CEF Context Broker
T06 - Maintenance of technical documentation and of technical handbooks

Version 20200318

Orion-LD
Developer Guide

18/03/2020

Table of Contents

Document characteristics

Property	Value
Release date	18/03/2020
Status:	Final version
Version:	20200318
Authors:	Ken Gunnar Zangelin
Reviewed by:	Stefano De Panfilis
Approved by:	

Document history

Version	Description	Date
20200318	Document submitted for review	18/03/2020

1 Executive Summary

The present document is the Orion-LD Developer Guide as per today version of software and documentation. Namely these guidelines describe the current Orion-LD Alpha Release 1.

For this reason we kindly ask the reader to check for latest updates on the GitHub Orion-LD official repository at https://github.com/FIWARE/context.Orion-LD.

2 Introduction

Welcome to the Developer Guide of Orion-LD, the NGSI-LD context broker!

The Orion-LD context broker doesn't have a GUI and the only way to contact the broker is programatically, be it a shell-script, a javascript program, a python program or whatever. Well, apart from using *Postman* and similar tools, of course.

This guide will *not* bother about what program language is used, as that is completely up to the programmer. The complete requests to be sent to Orion-LD will of course be presented, but in a neutral format, like this:

```
Content-Type: XXX
Accept: XXX
Link: XXX
GET /ngsi-ld/v1/xxx?p1&P2
```

I.e. first the HTTP headers (in this case: Content-Type, Accept, and Link), and then the Verb followed by the URL, including any URI parameters.

It is then up to the reader to "translate" this to a correct call in their respective programming language.

Strongly recommended reading before continuing with this document:

- *NGSI-LD API v1.1.1*
- *Guide to NGSI-LD entities and attributes*
- *Guide to the context*
- *Quick Start Guide*

Also, to be able to play with your own Orion-LD broker while reading this document definitely helps to understand everything better. If you don't have a running Orion-LD to play with, please follow the instructions in the *Orion-LD Installation Guide*.

This document will go into more detail on pretty much everything about the Orion-LD context broker:

- ProblemDetails to describe errors
- Contexts
- Creation of Entities
- Modification of Entities and Attributes
- Deletion of Entities and Attributes
- Forwarding of Creation/Modification of Entities/Attributes
- System Attributes
- Pagination
- Querying for Entities
- Retrieval of a Specific Entity
- Forwarding of Query Requests

- Creation of Subscriptions
- Modification of a Subscription
- Querying for Subscriptions
- Retrieval of a Specific Subscription
- Deletion of Subscription
- Notifications
- Creation of Registrations
- Modification of Registrations
- Querying for Registrations
- Retrieval of a Specific Registration
- Deletion of Registrations
- Subscription to Registrations
- Temporal Representation
- Geolocation
- Query Filter.

3 ProblemDetails to describe errors

ProblemDetails is a standard way of specifying errors in HTTP API responses and the NGSi-LD specifies this to be used in a variety of responses. *Here* is a tutorial on *ProblemDetails* and *here* is the RFC.

Orion-LD uses part of *ProblemDetails*, namely: * type * title * detail * status (sometimes)

4 Contexts

Contexts is a way to use shortnames instead of longnames in the payload and in URI parameters, for attribute names and entity types. It is also a way for the user to use its own aliases (shortnames) for attribute names and entity types.

The context is thoroughly explained in the *guide to the Context*.

4.1 In the Request

If a context is desired for a service, it can be put either in the payload, as part of the entity/subscription/registration, like a "special attribute", or in the HTTP header "Link". Especially the services that have no request payload data, like GET services will have to use the "Link" HTTP header.

If in the Link header, its syntax is a bit complex:

```
Link:  <URL-to-context>;  rel="http://www.w3.org/ns/json-ld#context";
 type="application/ld+json"
```

You need to replace `URL-to-context` with the URL of your context. There can only be a single context when passing it via HTTP header. Just a string is allowed.

If instead the context is put in the payload, the context can be of three different JSON types:

- String (the value is a URL to the context)
- Array (each item in the array is either a string or an object - contexts on their own)
- Object (a key-value list - an "inline" context!).

4.2 In the Response

The `Accept` HTTP header tells the broker where to put the context in the response. If `Accept: application/json` then the context is put in the Link HTTP header of the response, if `Accept: application/ld+json`, then the context is put in the payload data.

As the context in the response will *always* be the very same context of the request, be careful with services that return arrays of entities/subscriptions/registrations. If application/ld+json is used for such a service, then the context will be copied in each and every item (entity/subscription/registration) is the response payload data. Just sayin'.

5 Creation of Entities

Entities can be created in three different ways:

- `POST /ngsi-ld/v1/entities` (to create a single entity)
- `POST /ngsi-ld/v1/entityOperations/create` (to create more than one entity in one go)
- `POST /ngsi-ld/v1/entityOperations/upsert` (to both create and modify entities with one single request)

5.1 POST /ngsi-ld/v1/entities

The service `POST /ngsi-ld/v1/entities` is used for creation of a single entity.

5.1.1 Request Payload Data

The payload data of the request `POST /ngsi-ld/v1/entities` looks like this:

```
{
  "id": "entity-id",              # Mandatory - must be a URI
  "type": "entity-type",          # Mandatory - will be expanded if not a URI
  "location": {},                 # Optional
  "observationSpace": {},         # Optional
  "operationSpace": {},           # Optional
  "property1": {},                # Optional
  "property2": {},                # Optional
  ...
  "propertyN": {},                # Optional
  "relationship1" {},             # Optional
  "relationship2" {},             # Optional
```

```
  ...
  "relationshipN" {},             # Optional
  "@context": "" | [] | {}        # Optional
}
```

The syntax for the attributes is described in the *introduction to NGSI-LD entities and attributes*

5.1.2 Request URI Parameters

As all the information of the entity to create resides in the payload data, there are no URI parameters for this request.

5.1.3 Response HTTP Status Code

- 201 Created - if everything works, and no payload data is present.
- 400 Bad Request - for a payload data with an invalid JSON syntax or invalid JSON types for the fields in the payload data.
- 409 Conflict - the entity already exists.
- 422 Unprocessable Entity - operation not available

5.1.4 Response HTTP Headers

- Location - to inform the creator of where the created entity resides.
- Link - to echo back to the creator the context that was used during modification of the entities.

About `Link`, the issuer of the request already knows the context that was used, as he/she provided it! However, a gateway between the issuer and the broker may need to have this information as well, that's why the context is echoed back.

Below, a typical response to a `POST /ngsi-ld/v1/entities` request:

```
HTTP/1.1 201 Created
Content-Length: 0
Link:
<https://fiware.github.io/NGSI-LD_TestSuite/ldContext/testContext.jsonld>;
  rel="http://www.w3.org/ns/json-ld#context"; type="application/ld+json"
Location: /ngsi-ld/v1/entities/http://a.b.c/entity/E6
Date: xxx
```

As you can see, this response was for a successful request to create an entity with the id `http://a.b.c/entity/E6`, and the context used when creating the entity was `https://fiware.github.io/NGSI-LD_TestSuite/ldContext/testContext.jsonld`.

5.1.5 Response Payload Data

In case of success, there is no payload data in the response. In case of an error, a `ProblemDetails` in the payload data describes the error.

An example of a response payload data for an invalid JSON syntax in the request payload data:

```
HTTP/1.1 400 Bad Request
Content-Length: 150
Content-Type: application/json
Date: REGEX(.*)

{
     "detail": "JSON Parse Error: expecting comma or end of object",
     "title": "JSON Parse Error",
     "type": "https://uri.etsi.org/ngsi-ld/errors/InvalidRequest"
}
```

5.1.6 Pointers to the ETSI NGSI-LD documentation

The latest version (1.2.1) of the NGSI-LD API definition (as of December 2019) is found *here*. Some chapters of interest in said document for `POST /ngsi-ld/v1/entities` are:

- 5.2.4 - The NGSI-LD Entity Data Type
- 5.2.5 - The NGSI-LD Property Data Type
- 5.2.6 - The NGSI-LD Relationship Data Type
- 5.2.7 - The NGSI-LD GeoProperty Data Type
- 5.6.1 - Create Entity
- 6.4.3.1 - POST /ngsi-ld/v1/entities

5.2 POST /ngsi-ld/v1/entityOperations/create

The service `POST /ngsi-ld/v1/entityOperations/create` is used to create more than one entity in a single request. If any of the entities in the incoming payload data already exist, the broker will flag that entity with an error in the outgoing payload data.

5.2.1 Request Payload Data

The payload data of the request `POST /ngsi-ld/v1/entityOperations/create` is an JSON Array of entities:

```
[
  {
     <Entity 0>
  },
  {
     <Entity 1>
```

```
  },
  ...
  {
      <Entity N>
  },
]
```

The syntax of an entity is fully described in the previous chapter (about `POST /ngsi-ld/v1/entities`).

5.2.2 Request URI Parameters

There are no URI parameters for this request. All necessary information resides in the request payload data.

5.2.3 Response HTTP Status Code

201 Created - all entities were successfully created. No response payload data supplied. 207 Multi Status - some entities were successfully created, others weren't. Details of each error in the response payload data. 400 Bad Request - none of the entities in the request payload data were created. Details of each error in the response payload data.

5.2.4 Response HTTP Headers

- Link - to echo back to the creator the context that was used during modification of the entities.

5.2.5 Response Payload Data

In case of All OK, a `204 No Content` is returned and no payload data. In case of `400 Bad Request` (a JSON parse error, or similar errors), the response payload data is the typical {type, title and detail }. E.g.:

```
{
      "detail": "JSON Parse Error: expecting comma or end of object",
      "title": "JSON Parse Error",
      "type": "https://uri.etsi.org/ngsi-ld/errors/InvalidRequest"
}
```

If there is a mix of results (some entities are OK others aren't, then the response payload contains two arrays:

- success - a string array containing the Entity ID of the entities that were successfully created
- errors - an error of objects describing the error details for each entity that weren't created.

Example response payload data for a request to create two entities, urn:ngsi-ld:Vehicle:302 (which worked), and ABC_123456 (which did not work):

```
{
      "errors": [
      {
            "entityId": "ABC_123456",
            "error": {
                  "detail": "ABC_123456",
                  "status": 400,
                  "title": "Not a URI",
                  "type":
"https://uri.etsi.org/ngsi-ld/errors/BadRequestData"
            }
      }
      ],
      "success": [
      "urn:ngsi-ld:Vehicle:302"
      ]
}
```

The fields *"errors"* and *"success"* are *always* present, even if empty arrays.

5.2.6 Pointers to the ETSI NGSI-LD documentation

- 5.2.5 - The NGSI-LD Property Data Type
- 5.2.6 - The NGSI-LD Relationship Data Type
- 5.2.7 - The NGSI-LD GeoProperty Data Type
- 5.2.16 - BatchOperationResult
- 5.2.17 - BatchEntityError
- 5.6.7 - Batch Entity Creation
- 6.14 - POST /ngsi-ld/v1/entityOperations/create

5.3 POST /ngsi-ld/v1/entityOperations/upsert

The service `POST /ngsi-ld/v1/entityOperations/upsert` is used to both create and update entities, in a single request. Those entities that already exist are updated. Those that don't exist are created. In case of error, the details of the errors are specified per entity in the outgoing payload data.

5.3.1 Request Payload Data

Exactly like `POST /ngsi-ld/v1/entityOperations/create`.

5.3.2 Request URI Parameters

- options=update - existing entity content shall be updated and not replaced. The default mode is to *replace* the entire already existing entity with the corresponding entity in the payload data.

5.3.3 Response HTTP Status Code

201 Created - all entities were successfully created/updated. No response payload data supplied. 207 Multi Status - some entities were successfully created/updated, others weren't. Details of each error in the response payload data. 400 Bad Request - none of the entities in the request payload data were created/updated. Details of each error in the response payload data.

5.3.4 Response HTTP Headers

- Link - to echo back to the creator the context that was used during modification of the entities.

5.3.5 Response Payload Data

See the corresponding section for `POST /ngsi-ld/v1/entityOperations/create`.

5.3.6 Pointers to the ETSI NGSI-LD documentation

- 5.2.5 - The NGSI-LD Property Data Type
- 5.2.6 - The NGSI-LD Relationship Data Type
- 5.2.7 - The NGSI-LD GeoProperty Data Type
- 5.2.16 - BatchOperationResult
- 5.2.17 - BatchEntityError
- 5.6.8 - Batch Entity Creation or Update (Upsert)
- 6.15 - POST /ngsi-ld/v1/entityOperations/upsert

6 Modification of Entities and Attributes

There are a number of services for modification of entities and attributes:

- POST /ngsi-ld/v1/entities/{entityId}/attrs
- PATCH /ngsi-ld/v1/entities/{entityId}/attrs
- PATCH /ngsi-ld/v1/entities/{entityId}/attrs/{attrId}
- POST /ngsi-ld/v1/entityOperations/upsert
- POST /ngsi-ld/v1/entityOperations/update

6.1 Contexts

See 'Creation of Entities'.

6.2 POST /ngsi-ld/v1/entities/{entityId}/attrs

The service `POST /ngsi-ld/v1/entities/{entityId}/attrs` lets you append attributes to an entity. A URI parameter (`?options=noOverwrite`) tells the broker to *not overwrite* any already existing attributes and instead report those already existing attributes as erroneous in the response payload data.

6.2.1 Request Payload Data

The payload data for this service is a JSON object with attributes. Remove the "id" and "type" from the payload of a complete entity and there you have it:

```
{
  "location": {},                 # Optional
  "observationSpace": {},         # Optional
  "operationSpace": {},           # Optional
  "property1": {},                # Optional
  "property2": {},                # Optional
  ...
  "propertyN": {},                # Optional
  "relationship1" {},             # Optional
  "relationship2" {},             # Optional
  ...
  "relationshipN" {},             # Optional
  "@context": "" | [] | {}        # Optional
}
```

The syntax for attributes is described in the *introduction to NGSI-LD entities and attributes*

6.2.2 Request URI Parameters

- options=noOverwrite - to ask the broker to *not overwrite* any already existing attribute

6.2.3 Response HTTP Status Code

- 204 No Content - if all went well
- 207 Multi-Status - if partial success
- 400 Bad Request - in case of JSON parse error or similar (invalid entity id in the URL path)
- 404 Not Found - if the entity id of the URL path doesn't get a hit for an entity in the database

6.2.4 Response HTTP Headers

No HTTP headers relevant to NGSI-LD are present in the response.

6.2.5 Response Payload Data

If all went well, no payload data is returned, just the `204 No Content`. On partially successful operation, a `207 Multi-Status` is returned and a payload data that contains two arrays (similar to the response payload data of the Batch operations):

```
{
  "updated": [ "attr-name-1", "attr-name-2", ... "attr-name-N ],
  "unchanged": [
```

```
      {
      "attributeName": "attr-name-3",
      "reason": "error reason for not having changed the attribute"
      },
      {
      "attributeName": "attr-name-4",
      "reason": "error reason for not having changed the attribute"
      }
  ]
}
```

For 400 and 404, the typical { type, title, detail } error object (ProblemDetails) is returned as payload data.

6.2.6 Pointers to the ETSI NGSI-LD documentation

- 5.2.5 - The NGSI-LD Property Data Type
- 5.2.6 - The NGSI-LD Relationship Data Type
- 5.2.7 - The NGSI-LD GeoProperty Data Type
- 5.6.3 - Append Entity Attributes
- 6.6.3.1 - POST /ngsi-ld/v1/entities/{entityId}/attrs

6.3 PATCH /ngsi-ld/v1/entities/{entityId}/attrs

The service `PATCH /ngsi-ld/v1/entities/{entityId}/attrs` is used to modify more than one attribute in a single shot. Only already existing attributes are modified. The modification is done by replacing the "old" attribute with the attribute from the incoming payload. Non-existing attributes, i.e. attributes present in the payload but not in the entity to be patched, are ignored.

6.3.1 Request Payload Data

Just like POST for the same resource, the payload data is a JSON object with attributes:

```
{
  "location": {},                 # Optional
  "observationSpace": {},         # Optional
  "operationSpace": {},           # Optional
  "property1": {},                # Optional
  "property2": {},                # Optional
  ...
  "propertyN": {},                # Optional
  "relationship1" {},             # Optional
  "relationship2" {},             # Optional
  ...
  "relationshipN" {},             # Optional
  "@context": "" | [] | {}        # Optional
```

```
}
```

The syntax for attributes is described in the *introduction to NGSI-LD entities and attributes*

6.3.2 Request URI Parameters

There are no URI parameters for this request. All necessary information resides in the request payload data.

6.3.3 Response HTTP Status Code

Just like `POST` for the same resource.

6.3.4 Response HTTP Headers

No HTTP headers relevant to NGSI-LD are present in the response.

6.3.5 Response Payload Data

Just like `POST` for the same resource.

6.3.6 Pointers to the ETSI NGSI-LD documentation

- 5.2.5 - The NGSI-LD Property Data Type
- 5.2.6 - The NGSI-LD Relationship Data Type
- 5.2.7 - The NGSI-LD GeoProperty Data Type
- 5.6.2 - Update Entity Attributes
- 6.6.3.2 - PATCH /ngsi-ld/v1/entities/{entityId}/attrs

6.4 PATCH /ngsi-ld/v1/entities/{entityId}/attrs/{attrId}

This operation allows performing a partial update on an attribute of an entity. A partial update only changes the fields provided in the payload data, leaving the rest of the attribute unaffected.

6.4.1 Request Payload Data

The payload data for this service is a fragment of an attribute. E.g., to change the value to 45, and add a sub-property P11, the payload could look something like this:

```
{
  "value": 45,
  "P11": {
    "type": "Property",
    "value": "p"
 }
}
```

6.4.2 Request URI Parameters

There are no URI parameters for this request. All necessary information resides in the request payload data.

6.4.3 Response HTTP Status Code

- 204 No Content - if all is good and well
- 400 Bad Request - if the request or its content is somehow incorrect
- 404 Not Found - if the entity or the attribute specified in the URL does not exist

6.4.4 Response HTTP Headers

No HTTP headers relevant to NGSI-LD are present in the response.

6.4.5 Response Payload Data

If all went well, no payload data is returned, just the `204 No Content`. For 400 and 404, the typical { type, title, detail } (ProblemDetails) error object is returned as payload data.

6.4.6 Pointers to the ETSI NGSI-LD documentation

- 5.2.5 - The NGSI-LD Property Data Type
- 5.2.6 - The NGSI-LD Relationship Data Type
- 5.2.7 - The NGSI-LD GeoProperty Data Type
- 5.6.4 - Partial Attribute update
- 6.7.3.1 - PATCH

6.5 POST /ngsi-ld/v1/entityOperations/upsert

Already documented under "Creation of Entities".

6.6 POST /ngsi-ld/v1/entityOperations/update

NOT IMPLEMENTED IN ALPHA RELEASE 1

7 Deletion of Entities and Attributes

There are three services for deletion of entities and attributes:

- POST /ngsi-ld/v1/entityOperations/delete
- DELETE /ngsi-ld/v1/entities/{entityId}
- DELETE /ngsi-ld/v1/entities/{entityId}/attrs/{attrId}

The first service allows to delete a number of entities in one go, while the second service deletes one single entity and the last service is for deletion of a single attribute of a given entity.

7.1 POST /ngsi-ld/v1/entityOperations/delete

The "Batch Entity Delete" service allows for deletion of a number of entities in a single request.

7.1.1 Request Payload Data

The payload data of this service is a JSON Array of strings, each string being an Entity-ID:

```
[
  "entity-id 1",
  "entity-id 2",
  ...
  "entity-id N"
]
```

7.1.2 Request URI Parameters

This service has no URI parameters.

7.1.3 Response HTTP Status Code

- 204 No Content - if all is good and well
- 207 Multi-Status - if partial success
- 400 Bad Request - if the request or its content is somehow incorrect, e.g. JSON parse error

7.1.4 Response HTTP Headers

No HTTP headers relevant to NGSI-LD are present in the response.

7.1.5 Response Payload Data

See the corresponding section for `POST /ngsi-ld/v1/entityOperations/create`.

7.1.6 Pointers to the ETSI NGSI-LD documentation

- 5.2.16 - BatchOperationResult
- 5.2.17 - BatchEntityError
- 5.6.10 - Batch Entity Delete
- 6.17.3.1 - POST /ngsi-ld/v1/entityOperations/delete

7.2 DELETE /ngsi-ld/v1/entities/{entityId}

To delete a single entity, without any payload data, the service `DELETE /ngsi-ld/v1/entities/{entityId}`. Not much to say, if the entity with id `entityId` is found, it is deleted.

7.2.1 Request Payload Data

There is no payload data for this service.

7.2.2 Request URI Parameters

This service has no URI Parameters.

7.2.3 Response HTTP Status Code

- 200 OK - if all OK
- 400 Bad Request - if the entity id of the URL PATH is not a valid URI
- 404 Not Found - if the entity does not exist

7.2.4 Response HTTP Headers

No relevant HTTP headers are present in the response.

7.2.5 Response Payload Data

For 400 and 404, the typical { type, title, detail } error object (ProblemDetails) is returned as payload data.

7.2.6 Pointers to the ETSI NGSI-LD documentation

- 5.6.6 - Delete Entity
- 6.5.3.2 - DELETE /ngsi-ld/v1/entities/{entityId}

7.3 DELETE /ngsi-ld/v1/entities/{entityId}/attrs/{attrId}

This service allows a user to delete a single attribute of a given entity.

7.3.1 Request Payload Data

There is no payload data for this service.

7.3.2 Request URI Parameters

This service has no URI Parameters.

7.3.3 Response HTTP Status Code

- 204 No Content - if all is OK
- 400 Bad Request - in case of invalid entity id in the URL path
- 404 Not Found - if the entity id of the URL path doesn't get a hit for an entity in the database

7.3.4 Response HTTP Headers

No relevant HTTP headers are present in the response.

7.3.5 Response Payload Data

For 400 and 404, the typical { type, title, detail } error object (ProblemDetails) is returned as payload data.

7.3.6 Pointers to the ETSI NGSI-LD documentation

- 5.6.5 - Delete Entity Attribute
- 6.7.3.2 - DELETE /ngsi-ld/v1/entities/{entityId}/attrs/{attrId}

8 Forwarding of Creation/Modification of Entities/Attributes

The *Forwarding Concept* for NGSI-LD has still to be specified. We have started to look at this a little (we == the ETSI CIM group defining the NGSI-LD API), but we still have a long way to go before having anything decided and documented.

As Orion-LD builds on Orion and reuses (with modification for expansion and compaction of items) whatever Orion has implemented, Orion-LD actually supports forwarding already, but this will have to change to the NGSI-LD way, once that way is defined.

Forwarding is a mechanism to include context providers as "part of the broker". An example always makes it easier to explain/understand:

1. Context Provider CP-1 informs the broker that it knows about the entity XXX, with attributes A1, A2, ... This is done by sending a Registration request to the broker. The registration request contains information about the entities and attributes, and, importantly, the IP, port and URL-PATH to use to contact the context provider.
2. A query for entity E13, attributes A1-A12 enters the broker. The broker looks at its current registrations, and sees that the registration for CP-1 matches what the query is asking for, so:
3. The broker forwards the initial request to CP-1 and waits for CP-1 to answer.
4. Once the answer from CP-1 has arrived, the broker merges the response from CP-1 with what the broker found in its local database and:
5. The broker responds to the initial request with the merged results from the broker itself and the response from CP-1

That was for retrieval of entities. Updating of entities works pretty much the same way.

Now, for forwarding NGSi-LD requests in Orion-LD. please see the proper documentation of Orion and imagine the attribute names and entity types to be expanded/compacted according to the context. The context used for forwarding must of course be the context specified in the registration. This hasn't even been tested, but still, it should "work a little" ...

9 System Attributes

Each entity and attribute has two timestamps, namely "createdAt", and "modifiedAt". The broker makes sure these two "builtin special attributes" are created and updated accordingly during the lifetime of the entity/attribute. The values *can not* be altered from outside the broker, i.e., there is no service that let's a user set these timestamps to aleatory values. But of course, when modifying an entity, its "modifiedAt" is updated, same with attributes.

A user can ask the broker to include these special attributes in queries though, by specifying the value "sysAttrs" to the URI parameter "options":

```
GET /ngsi-ld/v1/entities?options=sysAttrs&type=T
```

10 Pagination

To avoid returning thousands of items (entities, subscriptions, registrations), Orion-LD establishes a maximum number of items to return. If there are more items to be returned, then the client will have to query again, and again until the client has retrieved all of the items. This concept is called *Pagination* and it's a mechanism for Orion-LD to protect itself against flooding.

The services that use pagination are:

- GET /ngsi-ld/v1/entities
- GET /ngsi-ld/v1/subscriptions
- GET /ngsi-ld/v1/csourceRegistrations
- GET /ngsi-ld/v1/csourceSubscriptions # Not implemented in Alpha Release 1
- GET /ngsi-ld/v1/temporal/entities # Not implemented in Alpha Release 1

The number of items and the index of the first item for pagination are defined by two URI parameters:

- limit=X # X is the number of items
- offset=Y # Y is the offset of the first item

The default values for these two are:

- limit: 20
- offset: 0

Orion-LD implements a maximum limit of 1000. If a request tries to set `limit` above 1000, an error is returned. In Alpha Release 1, Orion-LD reuses the pagination of Orion. Please see the documentation of pagination of *Orion* for more info.

11 Querying for Entities

11.1 GET /ngsi-ld/v1/entities

The `GET /ngsi-ld/v1/entities` service returns an array of entities matching the characteristics specified as URI parameters. Pagination is used to limit the number of entities returned. Also, one of the following URI parameters *must* be present, to narrow down the number of matching entities:

- type
- attrs
- q

This is what the ETSI NGSi-LD specification version 1.2.1 says, at least. Hopefully, this restriction will be removed in version 1.3.1.

11.1.1 Request Payload Data

There is no payload data for this service.

11.1.2 Request URI Parameters

- id - list of entity ids to be retrieved (comma-separated list of URIs)
- type - list of entity types to be retrieved (comma-separated list of types)
- idPattern - regular expression to be matched by entity ids
- attrs - list of attributes (only those attributes are returned and only entities with any of the attrs match)
- q - query string - see separate chapter on 'q'
- csf - not implemented in Alpha 1
- georel - geo relationship (near, within, etc)
- geometry - geometry (point, circle, polygon, ...)
- coordinates - coordinates array, serialized as a string
- geoproperty - not implemented in Alpha 1 - only "location" can be used as geo attribute
- limit - maximum number of entities to be returned
- offset - the index of the first entity

For all geofencing URI params (last four), please refer to the separate chapter on Gellocation

11.1.3 Response HTTP Status Code

- 200 OK
- 400 Bad Request

11.1.4 Response HTTP Headers

- Link - to echo back to the context (if Accept: application/json)

11.1.5 Response Payload Data

The response payload data is a JSON array of the matching entities:

```
[
  { Entity 1 },
  { Entity 2 },
  ...
  { Entity N }
]
```

11.1.6 Pointers to the ETSI NGSI-LD documentation

- 5.2.4 - The NGSI-LD Entity Data Type
- 5.2.5 - The NGSI-LD Property Data Type
- 5.2.6 - The NGSI-LD Relationship Data Type
- 5.2.7 - The NGSI-LD GeoProperty Data Type
- 5.7.2 - Query Entities
- 6.6.4.3.2 - GET /ngsi-ld/v1/entities

12 Retrieval of a Specific Entity

To retrieve a specific entity, one needs to know its *Entity ID*.

12.1 GET /ngsi-ld/v1/entities/{entityId}

The `GET /ngsi-ld/v1/entities/{entityId}` service returns the entity with ID `entityId`, possibly filtered by the URI parameter *attrs*, to only return a specific set of attributes.

12.1.1 Request Payload Data

There is no payload data for this service.

12.1.2 Request URI Parameters

- attrs - list of attributes (only those attributes are returned)

12.1.3 Response HTTP Status Code

- 200 OK
- 400 Bad Request - if the entity ID of the URL PATH is not a valid URI
- 404 Not Found - if the entity ID of the URL PATH does not specify an existing entity

12.1.4 Response HTTP Headers

- Link - to echo back to the context (if Accept: application/json)

12.1.5 Response Payload Data

The response payload data is a JSON object describing the entity in question:

```
{
  <Entity>
}
```

12.1.6 Pointers to the ETSI NGSI-LD documentation

- 5.2.4 - The NGSI-LD Entity Data Type
- 5.2.5 - The NGSI-LD Property Data Type
- 5.2.6 - The NGSI-LD Relationship Data Type
- 5.2.7 - The NGSI-LD GeoProperty Data Type
- 5.7.1 - Retrieve Entity
- 6.5.3.1 - GET /ngsi-ld/v1/entities/{entityId}

13 Forwarding of Query Requests

NGSI-LD forwarding is still to be specified. Please see chapter *Forwarding of Creation/Modification of Entities/Attributes* for more information.

14 Subscriptions

Subscriptions are used to obtain notifications whenever entities/attributes are updated/created, instead of using polling. It's much more efficient to subscribe than to continuously query and investigate the response. Like interrupts vs polling. Subscriptions are like interrupts. Whenever some criteria (defined by the subscription) is fulfilled, a notification is launched (much like an interrupt).

14.1 Creation of Subscriptions

14.1.1 POST /ngsi-ld/v1/subscriptions

Subscriptions are created using the service `POST /ngsi-ld/v1/subscriptions`.

14.1.1.1 Request Payload Data

A subscription with ALL the field should look like this:

```
{
    "id":  "URI",    # if not given,   it will be assigned during subscription process and returned to client
  "type": "Subscription",
  "name": "Name of the subscription",
  "description": "Description of the subscription",
  "entities": [
      {
```

```
      "id": "entity id",    # Optional, takes precedence over idPattern
      "idPattern": "REGEX",  # Optional
      "type": "entity type"  # Mandatory
      }
      {
      "id": "entity id",    # Optional, takes precedence over idPattern
      "idPattern": "REGEX",  # Optional
      "type": "entity type"  # Mandatory
      },
      ...
  ],
  "watchedAttributes": [ "attr1", "attr2", ..., "attrN" ],
  "timeInterval": Number,  # Not Implemented in Alpha 1
  "q": "Query Filter",
  "geoQ": {},  # Not Implemented in Alpha 1
  "csf": "",   # Not Implemented in Alpha 1
  "isActive": true/false,
  "notification": {
      "attributes": [ "attr1", "attr2", ..., "attrN" ],
      "format": "keyValues" / "normalized",
      "endpoint": {
      "uri": "URI which conveys the endpoint which will receive the
notification",
      "accept": "application/json" / "application/ld+json"
      },
      "status": "ok" / "failed"
  },
  "expires": "ISO 8601 String",
   "throttling": Number, # Minimal period of time in seconds which
shall elapse between two consecutive notifications
  "temporalQ": # Not Implemented in Alpha 1,
   "status": "active"/"paused"/"expired"   # Read-only - not to be
given at creation time
}
```

Lots of things to say about this structure. All is already said in the ETSI specification. No need to repeat it here. Please refer to page 46, Table 5.2.12-1 of the ETSI spec version 1.2.1.

The only mandatory fields are: "type" and "notification". Also, either "entities" or "watchedAttributes" *must* be present (both of them at the same time is OK too).

NOTE that the `q` query filter for subscriptions uses the Orion NGSIv2 `q` for subscriptions in Alpha Release 1. The new NGSI-LD `q` Query Filter has been implemented, but is in use only for Queries of Entities in Alpha Release 1.

14.1.1.1.1 MQTT NOTIFICATIONS

The notifications of a subscription can be sent either as REST requests or as MQTT publish messages. In the case of MQTT, the *endpoint::uri* would look as follows:

```
"endpoint": "mqtt://<server-ip>:<port>/<topic>"
```

The normal way for MQTT notifications would be to start an MQTT broker beside the NGSI-LD Broker and make the notifications go to the MQTT broker, while the client subscribes to the *topic* in the MQTT broker apart from creating the subscription in the NGSI-LD broker.

14.1.1.2 Request URI Parameters

This service has no URI Parameters.

14.1.1.3 Response HTTP Status Code

- 201 Created
- 400 Bad Request - in case the request or its content is incorrect
- 409 Already Exists - if the provided subscription id is the id of an already existing subscription

14.1.1.4 Response HTTP Headers

No HTTP headers relevant to NGSI-LD are present in the response.

14.1.1.5 Response Payload Data

If all is OK, a `201 Created` is returned and no payload data is present in the response. On error, the typical `ProblemDetails` structure is returned.

14.1.1.6 Pointers to the ETSI NGSI-LD documentation

- 5.2.8 - EntityInfo
- 5.2.12 - Subscription
- 5.2.14 - NotificationParams
- 5.8.1 - Create Subscription
- 6.10.3.1 - POST /ngsi-ld/v1/subscriptions

14.2 Modification of a Subscription

14.2.1 PATCH /ngsi-ld/v1/subscriptions/{subscriptionId}

Not Implemented in Alpha Release 1

14.3 Querying for Subscriptions

14.3.1 GET /ngsi-ld/v1/subscriptions

The service `GET /ngsi-ld/v1/subscriptions` lets a user query subscriptions. The response is an array of subscriptions that match the query criteria.

14.3.1.1 Request Payload Data

There is no payload data for this service.

14.3.1.2 Request URI Parameters

- limit - maximum number of subscriptions to be returned
- offset - the index of the first subscription

14.3.1.3 Response HTTP Status Code

- 200 OK
- 400 Bad Request - only way to get here is by setting the *limit* too high

14.3.1.4 Response HTTP Headers

- Link

14.3.1.5 Response Payload Data

In case of "200 OK", the response payload data is an array of subscriptions:

```
[
    { Subscription 1 },
    { Subscription 2 },
    ...
    { Subscription N }
]
```

In case of "400 Bad Request", the typical *ProblemDetails* structure is returned.

14.3.1.6 Pointers to the ETSI NGSI-LD documentation

- 5.2.8 - EntityInfo
- 5.2.12 - Subscription
- 5.2.14 - NotificationParams
- 5.8.4 - Query Subscriptions
- 6.10.3.2 - GET /ngsi-ld/v1/subscriptions

14.4 Retrieval of a Specific Subscription

14.4.1 GET /ngsi-ld/v1/subscriptions/{subscriptionId}

This service returns the full information of the subscription whose *Subscription ID* is exactly the same as the last item of the URL PATH.

14.4.1.1 Request Payload Data

There is no payload data for this service.

14.4.1.2 Request URI Parameters

There are no URI Parameters for this service.

14.4.1.3 Response HTTP Status Code

- 200 OK
- 400 Bad Request - the *subscriptionId* in the URL PATH is not a valid URL
- 404 Not Found - the subscription specified in the URL PATH does not exist

14.4.1.4 Response HTTP Headers

- Link

14.4.1.5 Response Payload Data

In case of "200 OK", the response payload data is the entire subscription. E.g.:

```
{
  "id": "http://a.b.c/subs/sub01",
  "type": "Subscription",
  "name": "Test subscription 01",
  "description": "Description of Test subscription 01",
  "entities": [
      {
      "type": "T1"
      },
      {
      "id": "http://a.b.c/E02",
      "type": "T2"
      },
      {
      "idPattern": ".*E03.*",
      "type": "T3"
      }
  ],
  "watchedAttributes": [
      "P2"
  ],
  "q": "P2>10",
  "geoQ": {
      "geometry": "circle",
      "coordinates": "1,2",
      "georel": "near"
  },
  "isActive": false,
```

```
  "notification": {
     "attributes": [
     "P1",
     "P2",
     "A3"
     ],
     "format": "keyValues",
     "endpoint": {
      "uri": "http://valid.url/url",
     "accept": "application/ld+json"
     }
  },
  "expires": "2028-12-31T10:00:00Z",
  "throttling": 5,
  "status": "paused"
}
```

On error, the typical *ProblemDetails* is returned.

14.4.1.6 Pointers to the ETSI NGSI-LD documentation

- 5.2.8 - EntityInfo
- 5.2.12 - Subscription
- 5.2.14 - NotificationParams
- 5.8.3 - Retrieve Subscription
- 6.11.3.1 - GET /ngsi-ld/v1/subscriptions/{subscriptionId}

14.5 Deletion of a Specific Subscription

14.5.1 DELETE /ngsi-ld/v1/subscriptions/{subscriptionId}

To delete a specific subscription, use the service `DELETE /ngsi-ld/v1/subscriptions/{subscriptionId}`.

14.5.1.1 Request Payload Data

There is no payload data for this service.

14.5.1.2 Request URI Parameters

There are no URI Parameters for this service.

14.5.1.3 Response HTTP Status Code

- 204 No Content
- 400 Bad Request - the subscription ID of the URI PATH is not a valid ID (not a URI)

- 404 Not Found - the subscription ID of the URI PATH is not found among the subscriptions.

14.5.1.4 Response HTTP Headers

No HTTP headers relevant to NGSI-LD are present in the response.

14.5.1.5 Response Payload Data

No payload data if all OK. If not OK, *ProblemDetails*.

14.5.1.6 Pointers to the ETSI NGSI-LD documentation

- 5.8.5 - Delete Subscription
- 6.11.3.3 - DELETE /ngsi-ld/v1/subscriptions/{subscriptionId}

15 Notifications

When an update/creation of an entity/attribute gets a hit in the list of subscriptions, a *notification* request is sent to the endpoint that is stated as the receptor of notifications for that subscription. There can of course be more than *one* hit is the list of subscriptions and thus, more than one notification may be sent as a result of the entity modification.

15.1 Notification Context

The context of the notification is the context that was used when creating the subscription. If the subscription that triggered the notification was created with `endpoint::accept` equal to *application/ld+json*, then the context is sent as part of the payload data. If instead `endpoint::accept` is equal to *application/json*, then the context is sent as a Link HTTP header.

15.2 Notification HTTP Headers

- Link (in case endpoint::accept == *application/json*)

15.3 Notification Payload Data

The payload data of a notification contains information of the triggering subscription and the entities that provoked the notification:

```
{
  "id": "notification identifier",
  "type": "Notification",
  "subscriptionId": "subscription identifier",
  "notifiedAt": "DateTime Timestamp corresponding to the instant when
the notification was generated",
  "data": [
      { Entity 1 },
      { Entity 2 },
```

```
      ...
    { Entity N}
  ]
}
```

Note that the number of attributes of the entities can be limited by specifying a list of "interesting" attributes in the field `notification::attributes` when creating the subscription.

16 Registrations

Context Source Providers can be "mini brokers" that implement only a small part of the NGSI-LD API. Typically a Context Provider isn't contacted directly by clients but instead it registers its entities in a broker and the broker will later contact the Context Provider when necessary. The registration is the way for a Context Provider to inform the broker of what entities it has knowledge.

16.1 Creation of Registrations

16.1.1 POST /ngsi-ld/v1/csourceRegistrations

16.1.1.1 Request Payload Data

The payload data at creating a registration looks like this:

```
{
 "id": "URI", # if not given the system assigns an ID for the
registration
 "type": "ContextSourceRegistration", # MANDATORY
 "name": "Name of the registration",
 "description": "Description of the registration",
 "information": [
     { "entities": [
          {
            "id": "URI", // Optional
            "idPattern": "REGEX" // Optional
            "type": "TYPE" // MANDATORY
          },
          { ... }
       ],
       "properties": [
          "Property 1",
          "Property 2",
          …
          "Property N"
       ]
```

```
      "relationships": [
         "Relationship 1",
         "Relationship 2",
         …
         "Relationship N"
      ]
    },
    { ... }
  ],
"observationInterval":
    {
      "start": "ISO 8601 DateTime",
      "end": "ISO 8601 DateTime"
},
"managementInterval":
    {
      "start": "ISO 8601 DateTime",
      "end": "ISO 8601 DateTime"
},
"location": { GeoLocation },
"observationSpace": { GeoLocation },
"operationSpace": { GeoLocation },
"expires": "ISO 8601 DateTime",
"endpoint": "URI",
"Property 1": ,
"Property 2": ,
 …
"Property N":
}
```

NOTE: In Alpha Release 1, the `information` array must have only one item. This is due to the data model of Orion, which Orion-LD follows.

Only three fields are mandatory:

- type
- information
- endpoint

16.1.1.2 Request URI Parameters

There are no URI Parameters for this service.

16.1.1.3 Response HTTP Status Code

- 201 Created
- 400 Bad Request

- 409 Already Exists
- 422 Unprocessable Entity (Unprocessable Context Source Registration)

16.1.1.4 Response HTTP Headers

- Location (if the creation went well)

16.1.1.5 Response Payload Data

No payload data if all OK. If not OK, *ProblemDetails*.

16.1.1.6 Pointers to the ETSI NGSI-LD documentation

- 4.7 Geospatial Properties
- 5.2.9 CsourceRegistration
- 5.2.10 RegistrationInfo
- 5.2.11 TimeInterval
- 5.9.2 Register Context Source
- 6.8.3.1 POST /ngsi-ld/v1/csourceRegistrations

16.2 Modification of Registrations

16.2.1 PATCH /ngsi-ld/v1/csourceRegistrations/{registrationId}

NOT IMPLEMENTED IN ALPHA RELEASE 1

16.3 Querying for Registrations

16.3.1 GET /ngsi-ld/v1/csourceRegistrations

16.3.1.1 Request Payload Data

There is no payload data for this service.

16.3.1.2 Request URI Parameters

- limit - Maximum number of subscriptions to be returned
- offset - The index of the first subscription
- id - Comma separated list of entity identificators
- type - Comma separated list of entity types
- idPattern - REGEX to match entity id
- attrs - Comma separated list of attribute names
- q - Query Filter (Not Implemented in Alpha Release 1)
- csf - Context Source Filter (Not Implemented in Alpha Release 1)
- georel - Geo relationship (Not Implemented in Alpha Release 1)
- geometry - Geometry (point, circle, polygon, ...) (Not Implemented in Alpha Release 1)

- coordinates - Coordinates array, serialized as a string (Not Implemented in Alpha Release 1)
- geoproperty - Which attribute to use as Geo-Attribute (Not implemented in Alpha Release 1)
- timeproperty - Not implemented in Alpha Release 1
- timerel - Not implemented in Alpha Release 1
- time - Not implemented in Alpha Release 1
- endTime - Not implemented in Alpha Release 1

16.3.1.3 Response HTTP Status Code

- 200 OK
- 400 Bad Request

16.3.1.4 Response HTTP Headers

- Link

16.3.1.5 Response Payload Data

The response payload data is an array of Context Source Registrations:

```
{
  < Registration 1>,
  < Registration 2>,
  ...
  < Registration N>
}
```

16.3.1.6 Pointers to the ETSI NGSI-LD documentation

- 5.2.9 - CsourceRegistration
- 5.2.10 - RegistrationInfo
- 5.2.11 - TimeInterval
- 5.9.2 - Register Context Source
- 6.8.3.2 - GET /ngsi-ld/v1/csourceRegistrations

16.4 Retrieval of a Specific Registration

16.4.1 GET /ngsi-ld/v1/csourceRegistrations/{registrationId}

This service lets a user retrieve a specific context source registration. The Registration Identifier must be known though.

16.4.1.1 Request Payload Data

There is no payload data for this service.

16.4.1.2 Request URI Parameters

There are no URI Parameters for this service.

16.4.1.3 Response HTTP Status Code

- 200 OK
- 400 Bad Request - the registration id in the URL PATH is not a valid URI
- 404 Not Found - there is no registration with an ID as the one specified in the URL PATH

16.4.1.4 Response HTTP Headers

- Link

16.4.1.5 Response Payload Data

The entire registration, as a JSON object, e.g.:

16.4.1.6 Pointers to the ETSI NGSI-LD documentation

- 5.2.9 - CsourceRegistration
- 5.2.10 - RegistrationInfo
- 5.2.11 - TimeInterval
- 5.10.1 - Retrieve Context Source Registration
- 6.9.3.1 - GET /ngsi-ld/v1/csourceRegistrations/{registrationId}

16.5 Deletion of Registrations

16.5.1 DELETE /ngsi-ld/v1/csourceRegistrations/{registrationId}

NOT IMPLEMENTED IN ALPHA RELEASE 1

16.6 Subscription to Registrations

Not Implemented in Alpha Release 1

17 Temporal Representation

Temporal Representation in FIWARE is taken care of by other GEs, such as Cygnus, and will not be implemented in Orion-LD.

18 Security

Apart from forbidden characters, all security concerns are taken care of by other GEs, such as PEP, KeyRock, etc and will not be implemented in Orion-LD.

19 Geolocation

TBD

20 Query Filter

TBD