Store` (mapped to `https://uri.etsi.org/ngsi-ld/primer/Store` in the `@context`) is shown. Notifications are to be sent every `timeInterval`, in this case every `60` seconds. The notification element is the same as in Figure 10.2.1-1, i.e. store Entities with the Attributes `storeName` and `address`, in `normalized` format are to be sent as HTTP POST requests to the notification endpoint that has the uri `http://localhost:8000/notify`.

```
POST /ngsi-ld/v1/subscriptions HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"

{
   "id": "urn:ngsi-ld:Subscription:intervalStoreSubscription",
      "type": "Subscription",
      "entities": [
      {
         "type": "Store"
         }
   ],
      "timeInterval": 60,
    "notification": {
         "attributes": ["storeName","address"],
         "format": "normalized",
         "endpoint": {
            "uri": "http://localhost:8000/notify",
            "accept": "application/json"
         }
    }
}
```

Figure 10.2.2-1: Subscription to Entities of type Store, based on time

In Figure 10.2.2-2, a notification sent as the result of the time-based subscription in Figure 10.2.2-1 is shown.

```
POST /notify HTTP/1.1
Host: localhost:8000
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
Content-Type: application/json
{
  "id": "ngsildbroker:notification:-6431227878819736798",
  "type": "Notification",
  "data": [ {
    "id": "urn:ngsi-ld:Store:002",
    "type": "Store",
    "address": {
      "type": "Property",
      "value": {
        "addressRegion": "Metropolis",
        "postalCode": "42420",
        "streetAddress": "Tiger Street 4",
        "addressLocality": "Cat City"
      }
    },
    "storeName": {
      "type": "Property",
      "value": "6-Stars"
    }
  }, {
    "id": "urn:ngsi-ld:Store:001",
    "type": "Store",
    "address": {
      "type": "Property",
      "value": {
        "address:Locality": "Duck Village",
        "addressRegion": "Metropolis",
        "postalCode": "42000",
        "streetAddress": "Main Street 65"
      }
    },
    "storeName": {
      "type": "Property",
      "value": "Mega Market"
    }
  }, {
    "id": "urn:ngsi-ld:Store:003",
    "type": "Store",
    "address": {
      "type": "Property",
      "value": {
        "addressRegion": "Metropolis",
        "postalCode": "42420",
        "streetAddress": "Lion Street 25",
        "addressLocality": "Cat City"
      }
    },
    "storeName": {
      "type": "Property",
      "value": "Giga Market"
    }
  } ],
  "notifiedAt": "2020-02-09T18:08:02.669000Z",
  "subscriptionId": "urn:ngsi-ld:Subscription:intervalStoreSubscription"
}
```

Figure 10.2.2-2: Notification to time-based subscription

Subscriptions can also be retrieved as shown in Figure 10.2.2-3.

```
GET /ngsi-ld/v1/subscriptions/urn:ngsi-ld:Subscription:intervalStoreSubscription HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
```

Figure 10.2.2-3: Retrieval of a subscription

In addition to the elements that were provided on creation, some status information is provided. In Figure 10.2.2-4, it can be seen when the last notification was sent, when the last successful notification was sent and how many notifications have been sent since the creation of the subscription. In case the sending of a notification failed, the time of the last failed attempt would also be shown.

```
{
  "id": "urn:ngsi-ld:Subscription:intervalStoreSubscription",
  "type": "Subscription",
  "entities": {
    "type": "Store"
  },
  "notification": {
    "attributes": [
      "storeName",
      "address"
    ],
    "endpoint": {
      "accept": "application/json",
      "uri": "http://localhost:8000/notify"
    },
    "format": "normalized",
    "lastNotification": "2020-02-09T18:13:02.676000Z",
    "lastSuccess": "2020-02-09T18:13:02.676000Z",
    "timesSent": 16
  },
  "timeInterval": 60
}
```

Figure 10.2.2-4: Retrieved Subscription

11 Batch Operations

11.1 Overview Batch Operations

Clause 11 describes the Batch operations for the provision and deletion of groups of Entities (i.e. batches). Clause 11.2 describes the operation for creating a group of Entities that do not exist inside the Context Broker. Clause 11.3 describes the operation for updating a group of Entities that exist inside the Context Broker. Clause 11.4 describes the operation for both creating and updating a group of Entities inside the Context Broker. Clause 11.5 describes the operation for deleting a group of NGSI-LD Entities that exist inside the Context Broker.

11.2 Batch Entity Create

In clause 8.2.1 it was shown how to create one single Entity inside a Context Broker. If more suitable, groups of Entities can be provided to Context Brokers, to create more than one Entity in a single request, with the Batch Entity Create operation. Figure 11.2-1 shows the request to create two Entities - which represent a new Store and a new Shelf - in a single request using a link header for providing the `@context`.

```
POST /ngsi-ld/v1/entityOperations/create HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"

[
    {
        "id": "urn:ngsi-ld:Store:004",
        "type": "Store",
        "address": {
            "type": "Property",
            "value": {
                "streetAddress": "Tiger Street 7",
                "addressRegion": "Metropolis",
                "addressLocality": "Cat City",
                "postalCode": "42420"
            }
        },
        "location": {
            "type": "GeoProperty",
            "value": {
                "type": "Point",
                "coordinates": [57.5522, -20.3284]
            }
        },
        "storeName": {
            "type": "Property",
            "value": "6-Stars"
        },
        "contains":{
            "type":"Relationship",
            "object":"urn:ngsi-ld:Shelf:123",
            "observedAt":"2019-09-11T17:11:07Z"
        }
    },
    {
        "id": "urn:ngsi-ld:Shelf:234",
        "type": "Shelf",
        "maxCapacity": {
            "type": "Property",
            "value": 80
        }
    }
]
```

Figure 11.2-1: Store and Shelf Entities creation

If the creation is successful, the response in Figure 11.2-2 with HTTP return code `201` Created is returned.

```
HTTP/1.1 201 Created
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/entityOperations/create
```

Figure 11.2-2: Store and Shelf Entities creation result

When the creation is not successful, i.e. all the Entities were not created, or is partially successful, i.e. at least one Entity was created, the body in the HTTP response shows the reasons of failure for each Entity that was not created. For instance, trying to recreate the Entities in Figure 11.2-1 leads to the result shown in Figure 11.2-3 where it is clearly stated that the Entities could not be created because they already exist in the Context Broker.

```
HTTP/1.1 400 Bad Request
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/entityOperations/create

{
    "success": [],
    "errors": [
        {
            "@id": "urn:ngsi-ld:Store:004",
            "ProblemDetails": "{\n\t\"type\":\"https://uri.etsi.org/ngsi-
ld/errors/AlreadyExists\",\n\t\"title\":\"Already exists.\",\n\t\"detail\":\"urn:ngsi-ld:Store:004
already exists\"\n}"
        },
        {
            "@id": "urn:ngsi-ld:Shelf:234",
            "ProblemDetails": "{\n\t\"type\":\"https://uri.etsi.org/ngsi-
ld/errors/AlreadyExists\",\n\t\"title\":\"Already exists.\",\n\t\"detail\":\"urn:ngsi-ld:Shelf:234
already exists\"\n}"
        }
    ]
}
```

Figure 11.2-3: Store and Shelf Entities creation unsuccessful result

If only some or none of the Entities have been successfully created an HTTP return code `207` Multi-Status is returned.

11.3 Batch Entity Update

The Batch Entity Update operation allows to update more than one NGSI-LD Entity at time with a single HTTP request. Specifically, this operation allows the creation, modification and deletion of Properties and Relationships of **existing** Entities. The major difference between the operations described in clause 8.3 and the Batch Entity Update operation is that the body of the POST HTTP request always contains an array of valid NGSI-LD Entities, i.e. each Entity has the "id" and the "type" fields.

Figure 11.3-1 shows how to perform the Batch Entity Update operation using a link header for providing the `@context`. The successful result is shown in Figure 11.3-2.

```
POST /ngsi-ld/v1/entityOperations/update HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"

[
    {
        "id": "urn:ngsi-ld:Store:004",
        "type": "Store",
        "address": {
            "type": "Property",
            "value": {
                "streetAddress": "Tiger Street 7",
                "addressRegion": "Metropolis",
                "addressLocality": "Cat City",
                "postalCode": "42420"
            },
            "observedAt": "2019-09-12T17:11:07Z"
        }
    },
    {
        "id": "urn:ngsi-ld:Shelf:234",
        "type": "Shelf",
        "maxCapacity": {
            "type": "Property",
            "value": 85
        }
    }
]
```

Figure 11.3-1: Store and Shelf Entities update

```
HTTP/1.1 204 No Content
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/entityOperations/update
```

Figure 11.3-2: Store and Shelf Entities update

The default behaviour of the Batch Entity Update operation is to append new Properties and Relationships and overwrite existing Properties and Relationships for each Entity performing the Update operation described in clause 8.3.2. If desired, the "options=noOverwrite" URL parameter can be provided to deny the modification of existing Properties and Relationships of each Entity, performing the Append operation described in clause 8.3.1.

Figure 11.3-3 shows how to perform the Batch Entity Update operation without providing the "options=noOverwrite" URL parameter using a link header for providing the `@context`. The successful result is shown in Figure 11.3-4.

```
POST /ngsi-ld/v1/entityOperations/update?options=noOverwrite HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"

[
    {
        "id": "urn:ngsi-ld:Store:004",
        "type": "Store",
        "owner": {
            "type": "Property",
            "value": "John Doe"
        }
    },
    {
        "id": "urn:ngsi-ld:Shelf:234",
        "type": "Shelf",
        "minCapacity": {
            "type": "Property",
            "value": "65"
        }
    }
]
```

Figure 11.3-3: Store and Shelf Entities update with noOverwrite

```
HTTP/1.1 204 No Content
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/entityOperations/update
```

Figure 11.3-4: Store and Shelf Entities update with noOverwrite result

If only some or none of the Entities have been successfully updated an HTTP return code `207` Multi-Status is returned.

11.4 Batch Entity Upsert

The Batch Entity Upsert operation can be used to create or update a batch of NGSI-LD Entities with a single HTTP request.

Figure 11.4-1 shows how to perform the Batch Entity Upsert operation using a link header for providing the `@context`. The successful result is shown in Figure 11.4-2.

```
POST /ngsi-ld/v1/entityOperations/upsert HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"

[
    {
        "id": "urn:ngsi-ld:Store:004",
        "type": "Store",
        "address": {
            "type": "Property",
            "value": {
                "streetAddress": "Tiger Street 7",
                "addressRegion": "Metropolis",
                "addressLocality": "Cat City",
                "postalCode": "42420"
            }
        },
        "location": {
            "type": "GeoProperty",
            "value": {
                "type": "Point",
                "coordinates": [57.5522, -20.3484]
            }
        },
        "storeName": {
            "type": "Property",
            "value": "6-Stars"
        },
        "contains":{
            "type":"Relationship",
            "object":"urn:ngsi-ld:Shelf:345",
            "observedAt":"2019-09-11T17:11:07Z"
        }
    },
    {
        "id": "urn:ngsi-ld:Shelf:345",
        "type": "Shelf",
        "minCapacity": {
            "type": "Property",
            "value": "65"
        }
    }
]
```

Figure 11.4-1: Store and Shelf Entities upsert

```
HTTP/1.1 201 Created
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/entityOperations/upsert

[
   "urn:ngsi-ld:Shelf:345"
]
```

Figure 11.4-2: Store and Shelf Entities upsert

When updating, the default behaviour of the Batch Entity Upsert operation is to replace the existing Entities with the one provided in the body of the POST HTTP request. This behaviour can be changed using the "options=update" URL parameter. If given, when updating the Entities are not replaced, but they are just updated using the same logic of the Batch Entity Update operation described in clause 11.3 (default behaviour).

Figure 11.4-3 shows how to perform the Batch Entity Upsert operation providing the "options=update" URL parameter. The successful result is shown in Figure 11.4-4.

```
POST /ngsi-ld/v1/entityOperations/upsert?options=update HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"

[
    {
        "id": "urn:ngsi-ld:Store:004",
        "type": "Store",
        "storeName": {
            "type": "Property",
            "value": "12-Stars"
        }
    },
    {
        "id": "urn:ngsi-ld:Shelf:345",
        "type": "Shelf",
        "minCapacity": {
            "type": "Property",
            "value": "40"
        },
        "maxCapacity": {
            "type": "Property",
            "value": "80"
        }
    }
]
```

Figure 11.4-3: Store and Shelf Entities upsert with options=update

```
HTTP/1.1 204 No Content
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/entityOperations/upsert?options=update
```

Figure 11.4-4: Store and Shelf Entities upsert with options=update

In case of successful results, the returned HTTP response has a code `201` Created if at least one of the Entities in the body of the HTTP request was created, otherwise an HTTP return code `204` No content is returned, i.e. all the Entities provided in the body of the HTTP request already existed in the Context Broker and they were all updated.

If only some or none of the Entities have been successfully created or updated an HTTP return code `207` Multi-Status is returned.

11.5 Batch Entity Delete

The Batch Entity Delete operation can be used to delete a batch of NGSI-LD Entities with a single HTTP request.

Figure 11.5-1 shows how to perform the Batch Entity Delete operation using a link header for providing the `@context`. The successful result is shown in Figure 11.5-2.

```
POST /ngsi-ld/v1/entityOperations/delete HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"

[
    "urn:ngsi-ld:Store:004",
    "urn:ngsi-ld:Shelf:345"
]
```

Figure 11.5-1: Store and Shelf Entities delete

```
HTTP/1.1 204 No Content
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/entityOperations/delete
```

Figure 11.5-2: Store and Shelf Entities delete

If only some or none of the Entities have been successfully deleted an HTTP return code `207` Multi-Status is returned.

12 Query for Available Types and Attributes

12.1 Overview Query for Available Types and Attributes

Clause 12 describes the NGSI-LD Query for Available Types and Attributes operations for the consumption of additional information regarding the Entities and the Attributes stored inside the Context Broker. Clause 12.2 describes the operations for retrieving additional data regarding the available Entity Types in a Context Broker. Clause 12.3 describes the operations for retrieving additional data regarding the available Attributes in a Context Broker.

12.2 Available Entity Type operations

12.2.1 Retrieve Available Entity Types and Retrieve Details of Available Entity Types operations

The "Retrieve Available Entity Types" refers to the process of obtaining a list of all the Entity Types that are available within a Context Broker. This operation allows a context consumer to retrieve information about the types of Entities that can be queried within the system.

To retrieve the list of the available Entity Types, a client can send an HTTP GET request like the one shown in Figure 12.2.1-1.

```
GET /ngsi-ld/v1/types/ HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
```

Figure 12.2.1-1: Retrieve Available Entity Type operation

The output of the request in Figure 12.2.1-1 is the list of all Entity Types available within the Context Broker. It should be similar to the one shown in Figure 12.2.1-2.

```
HTTP/1.1 200 OK
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/types/
{
    "id": "urn:ngsi-ld:EntityTypeList:3aa5a54e-038e-11ee-8591-0a580a82046a",
    "type": "EntityTypeList",
    "typeList": [
        "Shelf",
        "Store"
    ]
}
```

Figure 12.2.1-2: Retrieve Available Entity Type operation result

If needed, it is possible to retrieve a more detailed output thanks to the "Retrieve Details of Available Entity Types" operation. Instead of returning an object with the list of the available Entity Types in the Context Broker, this operation returns an array of objects where each object is the representation of each Entity Type in the Context Broker. Such representation includes the "typeName" and the list of the Attributes that any Entity of that type can have.

To retrieve the details of the available Entity Types, a client can send an HTTP GET request to the same endpoint of "Retrieve Available Entity Types" while also including the URL parameter "details=true", as shown in Figure 12.2.1-3.

```
GET /ngsi-ld/v1/types/?details=true HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
```

Figure 12.2.1-3: Retrieve Details of Available Entity Types operation

An example of the expected output is shown in Figure 12.2.1-4.

```
HTTP/1.1 200 OK
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/types/?details=true
[
    {
        "id": "ngsi-ld:primer/Shelf",
        "type": "EntityType",
        "typeName": "Shelf",
        "attributeNames": [
            "maxCapacity"
        ]
    },
    {
        "id": "ngsi-ld:primer/Store",
        "type": "EntityType",
        "typeName": "Store",
        "attributeNames": [
            "address",
            "contains",
            "storeName",
            "location"
        ]
    }
]
```

Figure 12.2.1-4: Retrieve Details of Available Entity Types operation result

These operations are useful for understanding the structure and contents of the Entity Types stored in a Context Broker, which can help developers and other users understand what kinds of data are available for querying or processing.

12.2.2 Retrieve Available Entity Type Information operation

This operation allows a client to retrieve detailed information of a specified NGSI-LD Entity Type. The detailed representation includes the type, the count of Entity instances, and details about Attributes that existing instances of this Entity Type have, including their name and a list of types the Attribute can have (e.g. Property, Relationship or GeoProperty).

To perform this operation, a client can send an HTTP GET request to the "types/{type}" resource endpoint, where {type} is the name of the Entity Type for which information is being requested as shown in Figure 12.2.2-1.

```
GET /ngsi-ld/v1/types/Store HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
```

Figure 12.2.2-1: NGSI-LD Retrieve Available Entity Type Information operation

An example of the expected output is shown in Figure 12.2.2-2.

```
HTTP/1.1 200 OK
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/types/Store
{
    "id": "ngsi-ld:primer/Store",
    "type": "EntityTypeInformation",
    "typeName": "Store",
    "entityCount": 1,
    "attributeDetails": [
        {
            "id": "https://uri.etsi.org/ngsi-ld/primer/address",
            "type": "Attribute",
            "attributeName": "address",
            "attributeTypes": [
                "Property"
            ]
        },
        {
            "id": "location",
            "type": "Attribute",
            "attributeName": "location",
            "attributeTypes": [
                "GeoProperty"
            ]
        },
        {
            "id": "https://uri.etsi.org/ngsi-ld/primer/storeName",
            "type": "Attribute",
            "attributeName": "storeName",
            "attributeTypes": [
                "Property"
            ]
        },
        {
            "id": "https://uri.etsi.org/ngsi-ld/primer/contains",
            "type": "Attribute",
            "attributeName": "contains",
            "attributeTypes": [
                "Relationship"
            ]
        }
    ]
}
```

Figure 12.2.2-2: NGSI-LD Retrieve Available Entity Type Information operation result

The "Retrieve Available Entity Type Information" operation is useful for understanding the structure and contents of a specific Entity Type stored in a Context Broker, which can help developers and other users understand what kinds of data are available for querying or processing.

12.3 Available Attributes operations

12.3.1 Retrieve Available Attributes and Retrieve Details of Available Attributes

The "Retrieve Available Attributes" is an operation that allows a client to retrieve a list of NGSI-LD Attributes that belong to Entity instances existing within the Context Broker.

To retrieve the list of available Attributes, a client can send an HTTP GET request as shown in Figure 12.3.1-1.

```
GET /ngsi-ld/v1/attributes/ HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
```

Figure 12.3.1-1: Retrieve Available Attributes operation

The output of the request in Figure 12.3.1-1 is a list of all available Attributes within the Context Broker. It should be similar to the one shown in Figure 12.3.1-2.

```
HTTP/1.1 200 OK
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/attributes/
{
    "id": "urn:ngsi-ld:AttributeList:35259",
    "type": "AttributeList",
    "attributeList": [
        "location",
        "address",
        "contains",
        "isContainedIn",
        "maxCapacity",
        "minCapacity",
        "storeName"
    ]
}
```

Figure 12.3.1-2: Retrieve Available Attributes operation result

If needed, it is possible to retrieve a more detailed output thanks to the "Retrieve Details of Available Attributes" operation. Instead of returning an object with the list of the available Attributes in the Context Broker, this operation returns an array of objects where each object is the representation of each Attribute stored in at least one existing Entity in the Context Broker. Such representation includes the Attribute name, the number of Attribute Instances with this Attribute name, the list of all the Attribute types (i.e. Property, Relationship, GeoProperty, etc.) that figures as the type of any Attribute Instances with this name and the list of Entity Types with Attributes with this name.

To retrieve the details of the available Attributes in a Context Broker, a client can send an HTTP GET request to the same endpoint of "Retrieve Available Attributes" while including the URL parameter "details=true" as shown in Figure 12.3.1-3.

```
GET /ngsi-ld/v1/attributes/?details=true HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
```

Figure 12.3.1-3: Retrieve Details of Available Attributes operation

An example of the expected output is shown in Figure 12.3.1-4.

```
HTTP/1.1 200 OK
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/attributes/?details=true
[
    {
        "id": "location",
        "type": "Attribute",
        "attributeCount": 2,
        "attributeTypes": [
            "GeoProperty"
        ],
        "typeNames": [
            "Store"
        ]
    },
    {
        "id": "ngsi-ld:primer/minCapacity",
        "type": "Attribute",
        "attributeCount": 1,
        "attributeTypes": [
            "Property"
        ],
        "typeNames": [
            "Shelf"
        ]
    },
    {
        "id": "ngsi-ld:primer/maxCapacity",
        "type": "Attribute",
        "attributeCount": 1,
        "attributeTypes": [
            "Property"
        ],
        "typeNames": [
            "Shelf"
        ]
    },
    {
        "id": "ngsi-ld:primer/address",
        "type": "Attribute",
        "attributeCount": 1,
        "attributeTypes": [
            "Property"
        ],
        "typeNames": [
            "Store"
        ]
    },
    {
        "id": "ngsi-ld:primer/storeName",
        "type": "Attribute",
        "attributeCount": 1,
        "attributeTypes": [
            "Property"
        ],
        "typeNames": [
            "Store"
        ]
    },
    {
        "id": "ngsi-ld:primer/contains",
        "type": "Attribute",
        "attributeCount": 1,
        "attributeTypes": [
            "Relationship"
        ],
        "typeNames": [
            "Store"
        ]
    }
]
```

Figure 12.3.1-4: Retrieve Details of Available Attributes operation result

Similar to the operations in clause 12.2.1, these operations are useful for understanding the structure and contents of the Attributes stored in a Context Broker, which can help developers and other users understand what kinds of data are available for querying or processing.

12.3.2 Retrieve Available Attribute Information

This operation allows a client to retrieve detailed information about a specified NGSI-LD Attribute. The detailed representation includes the Attribute name, the count of Attribute Instances, a list of types the Attribute can have (e.g. Property, Relationship or GeoProperty) and a list of Entity Types that can have an Attribute with this name.

To perform this operation, a client can send an HTTP GET request to the "attributes/{attrId}" resource endpoint, where {attrId} is the name of the Attribute for which information is being requested. As an example, Figure 12.3.2-1 shows how to retrieve the information for the Attribute with name "ngsi-ld:primer/contains".

```
GET /ngsi-ld/v1/attributes/contains HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>; rel="http://www.w3.org/ns/json-
ld#context"; type="application/ld+json"
```

Figure 12.3.2-1: Retrieve Available Attributes Information operation

Figure 12.3.2-2 shows the expected result.

```
HTTP/1.1 200 OK
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/attributes/contains
{
    "id": "ngsi-ld:primer/contains",
    "type": "Attribute",
    "attributeCount": 1,
    "attributeTypes": [
        "Relationship"
    ],
    "typeNames": [
        "Store"
    ]
}
```

Figure 12.3.2-2: Retrieve Available Attributes Information operation result

Similar to the operation described in clause 12.2.2, the "Retrieve Available Attributes Information" operation is useful for understanding the structure and contents of an Attribute stored in a Context Broker, which can help developers and other users understand what kinds of data are available for querying or processing.

13 Temporal API

13.1 Overview of Temporal API

Clause 13 describes the usage of the Temporal API for the consumption and provision of Temporal Representations of Entities. Clause 13.2 clarifies the motivations behind the Temporal API and provides an exemplary use case.
Clause 13.3 describes the operations to provide Temporal Representation of Entities. Clause 13.4 describes the operations to retrieve and consume Temporal Representations of Entities.

13.2 Motivation and Use Case

The Temporal API increases the expressive possibilities of Entities, allowing the storage of multiple Attribute Instances over time. These Attribute Instances make it possible to capture and represent the alteration over time of the content of the Entities, information that can be used to evaluate their trend/behaviour to support external applications (e.g. Machine Learning).

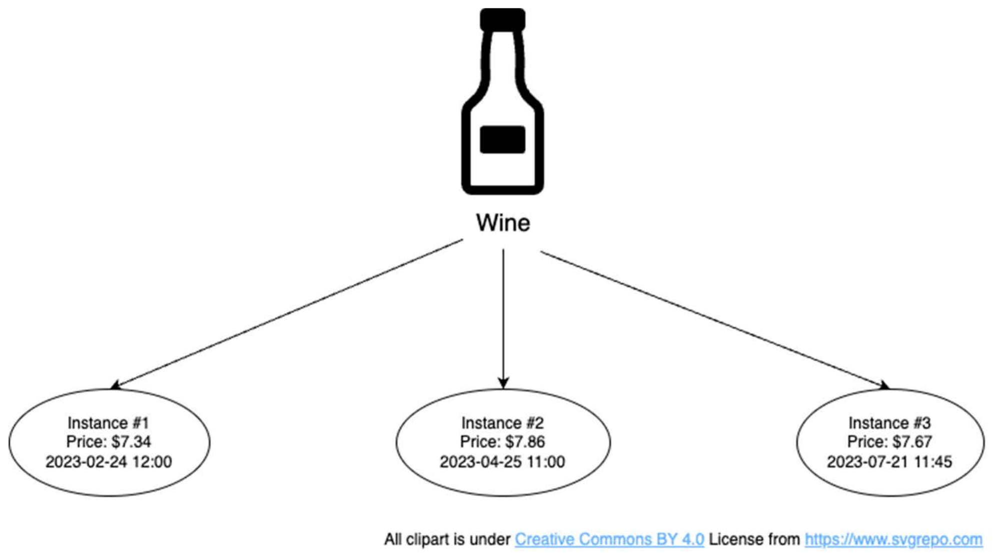


Figure 13.2-1: Temporal Representation of an Entity

Taking the example of the store presented in clause 4, the Temporal API can be used to show the trend of the price of a product (e.g. a bottle of wine), as depicted in Figure 13.2-1. Without the Temporal API, the price is overwritten with each update of the value of the "Price" Attribute. Thanks to the Temporal API, it is possible to keep track of all the previous values of "Price" within the Context Broker, having a dedicated timestamp for each one. These values can be reused to obtain refined information through aggregate queries to the Context Broker. For example, it is possible to obtain the average price of wine bottles, information that can be used by external applications to decide whether to change the supplier or modify the quantity of wine bottles to buy.

13.3 Temporal Information Provision

13.3.1 Create or Update Temporal Representation of an Entity

The endpoint `/temporal/entities/` can be used both to create and update the Temporal Representation of an Entity.

Figure 13.3.1-1 shows the request to **create** the Temporal Representation of a Wine Entity with its Attributes.

```
POST /ngsi-ld/v1/temporal/entities/ HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

{
    "id": "urn:ngsi-ld:Wine:001",
    "type": "Wine",
    "price": {
        "type": "Property",
        "value": 7.83,
        "observedAt": "2021-04-04T11:40:00.000000Z"
    },
    "quantity": {
        "type": "Property",
        "value": 20,
        "observedAt": "2021-04-04T11:40:00.000000Z"
    },
    "putBackOnShelf": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:Shelf:234",
        "observedAt": "2021-04-04T11:40:00.000000Z"
    }
}
```

Figure 13.3.1-1: Create Temporal Representation of an Entity

If the creation is successful, the response in Figure 13.3.1-2 with HTTP return code `201` Created is returned.

```
HTTP/1.1 201 Created
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/temporal/entities/
```

Figure 13.3.1-2: Temporal Representation of Wine Entity creation result

If the Temporal Representation of the Entity already exists, executing the request of Figure 13.3.1-1 would **update** the Temporal Representation. In case of a success, the response in Figure 13.3.1-3 with HTTP return code `204` No Content is returned.

```
HTTP/1.1 204 No Content
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/temporal/entities/
```

Figure 13.3.1-3: Update of the Temporal Representation of Wine Entity result

As one last remark, in Figure 13.3.1-1 the Temporal Property *observedAt* is provided. This is the only one User Generated Temporal Property supported by the Temporal API. The client can decide its value by setting a *DateTime* string (null by default).

13.3.2 Creating/Updating/Deleting Attribute of a Temporal Representation of an Entity

13.3.2.1 Create/Update Attribute in a Temporal Representation of an Entity

The creation of new Attributes or the update of existing Attributes in a Temporal Representation of an Entity, in both cases, leads to the creation of a **new instance** of the Attribute with a dedicated *instanceId*. The *instanceId* is needed to modify/delete the Attribute Instance as discussed in clause 13.3.3. Each Attribute Instance has its own set of Temporal Properties.

Figure 13.3.2.1-1 shows the request to **create** the "brand" Attribute and to **update** the "price" Attribute of the Temporal Representation of the Wine Entity. This operation leads to the addition of a new instance of the "price" Attribute and the creation of the first instance of the "brand" Attribute inside the Temporal Representation of the Wine Entity.

```
POST /ngsi-ld/v1/temporal/entities/urn:ngsi-ld:Wine:001/attrs HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

{
    "price": {
        "type": "Property",
        "value": 7.90,
        "observedAt": "2021-04-04T11:42:00.000000Z"
    },
    "brand": {
        "type": "Property",
        "value": "Corvo",
        "observedAt": "2021-04-04T11:42:00.000000Z"
    }
}
```

Figure 13.3.2.1-1: Create or Update Attribute of Temporal Representation of an Entity

If the creation is successful, the response in Figure 13.3.2.1-2 with HTTP return code `204` No Content is returned.

```
HTTP/1.1 204 No Content
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/temporal/entities/urn:ngsi-ld:Wine:001/attrs
```

Figure 13.3.2.1-2: Create or Update Attribute of Temporal Representation of an Entity result

13.3.2.2 Delete Attribute from a Temporal Representation of an Entity

The Delete Attribute operation is used to delete an Attribute from a Temporal Representation of an Entity. It supports two URL parameters:

- **deleteAll**, if true, it deletes all Attribute Instances (both the default Attribute instance and every instance with a *datasetId*).
- **datasetId**, if set, it deletes only the Attribute Instances with the given *datasetId*. It is ignored if *deleteAll* is true.

The default behaviour of this operation is to **delete all the Attribute Instances** of the default Attribute Instance.

Figure 13.3.2.2-1 shows the request to **delete** the "brand" Attribute of the Temporal Representation of the Wine Entity.

```
DELETE /ngsi-ld/v1/temporal/entities/urn:ngsi-ld:Wine:001/attrs/brand HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json
```

Figure 13.3.2.2-1: Delete Attribute of Temporal Representation of an Entity

If the deletion is successful, the response in Figure 13.3.2.2-2 with HTTP return code `204` No Content is returned.

```
HTTP/1.1 204 No Content
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/temporal/entities/urn:ngsi-ld:Wine:001/attrs/brand
```

Figure 13.3.2.2-2: Delete Attribute of Temporal Representation of an Entity result

13.3.3 Modifying/Deleting Attribute Instances of a Temporal Representation of an Entity

13.3.3.1 Modify Attribute Instance of a Temporal Representation of an Entity

As hinted in clause 13.3.2.1, each Attribute of a Temporal Representation of an Entity has one or more instances. Each Attribute Instance can be identified with the *instanceId*, a unique URI generated randomly by the Context Broker. The *instanceId* can be used to modify a specific instance of an Attribute by **overwriting** the value, every sub-Attribute or the Temporal Property *observedAt*. Successful modification also updates the System Generated Attribute ***modifiedAt*** of the given Attribute Instance. The *instanceId* is returned when retrieving the Temporal Representation of an Entity, as discussed in clause 13.4.

Figure 13.3.3.1-1 shows the request to **modify** the instance "urn:ngsi-ld:29e74a25-65c6-4371-98dc-1a97b3be5ab7" of the "price" Attribute of the Temporal Representation of the Wine Entity.

```
PATCH /ngsi-ld/v1/temporal/entities/urn:ngsi-ld:Wine:001/attrs/price/urn:ngsi-ld:29e74a25-65c6-4371-
98dc-1a97b3be5ab7 HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

{
    "price": {
        "type": "Property",
        "value": 8.10,
        "observedAt": "2021-04-04T11:40:00.000000Z"
    }
}
```

Figure 13.3.3.1-1: Modify Attribute instance of Temporal Representation of an Entity

If the modification is successful, the response in Figure 13.3.3.1-2 with HTTP return code `204` No Content is returned.

```
HTTP/1.1 204 No Content
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/temporal/entities/urn:ngsi-ld:Wine:001/attrs/urn:ngsi-ld:29e74a25-65c6-4371-
98dc-1a97b3be5ab7
```

Figure 13.3.3.1-2: Modify Attribute Instance of Temporal Representation of an Entity result

13.3.3.2 Delete Attribute Instance from a Temporal Representation of an Entity

Similarly to the modification, the *instanceId* can be also used to delete a specific instance of an Attribute.

Figure 13.3.3.2-1 shows the request to **delete** the Attrribute Instance "urn:ngsi-ld:29e74a25-65c6-4371-98dc-1a97b3be5ab7" of the "price" Attribute of the Temporal Representation of the Wine Entity.

```
DELETE /ngsi-ld/v1/temporal/entities/urn:ngsi-ld:Wine:001/attrs/price/urn:ngsi-ld:29e74a25-65c6-
4371-98dc-1a97b3be5ab7 HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json
```

Figure 13.3.3.2-1: Delete Attribute Instance of Temporal Representation of an Entity

If the deletion was successful, the response in Figure 13.3.3.2-2 with HTTP return code `204` No Content is returned.

```
HTTP/1.1 204 No Content
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/temporal/entities/urn:ngsi-ld:Wine:001/attrs/urn:ngsi-ld:29e74a25-65c6-4371-
98dc-1a97b3be5ab7
```

Figure 13.3.3.2-2: Delete Attribute Instance of Temporal Representation of an Entity result

13.3.4 Delete Temporal Representation of an Entity

```
DELETE /ngsi-ld/v1/temporal/entities/urn:ngsi-ld:Wine:001 HTTP/1.1
Host: localhost:9090
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json
```

Figure 13.3.4-1: Delete Temporal Representation of an Entity

Figure 13.3.4-1 shows the HTTP request for deleting the Temporal Representation of the Wine Entity with the id `urn:ngsi-ld:Wine:001`. Figure 13.3.4-2 shows the result in case of a successful deletion.

```
HTTP/1.1 204 No Content
Date: Wed, 04 Apr 2019 11:42:15 GMT
location: /ngsi-ld/v1/temporal/entities/urn:ngsi-ld:Wine:001
```

Figure 13.3.4-2: Delete Temporal Representation of an Entity result

13.4 Temporal Information Consumption

13.4.1 Differences between Query Language and Temporal Query Language

For the majority, the Temporal Query Language is very similar to the Query Language. As such, every query discussed in clause 9 is supported by the Temporal Query Language. Anyway, there are a couple of differences that differentiate the Temporal Query Language from the Query Language.

The first difference between the two query Languages is the output. The Temporal Query Language returns Temporal Representation of Entities, while the Query Language returns Entities. Also, it is not possible to return Temporal Representation of Entities as GeoJSON Features (hence the "geometryProperty" parameter is not supported by the Temporal Query Language).

Furthermore, the Temporal Query Language adds the support to time queries and aggregate queries. It is possible to apply temporal filters (e.g. "return Wine Entities between two dates") using a Temporal Property and it is also possible to obtain the Aggregated Temporal Representation of an Entity (e.g. "return the average price of a Wine Entity").

Finally, similar queries may return different results if executed using the Query Language and the Temporal Query Language as Entities created with the Core API are not influenced by operations on Temporal Representation of Entities with the same ID.

13.4.2 Retrieve Temporal Evolution of an Entity

Figure 13.4.2-1 shows how to retrieve the Temporal Representation of the Wine Entity with the id `urn:ngsi-ld:Wine:001`.

```
GET /ngsi-ld/v1/temporal/entities/urn:ngsi-ld:Wine:001 HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
```

Figure 13.4.2-1: Retrieval of Temporal Representation of the Wine:001 Entity

```
HTTP/1.1 200 OK
Date: Wed, 03 Apr 2019 15:50:09 GMT
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

{
    "id": "urn:ngsi-ld:Wine:001",
    "type": "Wine",
    "brand": {
        "type": "Property",
        "value": "Corvo",
        "instanceId": "urn:ngsi-ld:aa6b1ac2-63cf-44f8-9f63-221cf5097947"
    },
    "putBackOnShelf": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:Shelf:234",
        "instanceId": "urn:ngsi-ld:4c94a3cf-866b-439f-800f-306eb72b6221"
    },
    "quantity": {
        "type": "Property",
        "value": 20,
        "instanceId": "urn:ngsi-ld:10e94cdc-7cd2-4466-aff1-102009568079"
    },
    "price": [
        {
            "type": "Property",
            "value": 7.9,
            "instanceId": "urn:ngsi-ld:20052fee-648a-4924-aa17-eb563a81ad66"
        },
        {
            "type": "Property",
            "value": 7.83,
            "instanceId": "urn:ngsi-ld:708b250b-81ac-41d9-a0bd-a7e9f6d48572"
        }
    ]
}
```

Figure 13.4.2-2: Retrieved result of the Temporal Representation of the Wine:001 Entity

Figure 13.4.2-2 shows the result of the query operation, which contains the Temporal Representation of the Wine:001 Entity.

13.4.3 Retrieve Temporal Evolution of Entities

In addition to the queries listed in clause 9, the Temporal Query Language allows to filter out Temporal Representations of Entities based on Temporal Properties using the **timeproperty** parameter.

Figure 13.4.3-1 shows how to retrieve all the Temporal Representation of Wine Entities having at least one Attribute Instance with a value of *observedAt* **after** the following DateTime: "2021-04-04T11:41:00.000000Z".

```
GET /ngsi-ld/v1/temporal/entities type=Wine&timeproperty=observedAt&timerel=before&timeAt=2021-04-
04T11:41:00.000000Z HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
```

Figure 13.4.3-1: Retrieval of Temporal Evolution of Wine Entities

Figure 13.4.3-2 shows the expected result (after executing the HTTP requests of clauses 13.3.1 and 13.3.2.1).

```
HTTP/1.1 200 OK
Date: Wed, 03 Apr 2019 15:50:09 GMT
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

[
    {
        "id": "urn:ngsi-ld:Wine:001",
        "type": "Wine",
        "putBackOnShelf": {
            "type": "Relationship",
            "object": "urn:ngsi-ld:Shelf:234",
            "instanceId": "urn:ngsi-ld:f7ebb27b-5418-47f1-90ad-f771e11b70d8",
            "observedAt": "2021-04-04T11:40:00.000000Z"
        },
        "quantity": {
            "type": "Property",
            "value": 20,
            "instanceId": "urn:ngsi-ld:24657399-4fab-4c3d-ac20-625ad816006a",
            "observedAt": "2021-04-04T11:40:00.000000Z"
        },
        "price": {
            "type": "Property",
            "value": 7.83,
            "instanceId": "urn:ngsi-ld:e30c8036-2fdd-4bd6-a7a1-77997f07f98c",
            "observedAt": "2021-04-04T11:40:00.000000Z"
        }
    }
]
```

Figure 13.4.3-2: Retrieved result of the Temporal Evolution of Wine Entities

The Temporal Query Language also provides a parameter that can be used to limit the number of instances returned for each Attribute in the result to the last *N* **instances**, ordered by the given Temporal Property (descending order).

NOTE: The **lastN** parameter applies to the instances of an Attribute in the Temporal API for every Temporal Representation of Entities returned in the result, whereas the **limit** parameter can be used to reduce the number of Entities/Temporal Representation of Entities returned in the result.

Figure 13.4.3-3 shows how to use the *lastN* parameter in combination with the Temporal Property *createdAt*.

```
GET / ngsi-ld/v1/temporal/entities?type=Wine&timeproperty=createdAt&timerel=after&timeAt=2021-04-
04T11:40:00.000000Z&lastN=1 HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
```

Figure 13.4.3-3: Usage of lastN parameter

Figure 13.4.3-4 shows the expected result.

```
HTTP/1.1 200 OK
Date: Wed, 03 Apr 2019 15:50:09 GMT
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

[
    {
        "id": "urn:ngsi-ld:Wine:001",
        "type": "Wine",
        "brand": {
            "type": "Property",
            "value": "Corvo",
            "instanceId": "urn:ngsi-ld:90bbcfd2-1f80-4951-a865-edaf2de74396",
            "observedAt": "2021-04-04T11:42:00.000000Z"
        },
        "putBackOnShelf": {
            "type": "Relationship",
            "object": "urn:ngsi-ld:Shelf:234",
            "instanceId": "urn:ngsi-ld:f7ebb27b-5418-47f1-90ad-f771e11b70d8",
            "observedAt": "2021-04-04T11:40:00.000000Z"
        },
        "quantity": {
            "type": "Property",
            "value": 20,
            "instanceId": "urn:ngsi-ld:24657399-4fab-4c3d-ac20-625ad816006a",
            "observedAt": "2021-04-04T11:40:00.000000Z"
        },
        "price": {
            "type": "Property",
            "value": 7.9,
            "instanceId": "urn:ngsi-ld:99cd127b-c500-4174-aa19-39a93879a0ad",
            "observedAt": "2021-04-04T11:42:00.000000Z"
        }
    }
]
```

Figure 13.4.3-4: Usage of lastN parameter result

13.4.4 Aggregate Queries

The Temporal Query Language supports **aggregate** functions. The presence of the Aggregate functions allows the support to special queries that are not possible to be performed with the Query Language: for example, given an Entity, it is possible to compute the sum of all the instances of a numeric Property or it is possible to search for the longest word contained in an instance of a Property which type is *String*.

The supported aggregate functions are the following:

- *totalCount*
- *distinctCount*
- *sum*
- *avg*
- *min*
- *max*
- *stddev*
- *sumsq*

totalCount and *distictCount* are supported for every Property and Relationship. The other ones are always supported for numbers, but they may not be supported for different data types (e.g. it is not possible to make the average of Strings). A more in-depth analysis of each aggregate function and the supported data types can be found in clause 4.5.19.1 of ETSI GS CIM 009 [i.1].

Figure 13.4.4-1 shows how to request the average and maximum price of Wine Entities with the aggregate methods *avg* and *max*.

```
GET / ngsi-ld/v1/temporal/entities ?type=Wine&timerel=after&timeAt=2020-04-
04T11:38:00.000000Z&aggrMethods=max,avg&options=aggregatedValues&attrs=ngsi-ld:primer/price HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
```

Figure 13.4.4-1: Retrieval of the average and maximum value of the price of Wine Entities

Figure 13.4.4-2 shows the expected result.

```
HTTP/1.1 200 OK
Date: Wed, 03 Apr 2019 15:50:09 GMT
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

[
    {
        "id": "urn:ngsi-ld:Wine:001",
        "type": "Wine",
        "price": {
            "type": "Property",
            "max":[
                  [
                     7.9,
                     "2020-04-04T11:40:00.000000000Z",
                     "2023-07-24T10:00:25.574761272Z"
                  ]
            ],
            "avg":[
                  [
                     7.865,
                     "2020-04-04T11:40:00.000000000Z",
                     "2023-07-24T10:00:25.574761272Z"
                  ]
            ]
        }
    }
]
```

Figure 13.4.4-2: Result of the retrieval of the average and maximum value of the price of Wine Entities

The *price* Property contains two new non-reified sub-Properties having the name of the requested aggregated functions (avg and max). Both of them are structured this way:

- The computed value (which depends on the Aggregated Function)
- Starting time of the period
- Ending time of the period

The period is by default the whole lifespan of the Property (i.e. it comprises all the instances), but it can be set to group the instances in specific time-frames (e.g. 1 year, 6 months, etc.). To do that, the optional URL parameter **aggrPeriodDuration** can be set as described in clause 4.5.19.1 of ETSI GS CIM 009 [i.1].

14 Advanced Attribute Types

14.1 Overview Advanced Attribute Types

Clause 14 describes various types of advanced attribute types that can be used to model complex data structures in NGSI-LD. Clause 14.2 illustrates how to use the Relationship datatype for representing 1-to-n unordered relationships. Clause 14.3 introduces the ListRelationship datatype, which can hold an ordered list of Relationship objects. Clause 14.4 contrasts the modelling of 1-to-n Relationships with the modelling using a multi-relationship, i.e. multiple independent Relationships with the same name. Clause 14.5 introduces the ListProperty datatype, which can hold an ordered list of Property values. Clause 14.6 introduces the LanguageProperty datatype, which enables the representation of multilingual text. Clause 14.7 introduces the VocabProperty datatype, whose value is expanded accordingly with the provided @context. Finally, clause 14.8 introduces the JSONProperty datatype, which allows the storage data as "literal" JSON data whose keys are not expanded opposed to the default behaviour of JSON-LD.

14.2 Unordered 1-to-n Relationship

There are instances where it is necessary to associate an NGSI-LD Entity with multiple Entities simultaneously. In the recurring use-case provided in the present document, this could occur if one wanted to link a Store Entity with several Shelf Entities. How can this be achieved? The Relationship type described in clause 6 can be utilized in this scenario with the appropriate considerations. Figure 14.2-1 demonstrates that the object field of the "contains" Relationship can also contain an array of URIs. This type of value is entirely valid in NGSI-LD and corresponds to the definition of an **unordered** list of URIs.

NOTE: The present document assumes that the Entities with identifier `urn:ngsi-ld:Shelf:111`, `urn:ngsi-ld:Shelf:112` and `urn:ngsi-ld:Shelf:113` exist, so they should be created as shown in Figure 8.2.1-3 before proceeding with the request shown in Figure 14.2-1.

```
POST /ngsi-ld/v1/entities/ HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
   "id": "urn:ngsi-ld:Store:005",
   "type": "Store",
   "address": {
        "type": "Property",
        "value": {
            "streetAddress": "Tiger Street 5",
            "addressRegion": "Metropolis",
            "addressLocality": "Cat City",
            "postalCode": "42420"
        }
    },
    "storeName": {
         "type": "Property",
         "value": "Little Shop"
    },
    "contains": {
        "type": "Relationship",
        "object": [
            "urn:ngsi-ld:Shelf:111",
            "urn:ngsi-ld:Shelf:112",
            "urn:ngsi-ld:Shelf:113"
        ]
    }
}
```

Figure 14.2-1: Creation of Store:005 with an unordered 1-to-n Relationship

If the newly created Entity were requested, it would be observed that the "contains" field indeed contains a list of URIs, as intended at the beginning of this clause. It is worth to underline that the list is **unordered**: the Context Broker will not check the order, hence clients cannot expect the original order to be preserved.

14.3 List Relationship

In the previous clause, it was shown how one can use the Relationship datatype for representing unordered 1-to-n relationships. The same structure cannot be used to represent ordered lists of URIs.

Why is the list of URIs unordered? Without delving into excessive technicalities, this behaviour is due to the representation of the "object" field within the Relationship datatype. This field can hold either a single URI or an array of URIs. Unlike standard JSON, where arrays typically preserve order, JSON-LD arrays, without specific configurations, do not maintain a strict order. If not handled, the order may be altered by the JSON-LD pre-processor. Furthermore, the JSON-LD expansion algorithm presents a counter-intuitive behaviour when dealing with arrays containing **only one** element. For a visual representation, refer to Figure 14.3-1.

```
POST /ngsi-ld/v1/entities/ HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
   "id": "urn:ngsi-ld:Store:006",
   "type": "Store",
   "address": {
        "type": "Property",
        "value": {
            "streetAddress": "Tiger Street 5",
            "addressRegion": "Metropolis",
            "addressLocality": "Cat City",
            "postalCode": "42420"
        }
    },
    "storeName": {
         "type": "Property",
         "value": "Little Shop 2"
    },
    "contains": {
         "type": "Relationship",
         "object": ["urn:ngsi-ld:Shelf:111"]
    }
}
```

Figure 14.3-1: Creation of Store:006 with a Relationship containing an array with one element

```
HTTP/1.1 200 OK
Date: Wed, 05 Apr 2019 16:20:11 GMT
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

{
   "id": "urn:ngsi-ld:Store:006",
   "type": "Store",
   "address": {
        "type": "Property",
        "value": {
            "streetAddress": "Tiger Street 5",
            "addressRegion": "Metropolis",
            "addressLocality": "Cat City",
            "postalCode": "42420"
        }
    },
    "storeName": {
         "type": "Property",
         "value": "Little Shop 2"
    },
    "contains": {
         "type": "Relationship",
         "object": "urn:ngsi-ld:Shelf:111"
    }
}
```

Figure 14.3-2: Store:006 retrieved with Relationship "contains" being an URI instead of an array of URIs

When creating the Store:006 Entity using the HTTP POST request provided in Figure 14.3-1, the "contains" Relationship in Store:006 is represented as a URI, instead of an array of URIs (as one may expect), as shown in Figure 14.3-2. This is very inconvenient in specific use cases.

The type ListRelationship solves this issue by forcing the type of the "object" field to be always an array. Figure 14.3-3 shows how to instantiate a new Entity with an Attribute of type ListRelationship.

NOTE 1: The present document assumes that the Entities with identifier `urn:ngsi-ld:Shelf:111`, `urn:ngsi-ld:Shelf:112` and `urn:ngsi-ld:Shelf:113` exist, so they should be created as shown in Figure 8.2.1-3 before proceeding with the request shown in Figure 14.3-3.

```
POST /ngsi-ld/v1/entities/ HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
   "id": "urn:ngsi-ld:Store:007",
   "type": "Store",
   "address": {
        "type": "Property",
        "value": {
            "streetAddress": "Tiger Street 9",
            "addressRegion": "Metropolis",
            "addressLocality": "Cat City",
            "postalCode": "42420"
        }
    },
    "storeName": {
         "type": "Property",
         "value": "News Comics"
    },
    "contains": {
         "type": "ListRelationship",
         "objectList": ["urn:ngsi-ld:Shelf:111", "urn:ngsi-ld:Shelf:112", "urn:ngsi-ld:Shelf:113"]
    }
}
```

Figure 14.3-3: Creation of Store:007 with a ListRelationship containing an ordered array of URIs

If the newly created Entity were requested, it would be observed that the "contains" field indeed contains an ordered list of URIs that is not altered by the JSON-LD pre-processor. This is ideal when the original order has to be kept unchanged.

The type ListRelationship also fixes the issue with array containing one single URI, because it forces the right-hand side of the "objectList" field to be **always** an array. This can be tested by executing the request shown in Figure 14.3-4.

NOTE 2: The present document assumes that the Entities with identifier `urn:ngsi-ld:Store:006` was deleted before proceeding with the request shown in Figure 14.3-4.

```
POST /ngsi-ld/v1/entities/ HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
   "id": "urn:ngsi-ld:Store:008",
   "type": "Store",
   "address": {
        "type": "Property",
        "value": {
            "streetAddress": "Tiger Street 5",
            "addressRegion": "Metropolis",
            "addressLocality": "Cat City",
            "postalCode": "42420"
        }
    },
    "storeName": {
         "type": "Property",
         "value": "Little Library"
    },
    "contains": {
         "type": "ListRelationship",
         "objectList": ["urn:ngsi-ld:Shelf:111"]
    }
}
```

Figure 14.3-4: Creation of Store:008 with a ListRelationship containing an array with one element

```
HTTP/1.1 200 OK
Date: Wed, 05 Apr 2019 16:20:11 GMT
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

{
   "id": "urn:ngsi-ld:Store:008",
   "type": "Store",
   "address": {
        "type": "Property",
        "value": {
            "streetAddress": "Tiger Street 5",
            "addressRegion": "Metropolis",
            "addressLocality": "Cat City",
            "postalCode": "42420"
        }
    },
    "storeName": {
         "type": "Property",
         "value": "Little Library"
    },
    "contains": {
         "type": "ListRelationship",
         "objectList": ["urn:ngsi-ld:Shelf:111"]
    }
}
```

Figure 14.3-5: Store:008 retrieved with Relationship "contains" being an array

In this case, differently from Figure 14.3-2, the retrieval of Store:008 Entity will confirm that the "objectList" field of the "contains" Attribute is still an array, as shown in Figure 14.3-5. This helps making clients simpler: clients do not need to be aware that an array may be decomposed into one single value, avoiding run-time errors.

14.4 Multi-Relationship

Clauses 14.2 and 14.3 have covered 1-to-n Relationships, i.e. Relationships that relate one Entity with a group of *n* Entities. Such Relationships can have meta information modelled as Subproperties and Subrelationships that relates to the whole group, but cannot be individual with respect to an Entity in this group. If the Relationship are rather individual to each Entity separately, theses can be modelled as individual Relationships, leading to the concept of a multi-relationship. To be able to manage each such Relationship, the Relationships have to be identifiable. This is achieved by attaching a *datasetId* as a Property. For each Relationship, there can be one special instance that is considered the *default* and does not need to have a *datasetId* - this corresponds to the Relationship used in previous clauses.

Assuming meta information is to be added for the "contains" Relationship, i.e. a Subproperty *observedAt* indicating since when the shelf is in the store, the above use case can alternatively be modelled as a Multi-Relationship as shown in Figure 14.4-1.

```
POST /ngsi-ld/v1/entities/ HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
   "id": "urn:ngsi-ld:Store:009",
   "type": "Store",
   "address": {
        "type": "Property",
        "value": {
            "streetAddress": "Tiger Street 5",
            "addressRegion": "Metropolis",
            "addressLocality": "Cat City",
            "postalCode": "42420"
        }
    },
    "storeName": {
         "type": "Property",
         "value": "Mega market"
    },
    "contains": [{
         "type": "Relationship",
         "object": "urn:ngsi-ld:Shelf:123",
         "datasetId": "urn:contains:123",
         "observedAt": "2019-09-09T14:09:07Z"
    },{
         "type": "Relationship",
         "object": "urn:ngsi-ld:Shelf:111",
         "datasetId": "urn:contains:111",
         "observedAt": "2025-04-30T14:09:07Z"
    },{
         "type": "Relationship",
         "object": "urn:ngsi-ld:Shelf:112",
         "datasetId": "urn:contains:112",
         "observedAt": "2025-04-30T14:09:07Z"
    },{
         "type": "Relationship",
         "object": "urn:ngsi-ld:Shelf:113",
         "datasetId": "urn:contains:113",
         "observedAt": "2025-04-30T14:09:07Z"
    }]
}
```

Figure 14.4-1: Creation of Store:009 with multiple individual "contains" Relationships

In the example, there is one shelf that has been in the store since 2019 and three shelves that have been added in 2025.

It is also possible to Multi-Relationships of 1-to-n Relationships, i.e. each Relationship refers to a group of Entities.

14.5 List Property

Similarly to Relationship, the "value" field of Property can contain array of values, but as discussed in the previous clause, there are two issues that one may encounter in specific use cases:

- When there is **more** than one element, the order is **not** preserved.
- When there is only **one** element, the right-hand side of "value" is the element itself.

The ListProperty type is the counterpart of the ListRelationship type for Properties. It is used to store arrays, while avoiding the above listed issues.

Figure 14.5-1 shows how ListProperty can be used when creating an Entity having two addresses while keeping the original order intact.

```
POST /ngsi-ld/v1/entities/ HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
   "id": "urn:ngsi-ld:Store:010",
   "type": "Store",
   "address": {
        "type": "ListProperty",
        "valueList": [
            {
              "streetAddress": "Tiger Street 10",
              "addressRegion": "Metropolis",
              "addressLocality": "Cat City",
              "postalCode": "42420"
            },
            {
              "streetAddress": "Tiger Street 11",
              "addressRegion": "Metropolis",
              "addressLocality": "Cat City",
              "postalCode": "42420"
            }
        ]
    }
}
```

Figure 14.5-1: Creation of Store:010 with a ListProperty

As hinted previously for "objectList", the "valueList" field in a ListProperty is always an array which keeps the original logic. This helps simplifying the logic of clients since they do not need to be aware that an array could be decomposed into a single value under specific circumstances.

14.6 Language Property

LanguageProperty is specifically used to represent the translation of a string literal in many languages. A LanguageProperty contains a field called "languageMap" which is a map where the keys are strings representing IETF RFC 5646 [i.7] language codes and values are strings (the actual translation for the given language) or array of strings (if more than one translation is given for the same term).

Figure 14.6-1 shows how to use LanguageProperty when creating a new Wine Entity with a description translated in different languages.

```
POST /ngsi-ld/v1/entities/ HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
   "id": "urn:ngsi-ld:Wine:abc",
   "type": "Wine",
   "description": {
        "type": "LanguageMap",
        "languageMap": {
            "en": "Elegant aromas, balanced structure, rich flavors, smooth finish, timeless
character.",
            "fr": "Arômes élégants, structure équilibrée, saveurs riches, finale douce, caractère
intemporel.",
            "de": "Elegante Aromen, ausgewogene Struktur, reiche Geschmacksnoten, sanftes Finish,
zeitloser Charakter.",
            "es": "Aromas elegantes, estructura equilibrada, sabores intensos, final suave, carácter
único.",
            "it": "Aromi eleganti, struttura equilibrata, sapori intensi, finale morbido, carattere
unico."
        }
    }
}
```

Figure 14.6-1: Creation of Wine:abc with the translation of the description

Similar to other Attributes, it is possible to query an Entity while targeting the value of a LanguageMap. Figure 14.6-2 shows an example which illustrates how to perform a query with LanguageMaps.

```
GET /ngsi-ld/v1/entities?q=description[en]="Elegant aromas, balanced structure, rich flavors, smooth
finish, timeless character." HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
```

Figure 14.6-2: Query of an Entity by the translated value contained in a LanguageMap

The square brackets ([ ]) are used to target the appropriate language within the languageMap object contained in a LanguageMap. The string in the brackets can be either:

- A string identifying the specific language to match (e.g. en, fr, etc.).
- The wildcard symbol * when it is intended to match against all languages.

An Entity can also be retrieved for one specific language, i.e. instead of each Language Property, a Property with the value being in the requested language is returned. For this the parameter lang has to be provided (e.g. lang=en) as shown in Figure 14.6-3. The result of the Entity retrieval is shown in Figure 14.6-4.

```
GET /ngsi-ld/v1/entities? /ngsi-ld/v1/entities/urn:ngsi-ld:Wine:abc?lang=en HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
```

Figure 14.6-3: Retrieve Entity in one specific language

```
HTTP/1.1 200 OK
Date: Wed, 06 Apr 2025 15:47:11 GMT
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
Content-Type: application/json

{
  "id": "urn:ngsi-ld:Wine:abc",
  "type": "Wine",
  "description": {
    "type": "Property",
    "lang": "en",
    "value": "Elegant aromas, balanced structure, rich flavors, smooth finish, timeless character."
  }
}
```

Figure 14.6-4: Result of Entity retrieved in one specific language

14.7 JSON Property

The JsonProperty type allows storing a JSON (or array of JSONs) while assuring that its keys are not type coerced (and expanded). This is particularly useful when the value is a literal JSON and/or if the JSON contains ambiguous terms utilized within a NGSI-LD document (such as id, type).

Figure 14.7-1 shows how to use JsonProperty when creating a new Wine Entity which includes the id of the last order.

```
POST /ngsi-ld/v1/entities/ HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
   "id": "urn:ngsi-ld:Wine:bcd",
   "type": "Wine",
   "brand": {
        "type": "Property",
        "value": "Corvo"
   },
   "quantity": {
        "type": "Property",
        "value": "100",
        "lastOrder": {
            "type": "JsonProperty",
            "json": {
                "id": "1234abcd",
                "paymentDueDate": "2019-09-09T12:09:07Z"
            }
        }
   }
}
```

Figure 14.7-1: Creation of Wine:bcd with the id of the last order

The two fields included in the JsonProperty, "id" and "paymentDueDate", will not be expanded into IRIs. In this case, this is desirable since the goal is to represent the instance of a concept (Order) that is not mapped into an NGSI-LD Entity (thus, one cannot use a Relationship to link the Transaction to Wine:bcd).The JsonProperty datatype is also very conveniently bundled with a dedicated behaviour with the NGSI-LD Query Language that allows accessing the fields within the "json" field when querying. As an example, Figure 14.7-2 shows how to leverage the Query Language to look for an Entity having a JsonProperty with a field called "id" and the value "1234abcd".

```
GET /ngsi-ld/v1/entities?q=quantity.lastOrder[id]="1234abcd"&jsonKeys=quantity.lastOrder HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
```

Figure 14.7-2: Retrieval of an Entity by the id value contained in a JsonProperty

The figure shows two facts to keep in mind when working with JsonProperties:

- The square brackets ([ ]) are used to target the appropriate field within the JSON object contained in a JsonProperty. The string in the brackets should be the key in the "json" field to target.
- The "jsonKeys" URL parameter is used to prevent the JSON-LD expansion logic to happen in an Attribute prior to the execution of the query. This is necessary when working with JsonProperties.

14.8 Vocab Property

The VocabProperty type allows storing a string or an array of strings that are type coerced into an IRI or an array of IRIs. The expansion depends on the @context applied during the JSON-LD expansion process, and it follows the same logic applied to other fields (such as "type"). This is particularly useful when one wants to take advantage of the JSON-LD type coercion. For instance, considering the example shown in the previous clause, this could be used to represent the type of the value of "lastOrder". Figure 14.8-1 shows how to use VocabProperty to achieve what said.

```
POST /ngsi-ld/v1/entities/ HTTP/1.1
Host: localhost:9090
Content-Type: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"

{
   "id": "urn:ngsi-ld:Wine:cde",
   "type": "Wine",
   "brand": {
        "type": "Property",
        "value": "Corvo"
   },
   "quantity": {
        "type": "Property",
        "value": "100",
        "lastOrder": {
            "type": "JsonProperty",
            "json": {
                "id": "1234abcd",
                "paymentDueDate": "2019-09-09T12:09:07Z"
            },
            "jsonType": {
                "type": "VocabProperty",
                "vocab": "schema:Order"
            }
        }
   }
}
```

Figure 14.8-1: Creation of Wine:cde with the Transaction type as a VocabProperty

Similar to LanguageProperty and JsonProperty, the VocabProperty allows for more sophisticated queries to happen. Specifically, VocabProperties allows leveraging the JSON-LD expansion behaviour and match against long names (similar to what happens with queries by type). Figure 14.8-2 shows an example of how to query VocabProperties.

```
GET /ngsi-
ld/v1/entities?q=quantity.lastOrder.jsonType=schema:Order&expandValues=quantity.lastOrder.jsonType
HTTP/1.1
Host: localhost:9090
Accept: application/json
Link: <https://uri.etsi.org/ngsi-ld/primer/store-context.jsonld>;rel="http://www.w3.org/ns/json-
ld#context";type="application/ld+json"
```

Figure 14.8-2: Retrieval of an Entity by the expanded value of its VocabProperty

The figure shows two facts to keep in mind when working with VocabProperties:

- As for types, the target value is an IRI that can either be a short name (which will be later expanded) or a long name.

- The "expandValues" URL parameter is used to force the JSON-LD expansion logic to happen within an Attributes prior the execution of the query. This is necessary to correctly match the content of the "vocab" field of a VocabProperty with the IRI supplied in the request.

Annex A:
Change history

Date	Version	Information about changes
February 2024	V1.2.2	Created ToC
May 2025	V1.2.3	Added content of clause 14 on Advanced Attribute Types
May 2025	V1.2.4	Corrections and additions of requests in clause 14
June 2025	V1.2.5	Some minor fixes, added terms JSONProperty, ListProperty and ListRelationship

History

Document history		
V1.1.1	March 2020	Publication
V1.2.1	October 2023	Publication
V1.3.1	August 2025	Publication