TEMA

TRUSTED
EXTREMELY PRECISE
MAPPING AND PREDICTION
FOR EMERGENCY
MANAGEMENT

NGSI-LD Context Broker Workshop

Antonio Filograna, Matteo Basile

European Commission

This Project has received funding from the European union's
HE Research and Innovation programme under grant agreement 101093003

Agenda

- JSON-LD (@context)
- NGSI: History and Evolution to NGSI-LD
- NGSI-LD Entity
- NGSI-LD Tenant and Scope
- NGSI-LD Attributes and SubAttributes
- Different Payload Format of a JSON-LD
- The @context attribute
- CRUD Operations on NGSI-LD Context Broker
- Query Langugage on an NGSI-LD Context Broker
- NGSI-LD Subscriptions
- Using NGSI-LD Context Broker – TEMA Case
- NGSI-LD vs NGSIv2
- Q&A Session (10 minutes)

JSON-LD

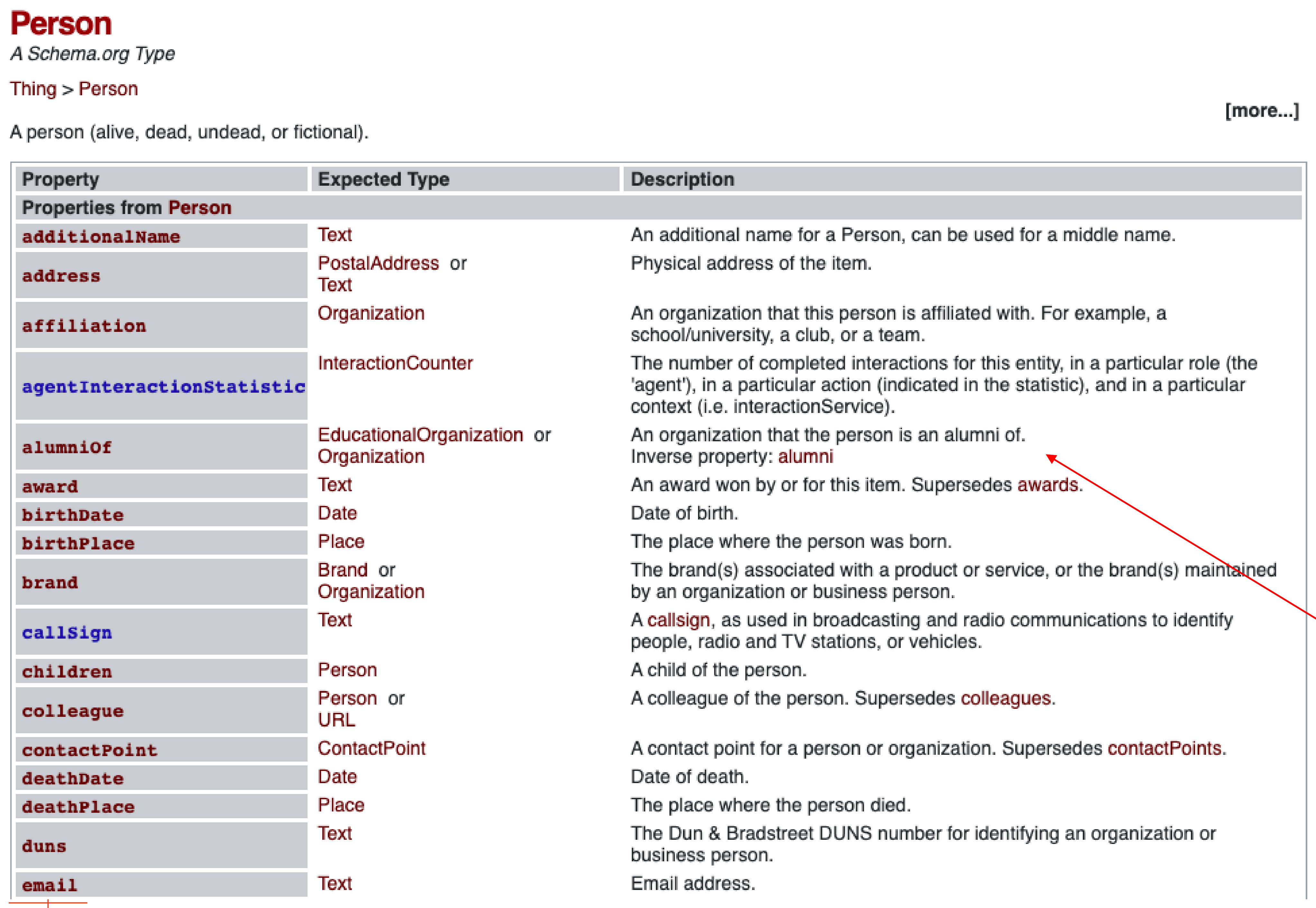
- **JSON** is in data exchange format. However, it is not so easy for machines to read. The attributes of a JSON can have different meanings even among humans themselves.

- For example, consider a JSON that has "***name***" as an attribute. For a person that attribute could mean the person's native name. For another, it could mean the person's stage name. For yet another the username of the person.

- What has been done is to define a **JSON extension** in which I **annotate** a piece of JSON with **additional information**.

- Instead of having the *name* attribute **without context (JSON)** I make that attribute a URI rather than a simple key-value pair. In this way, via the URI, I publicise the fact that that name attribute actually represents the person's real name: **easier for people and machines to understand it**.

JSON-LD Example



Person

A Schema.org Type

Thing > Person

[more...]

A person (alive, dead, undead, or fictional).

Property	Expected Type	Description
Properties from Person		
additionalName	Text	An additional name for a Person, can be used for a middle name.
address	PostalAddress or Text	Physical address of the item.
affiliation	Organization	An organization that this person is affiliated with. For example, a school/university, a club, or a team.
agentInteractionStatistic	InteractionCounter	The number of completed interactions for this entity, in a particular role (the 'agent'), in a particular action (indicated in the statistic), and in a particular context (i.e. interactionService).
alumniOf	EducationalOrganization or Organization	An organization that the person is an alumni of. Inverse property: alumni
award	Text	An award won by or for this item. Supersedes awards.
birthDate	Date	Date of birth.
birthPlace	Place	The place where the person was born.
brand	Brand or Organization	The brand(s) associated with a product or service, or the brand(s) maintained by an organization or business person.
callSign	Text	A callsign, as used in broadcasting and radio communications to identify people, radio and TV stations, or vehicles.
children	Person	A child of the person.
colleague	Person or URL	A colleague of the person. Supersedes colleagues.
contactPoint	ContactPoint	A contact point for a person or organization. Supersedes contactPoints.
deathDate	Date	Date of death.
deathPlace	Place	The place where the person died.
duns	Text	The Dun & Bradstreet DUNS number for identifying an organization or business person.
email	Text	Email address.

```
{
  "@context": {
    "name": "http://schema.org/name",
    "description": "http://schema.org/description",
    "author": "http://schema.org/author",
    "datePublished": {
      "@id": "http://schema.org/datePublished",
      "@type": "http://www.w3.org/2001/XMLSchema#date"
    }
  },
  "@type": "http://schema.org/Book",
  "name": "Example Book",
  "description": "An example book written by John Doe",
  "author": {
    "@type": "http://schema.org/Person",
    "name": "John Doe"
  },
  "datePublished": "2022-01-01"
}
```

NSGI-LD

Next Generation Service Interface-Linked Data (NGSI-LD) is an open **standard** for context information management developed to facilitate the exchange of information between applications in the context of the Internet of Things (IoT).

FIWARE

+ 300 FIWARE Smart Cities over the globe

AWS

AWS for its STF – Smart Territory Framework

IUDX

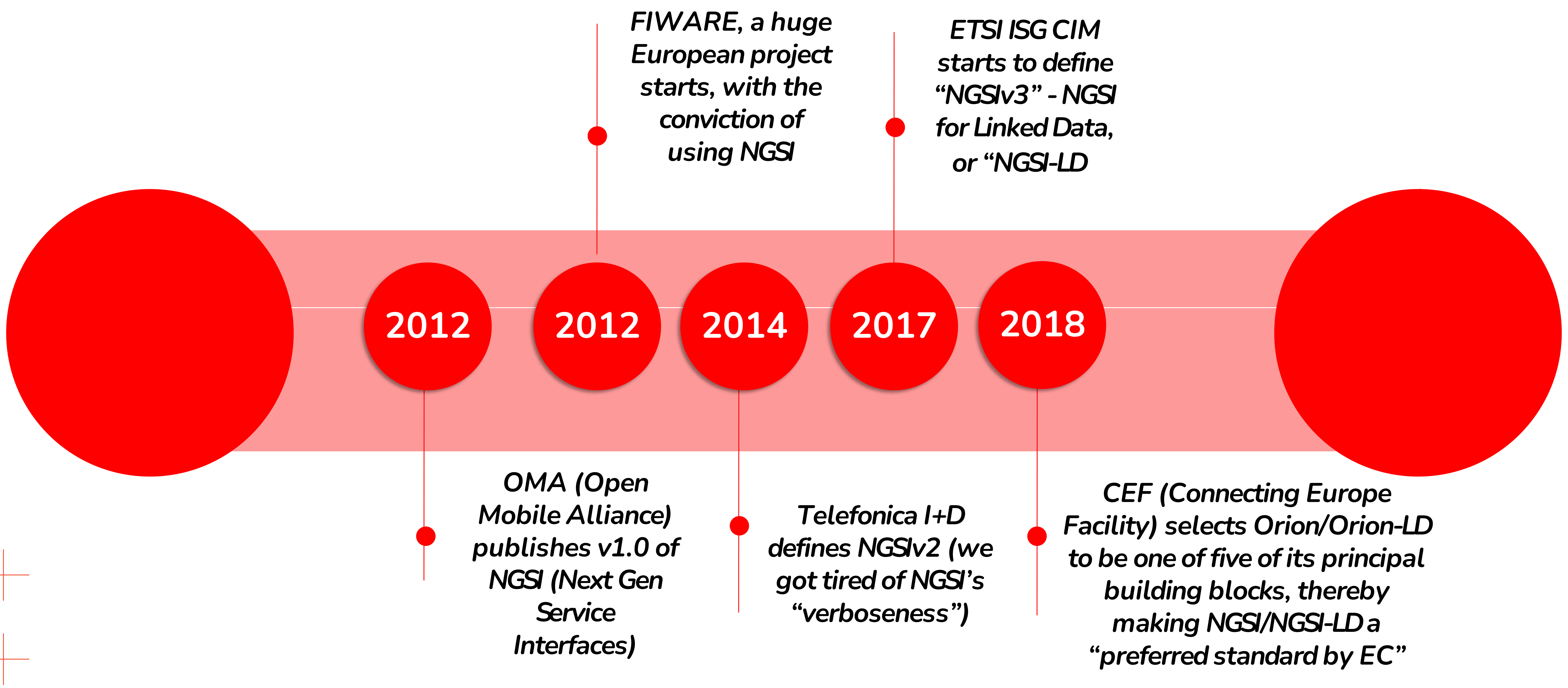
Adopted by IUDX, currently deployed in +35 Smart Cities

Partners

FIWARE's partners supporting NGSI-LD: Red Hat, Atos, Telefonica, NEC

TEMA

NGSI History



2012

OMA (Open Mobile Alliance) publishes v1.0 of NGSI (Next Gen Service Interfaces)

2012

FIWARE, a huge European project starts, with the conviction of using NGSI

2014

Telefonica I+D defines NGSIv2 (we got tired of NGSI's "verboseness")

2017

ETSI ISG CIM starts to define "NGSIv3" - NGSI for Linked Data, or "NGSI-LD

2018

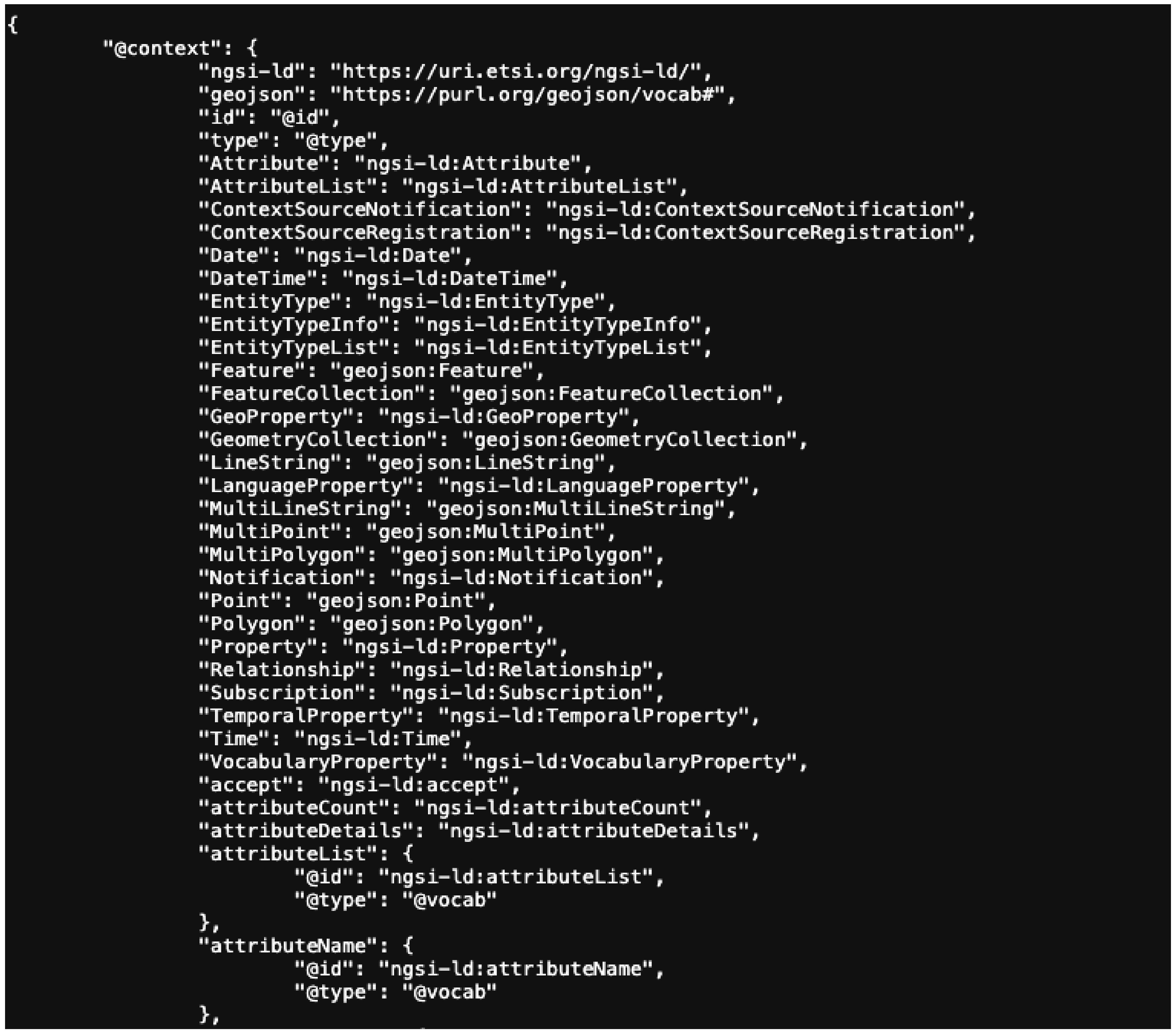
CEF (Connecting Europe Facility) selects Orion/Orion-LD to be one of five of its principal building blocks, thereby making NGSI/NGSI-LD a "preferred standard by EC"

NGSIv2 to NGSI-LD

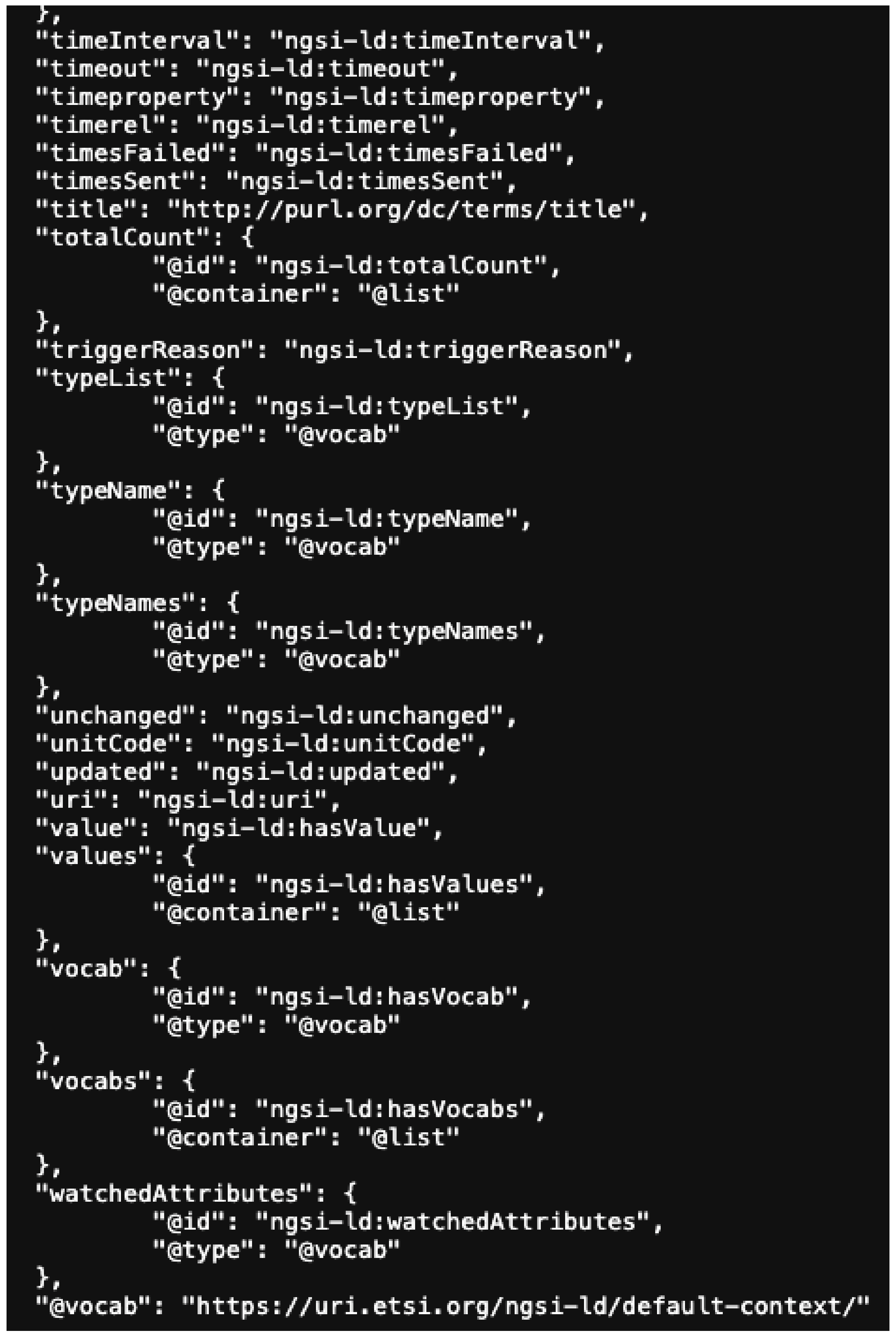
NGSI-LD is the **evolution of the NGSIv2** information model, which has been updated and improved to support entity relationships, property graphs and semantics (JSON-LD):

- The "***id***" now must be an **URN** (or an URI HTTP)
- The entity must have a "***type***" attribute which represent the **class** of the entity
- The class must then be defined in the **@context**
- @context implicitly includes the **core @context** of NGSI-LD: *https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.7.jsonld*

Core @context

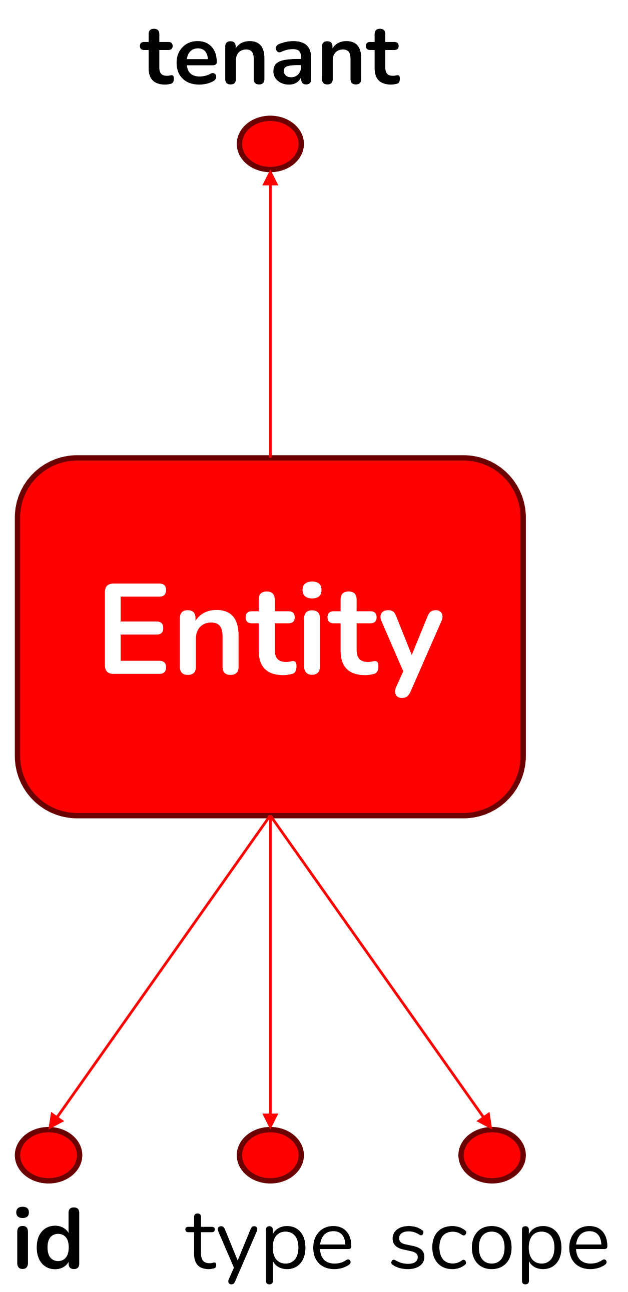


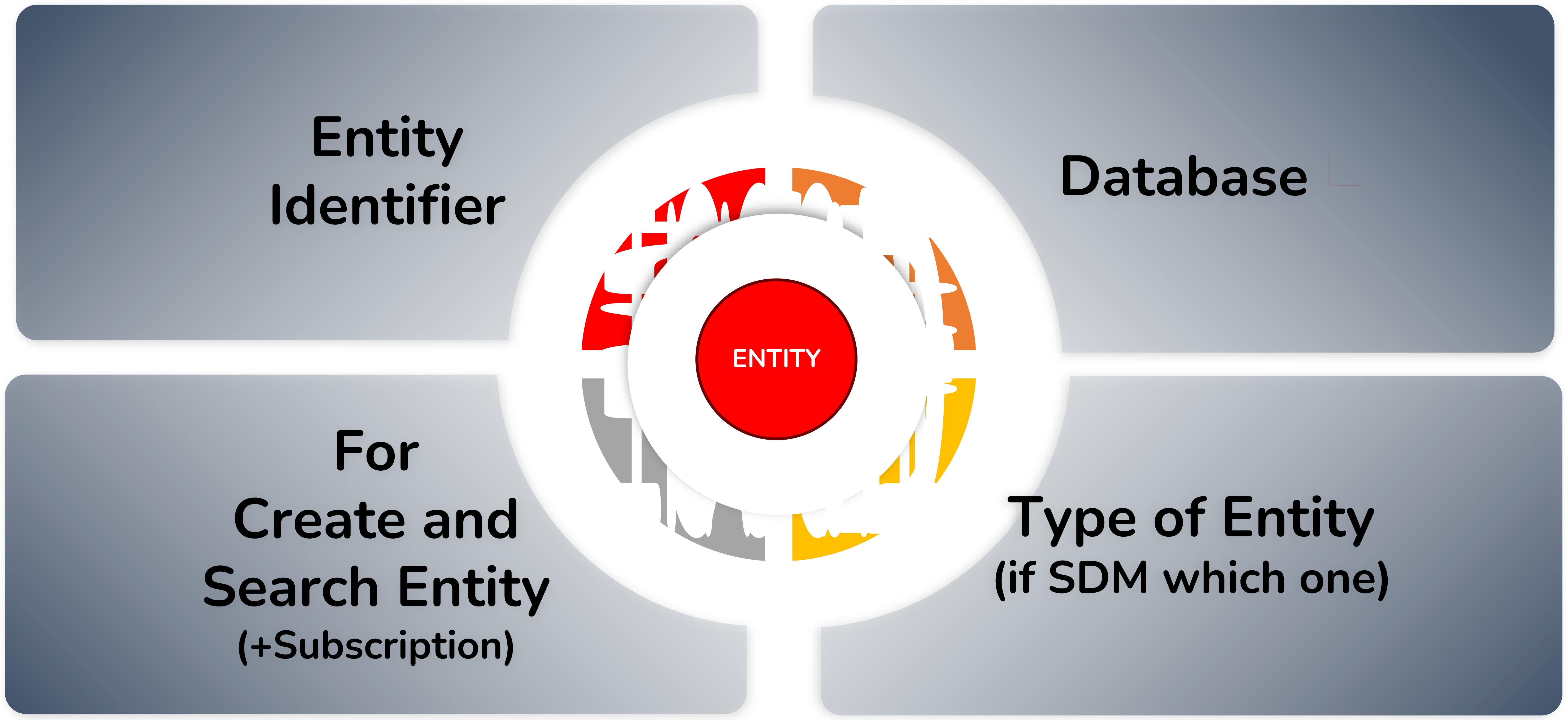
```
{
        "@context": {
                "ngsi-ld": "https://uri.etsi.org/ngsi-ld/",
                "geojson": "https://purl.org/geojson/vocab#",
                "id": "@id",
                "type": "@type",
                "Attribute": "ngsi-ld:Attribute",
                "AttributeList": "ngsi-ld:AttributeList",
                "ContextSourceNotification": "ngsi-ld:ContextSourceNotification",
                "ContextSourceRegistration": "ngsi-ld:ContextSourceRegistration",
                "Date": "ngsi-ld:Date",
                "DateTime": "ngsi-ld:DateTime",
                "EntityType": "ngsi-ld:EntityType",
                "EntityTypeInfo": "ngsi-ld:EntityTypeInfo",
                "EntityTypeList": "ngsi-ld:EntityTypeList",
                "Feature": "geojson:Feature",
                "FeatureCollection": "geojson:FeatureCollection",
                "GeoProperty": "ngsi-ld:GeoProperty",
                "GeometryCollection": "geojson:GeometryCollection",
                "LineString": "geojson:LineString",
                "LanguageProperty": "ngsi-ld:LanguageProperty",
                "MultiLineString": "geojson:MultiLineString",
                "MultiPoint": "geojson:MultiPoint",
                "MultiPolygon": "geojson:MultiPolygon",
                "Notification": "ngsi-ld:Notification",
                "Point": "geojson:Point",
                "Polygon": "geojson:Polygon",
                "Property": "ngsi-ld:Property",
                "Relationship": "ngsi-ld:Relationship",
                "Subscription": "ngsi-ld:Subscription",
                "TemporalProperty": "ngsi-ld:TemporalProperty",
                "Time": "ngsi-ld:Time",
                "VocabularyProperty": "ngsi-ld:VocabularyProperty",
                "accept": "ngsi-ld:accept",
                "attributeCount": "ngsi-ld:attributeCount",
                "attributeDetails": "ngsi-ld:attributeDetails",
                "attributeList": {
                        "@id": "ngsi-ld:attributeList",
                        "@type": "@vocab"
                },
                "attributeName": {
                        "@id": "ngsi-ld:attributeName",
                        "@type": "@vocab"
                },
```



```
},
"timeInterval": "ngsi-ld:timeInterval",
"timeout": "ngsi-ld:timeout",
"timeproperty": "ngsi-ld:timeproperty",
"timerel": "ngsi-ld:timerel",
"timesFailed": "ngsi-ld:timesFailed",
"timesSent": "ngsi-ld:timesSent",
"title": "http://purl.org/dc/terms/title",
"totalCount": {
        "@id": "ngsi-ld:totalCount",
        "@container": "@list"
},
"triggerReason": "ngsi-ld:triggerReason",
"typeList": {
        "@id": "ngsi-ld:typeList",
        "@type": "@vocab"
},
"typeName": {
        "@id": "ngsi-ld:typeName",
        "@type": "@vocab"
},
"typeNames": {
        "@id": "ngsi-ld:typeNames",
        "@type": "@vocab"
},
"unchanged": "ngsi-ld:unchanged",
"unitCode": "ngsi-ld:unitCode",
"updated": "ngsi-ld:updated",
"uri": "ngsi-ld:uri",
"value": "ngsi-ld:hasValue",
"values": {
        "@id": "ngsi-ld:hasValues",
        "@container": "@list"
},
"vocab": {
        "@id": "ngsi-ld:hasVocab",
        "@type": "@vocab"
},
"vocabs": {
        "@id": "ngsi-ld:hasVocabs",
        "@container": "@list"
},
"watchedAttributes": {
        "@id": "ngsi-ld:watchedAttributes",
        "@type": "@vocab"
},
"@vocab": "https://uri.etsi.org/ngsi-ld/default-context/"
```

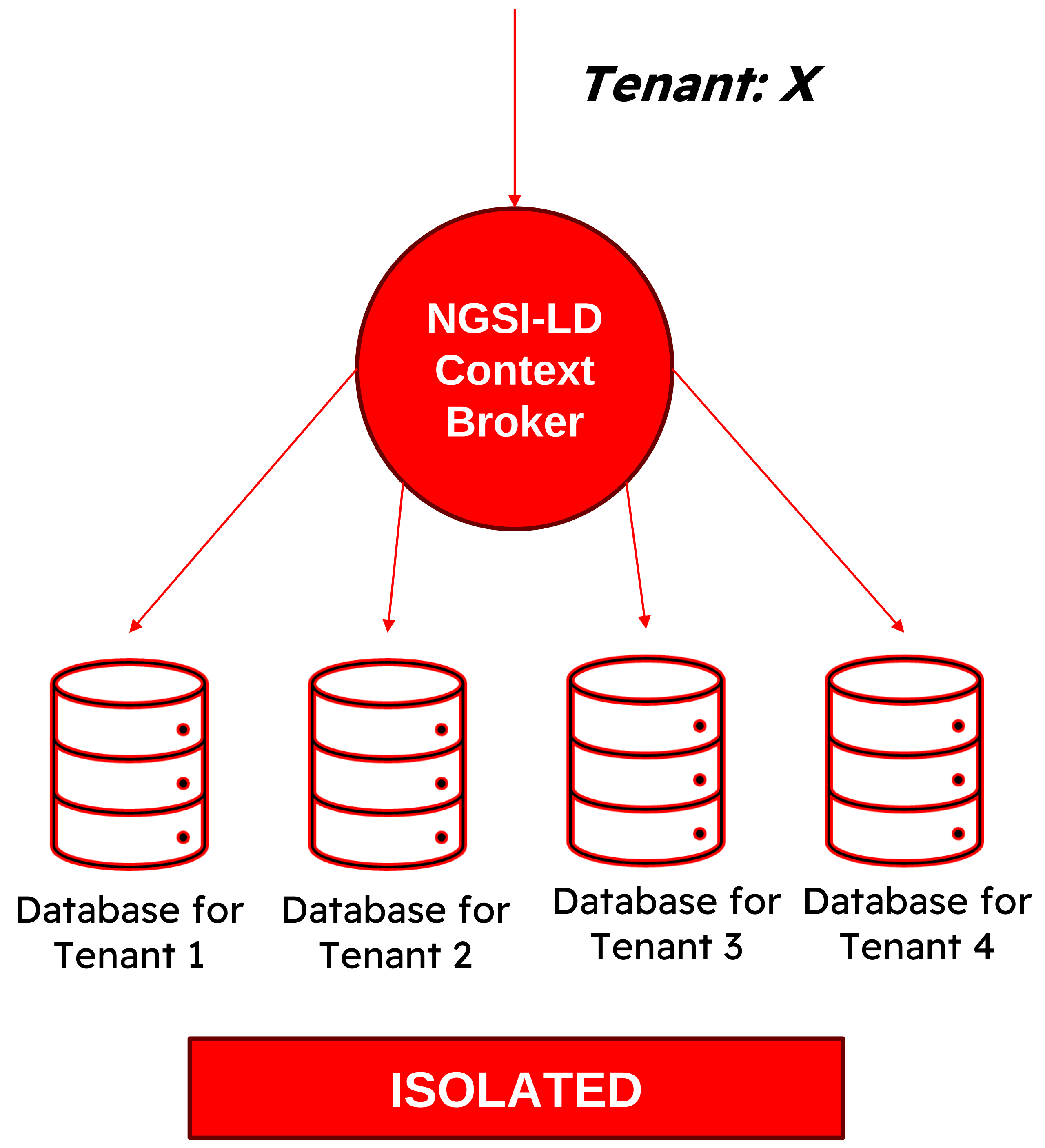
NGSI-LD Entity




Entity Identifier
Database
ENTITY
For Create and Search Entity (+Subscription)
Type of Entity (if SDM which one)

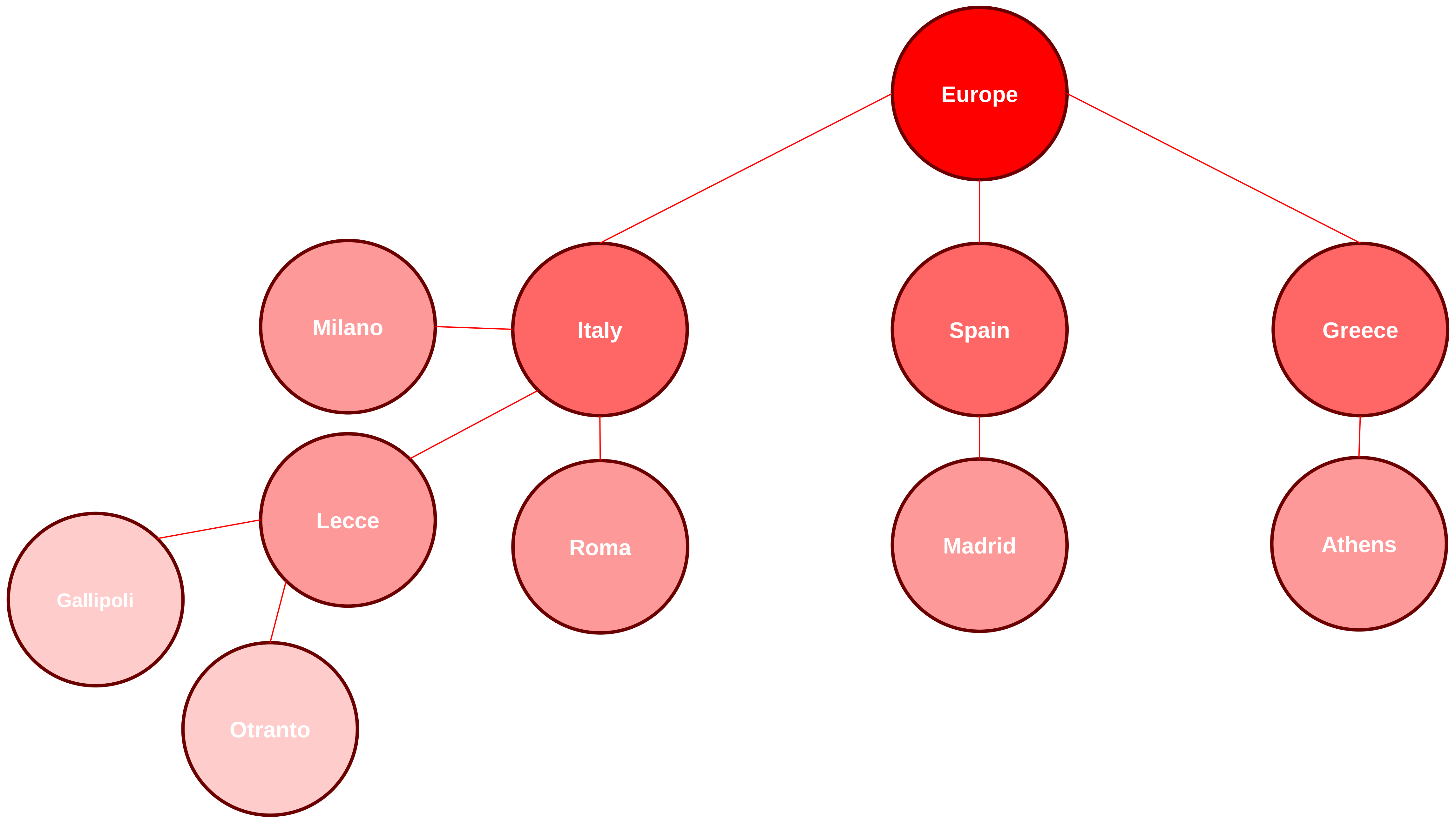
NGSI-LD – Tenant

In NGSI-LD, a 'tenant' refers to a separate instance within a system or platform using this standard. The idea is to **isolate information and resources between different organisations or users within the same system**. Using tenants, **a single broker can serve more than one "system" at a time**, each system with its own database, its own "tenant". The header "***NGSILD-Tenant: <tenant>***" is used to choose the tenant.



NGSI-LD – Scope

In NGSI-LD, the "NGSI-LD-Scope" is used when creating and searching for entities. An entity can have more than one scope, and searches support a list of scopes, as well as wildcards (e.g. /Europe/Italy/Lecce/#).



NGSI-LD – Entity Type

NGSI-LD utilizes the type attribute to specifically identify an entity's type. This serves as a representation of the entity's **class or category**. Each individual entity can possess one or multiple types, which assist in defining its unique **characteristics** and **properties**.

The type attribute plays a crucial role in **data semantics**. It provides valuable insights about an entity's characteristics and behaviour by defining its specific type. For instance, the type "Car" suggests that the entity will possess attributes like "speed", "location” and "fuelLevel".

TEMA

NGSI-LD – Entity Id

The identifier of an entity in NGSI-LD must be a URI (**Uniform Resource Identifier**). URIs are strings that uniquely identify a resource on the Internet. In the context of NGSI-LD, the entity id represents a specific entity in the context of the Internet of Things (IoT). Infact, the entity id must be unique within the context in which it is used. For example, it could have the following structure:

urn:ngsi-ld:<*entity_type*>:<*entity_id*>

(URIs may be represented in different formats, such as HTTP URL, URN, or any other scheme supported by URIs)

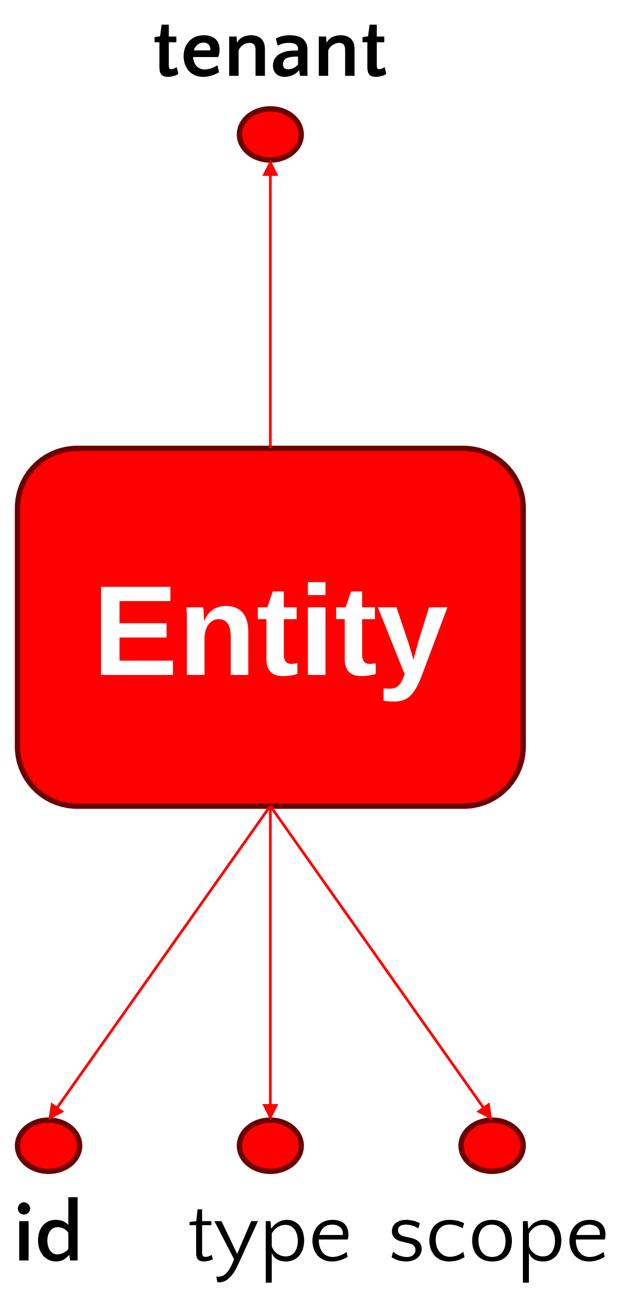
Where:

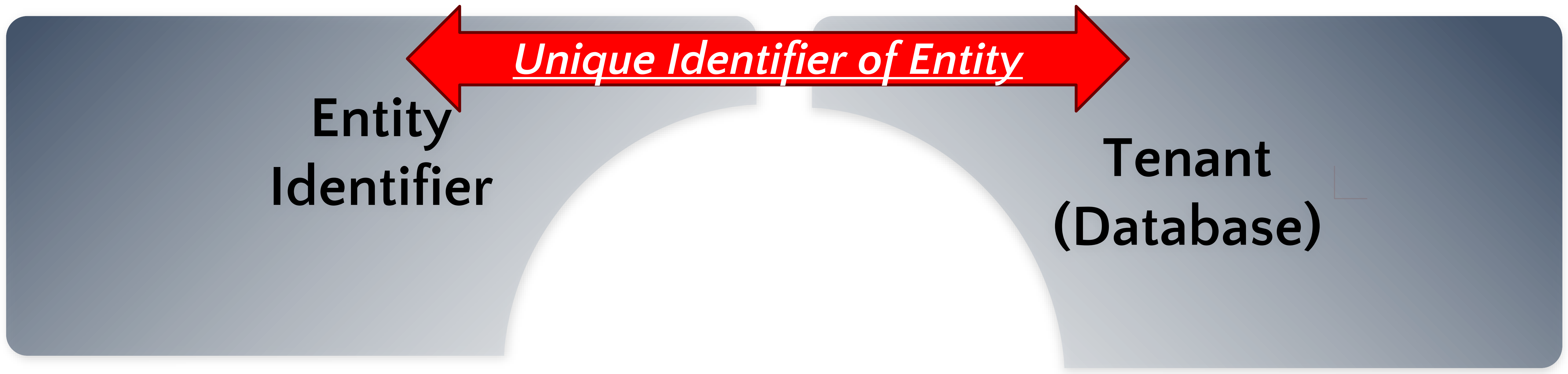
- <*entity_type*> is the type of the entity (e.g. "Car", "WeatherStation")
- <*entity_id*> is a unique identifier for that entity within the specific entity type

Example:

urn:ngsi-ld:car:1234

Entity Unique Identifier



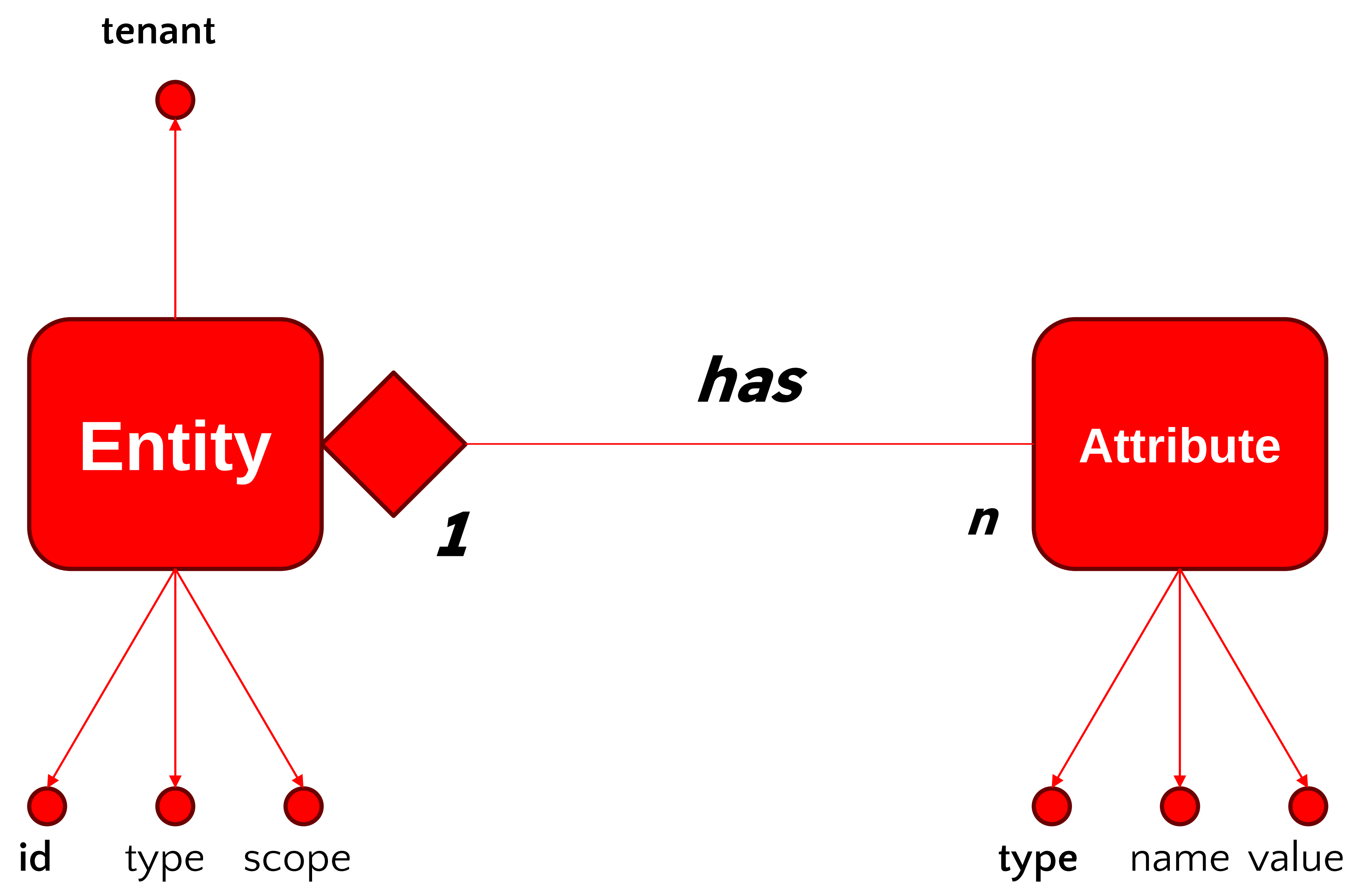


The "total" identifier of an entity is the set of { id, tenant }
In other words, there can be two entities with the same "id" but in different tenants.

NGSI-LD – Entity Attributes Entity-Attribute-Value

In NGSI-LD, the attributes of an entity represent the characteristics or properties of the entity. In adherence to the Entity-Attribute-Value (EAV) model, an entity in NGSI-LD consists of:

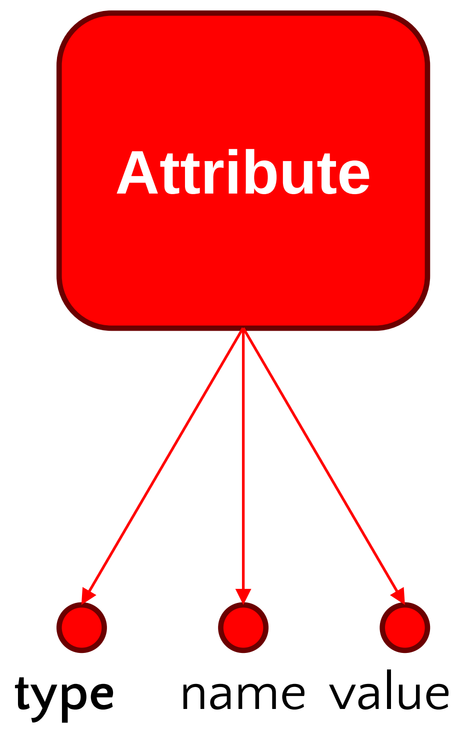
- Entity: A tangible object in the physical world, such as a car or weather station, capable of being defined and tracked.
- **Attribute**: Descriptors that define an entity's specific characteristics.
- **Value**: The current measurement or state of an attribute.



NGSI-LD – Entity Attributes

An attribute is a component of an entity that describes a particular characteristic or property of that entity. Each attribute is composed of:

- **Type**:
  - *Property*: an attribute of an entity that has a single value. For example, the temperature of a sensor may be an attribute of type Property.
  - *GeoProperty*: a geographical attribute of an entity (usually a GeoJSON).
  - *TemporalProperty*: keeps track of the temporal information associated with a given attribute.
  - *LanguageProperty*: linguistic information, e.g. the language of a text associated with an entity.
  - *VocabularyProperty*: uses a specific vocabulary to define its value, contributing to better semantic interoperability.
  - *Relationship*: a relationship between two entities.
  - *ListProperty*: a list of values rather than a single value (list of temperature measurements).
  - *ListRelationship*: a relationship involving more than two entities.
- **Name**: refers to the field that uniquely identifies an attribute within an entity.
- **Value**: the actual information associated with a specific attribute.



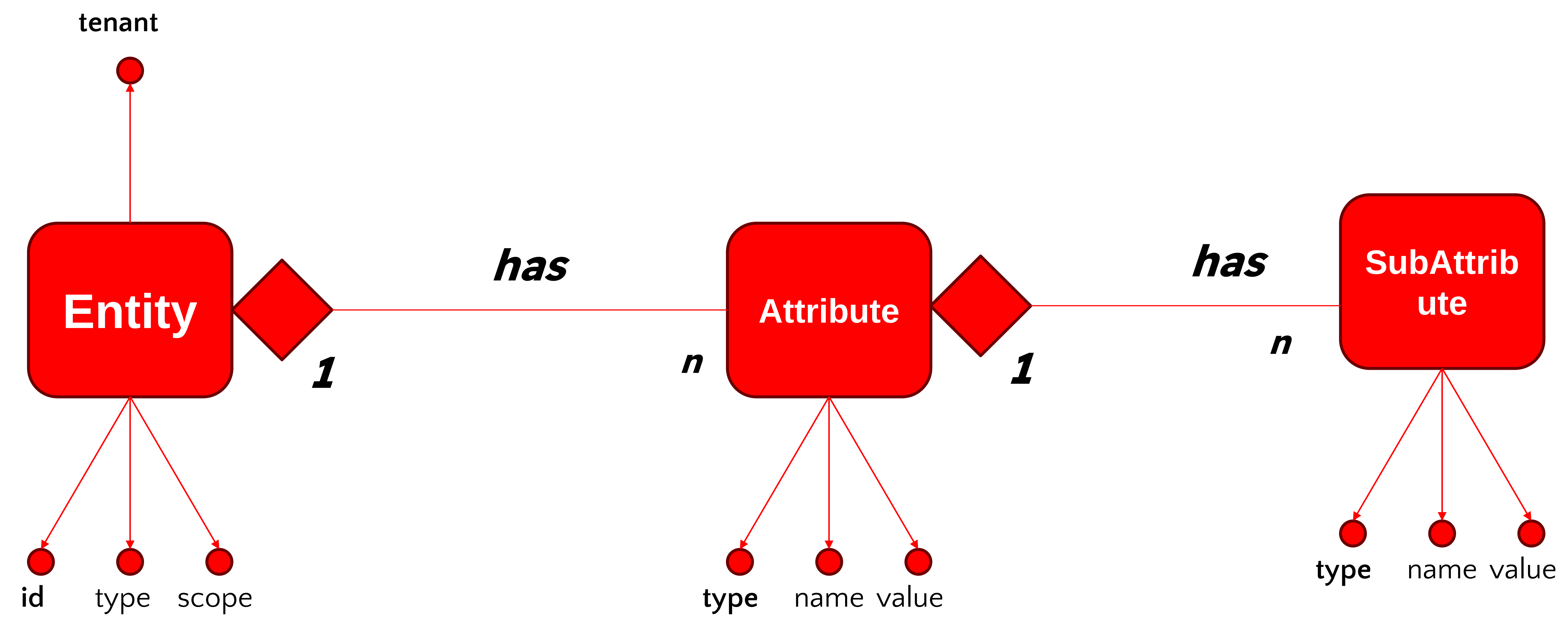
TEMA

NGSI-LD – Sub-Attribute

The concept of “Sub-Attribute” can sometimes be used informally to refer to a hierarchical structure of attributes within a data model.

Name:

- unitCode
- observedAt



NGSI-LD – Different Payload Formats

NGSI-LD supports three different formats for JSON payload (for input as well as output):

- Normalized
- Concise
- Simplified

Normalized

```
{
  "id": "urn:ngsi-ld:Vehicle:0001",
  "type": "Vehicle",
  "speed": {
    "type": "Property",
    "value": 90,
    "unitCode": "km/h"
  },
  "brand": {
    "type": "Property",
    "value": "Tesla"
  }
}
```

Concise

```
{
  "id": "urn:ngsi-ld:Vehicle:0001"
  "type": "Vehicle",
  "speed": {
    "value": 90,
    "unitCode": "km/h"
  },
  "brand": "Tesla"
}
```

Simplified

```
{
  "id": "urn:ngsi-ld:Vehicle:0001",
  "type": "Vehicle",
  "speed": 90,
  "brand": "Tesla"
}
```

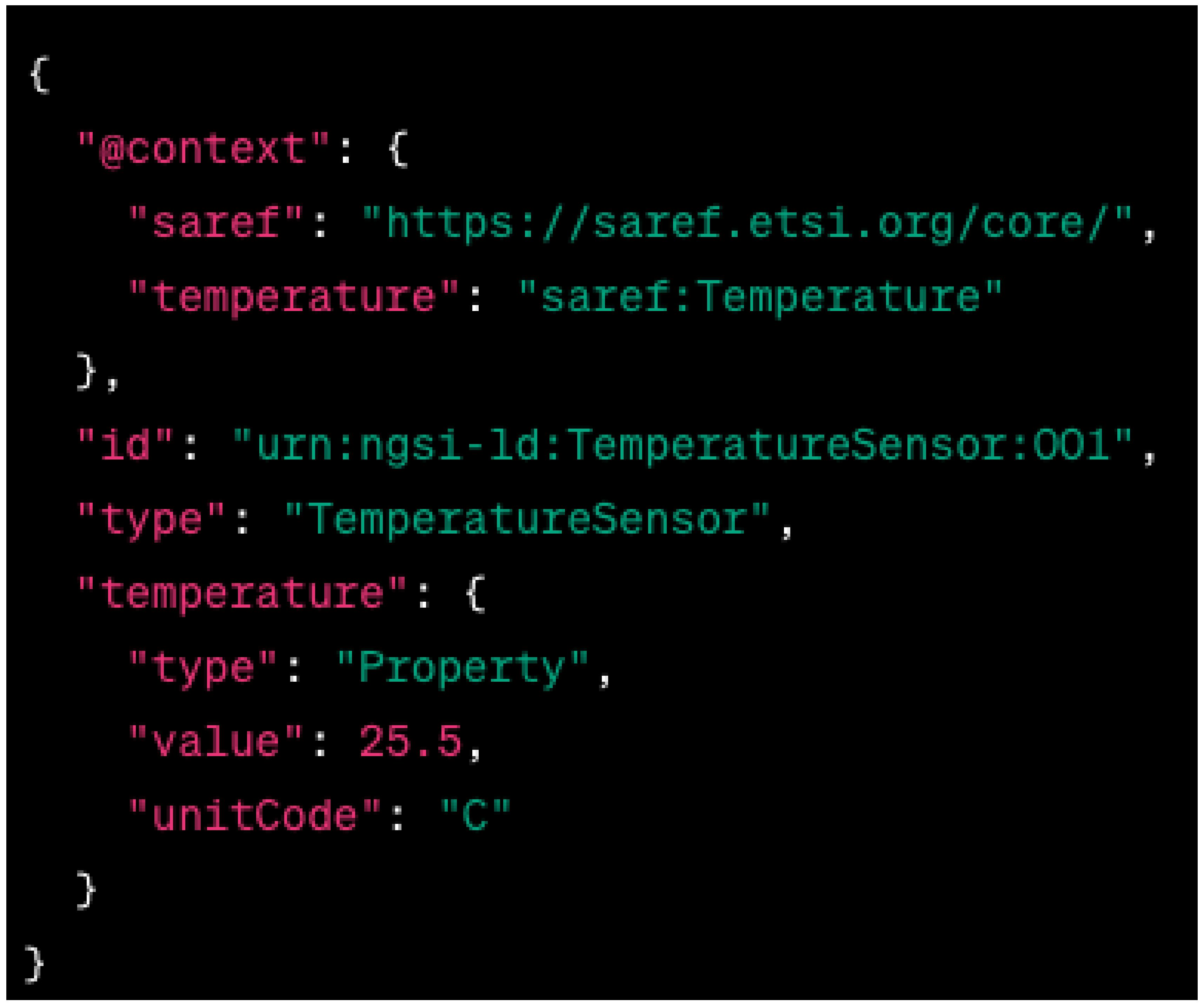
NGSI-LD – @context

The @context field in NGSI-LD is used to define the **semantic context of the data in a resource**. The context helps interpret and understand the semantics of the data, indicating how to interpret data types, attributes and other elements within a JSON-LD resource. This context may be a URL pointing to an external context document or a JSON object embedded in the document itself. Context is crucial to correctly interpret the meaning of the data in the JSON-LD document and to **ensure proper semantic interoperability between different systems**.

Default Context: The Core @context is a built-in in an ETSI NGSI-LD compliant broker and it has precedence over any user supplied @context.

NGSI-LD – @context: Example

The @context field in NGSI-LD is used to define the **semantic context of the data in a resource**. The context helps interpret and understand the semantics of the data, indicating how to interpret data types, attributes and other elements within a JSON-LD resource. This context may be a URL pointing to an external context document or a JSON object embedded in the document itself.



```
{
  "@context": {
    "saref": "https://saref.etsi.org/core/",
    "temperature": "saref:Temperature"
  },
  "id": "urn:ngsi-ld:TemperatureSensor:001",
  "type": "TemperatureSensor",
  "temperature": {
    "type": "Property",
    "value": 25.5,
    "unitCode": "C"
  }
}
```

NGSI-LD – CRUD on Entities/Attributes

The Base Path of an NGSI-LD Context Broekr is:

https://<YOUR_DOMAIN>/ngsi-ld/v1

(The /v2 is for NGSI-v2)

Create

POST	/entities
POST	/entityOperations/create
POST	/entityOperations/upsert

Read

GET	/entities
GET	/entities/{entityId}
POST	/entityOperations/query

Update

PUT	/entities/{entityId}
PATCH	/entities/{entityId}
POST	/entities/{entityId}/attrs
PATCH	/entities/{entityId}/attrs
PATCH	/entities/{entityId}/attrs/{attrName}
POST	/entityOperations/update

Delete

DELETE	/entities/{entityId}
DELETE	/entities/{entityId}/attrs/{attrName}
POST	/entityOperations/delete

NGSI-LD – Options for Querying Entities

Filters are criteria or parameters used to restrict and select specific entities or information during a query of entities. These filters are used to customise queries and obtain only the desired data. Filters are represented as parameters in the URI, except when using a “POST Query”, where they are included in the body of the request (*).

Id
Comma-separated list of Entity iDentifiers

type
Comma-separated list of Entity Types

q
NGSI-LD Query Language, over attributes

geo-query*
Really 4 different URL-params

idPattern
REGEX to match an Entity IDentifier

attrs*
Comma-separated list of attribute names (two functions)

scopeQ
Comma-separated list of scopes

lang*
To select the language to use for LanguageProperties

TEMA

NGSI-LD – Query Language: Operators

The “NGSI-LD Query Language” is a query language used in the context of NGSI-LD to **perform advanced queries on entity attributes**. **Operators** are fundamental elements of this language and are used to **define query conditions**.

q

NGSI-LD Query Language, over attributes

Existence

q=A (no operator, just the name of the attr)

Non-Existence

q=!A

Equality

q=A==7
q=A==7,13,23
q=A==(7,23)

Inequality

q=A!=7
q=A!=7,13,23
q=A!=(7,23)

GT

q=B.observedAt>”2023”

LT

q=C[present]<23.4

GTE

q=D>=16

LTE

q=E<=19

Pattern Match

q=A~=“<REGEX>”

AND

OR

Parenthesis

NGSI-LD – Query Language: Operators

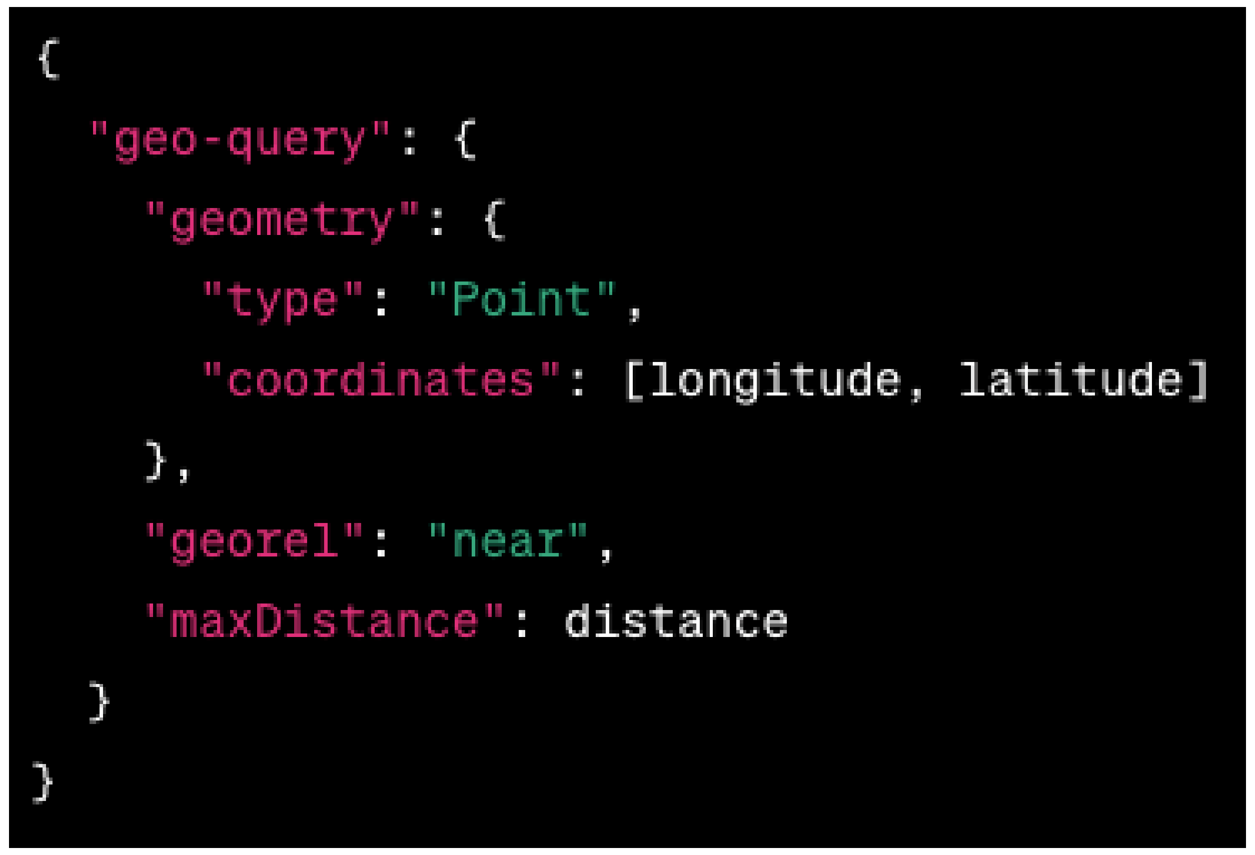
The "geo-query" is a functionality within NGSI-LD that allows **spatial queries to be performed to select entities based on geographical criteria**. This functionality is particularly useful when managing geographical or geospatial location data.z

Point	**geo-query***	MultiLineString
MultiPoint	Really 4 different URL-params	Polygon
LineString		MultiPolygon

Each geometry type has its own "**georels**" that specifies the **geographical relationship** between the entity and the specified point. (e.g., "near;minDistance==X" for "Point"). See: https://www.rfc-editor.org/rfc/rfc7946.

Example (GET) */entities?geoProperty=location&geometry=Point&coordinates=[1,2]&georel=near;maxDistance==100*

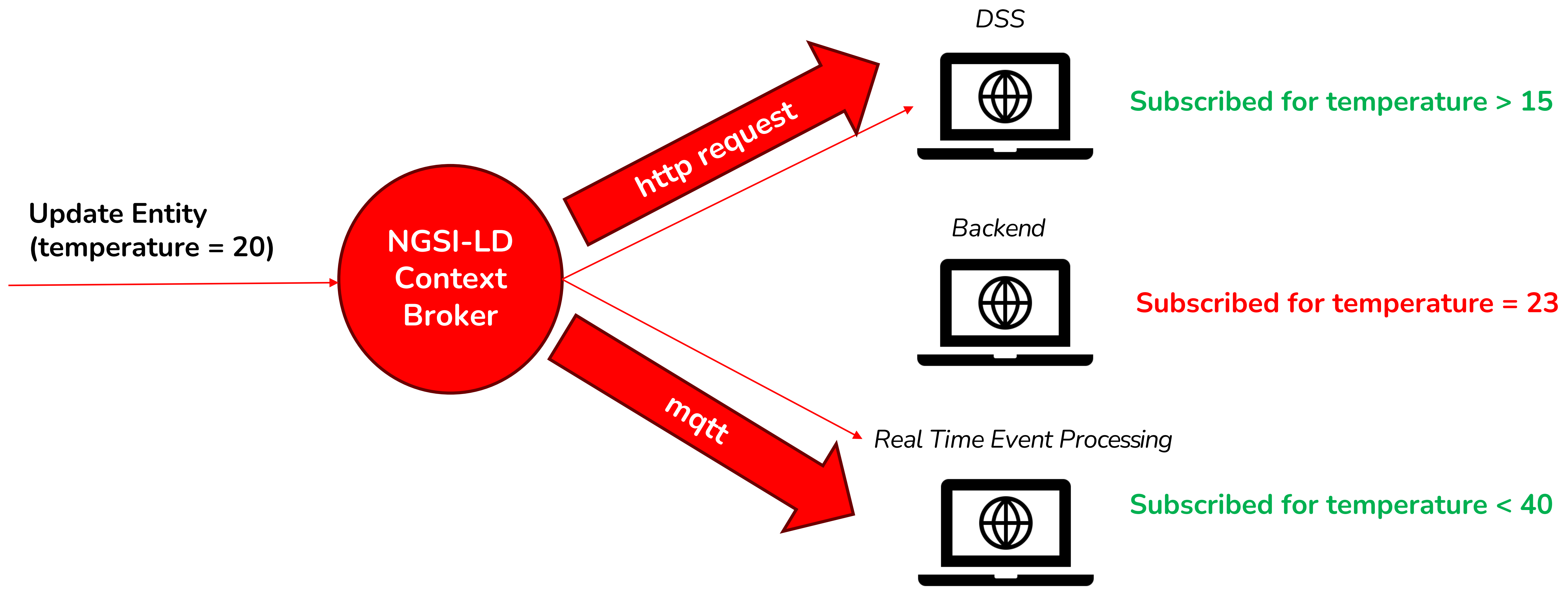
Example (POST)



```
{
  "geo-query": {
    "geometry": {
      "type": "Point",
      "coordinates": [longitude, latitude]
    },
    "georel": "near",
    "maxDistance": distance
  }
}
```

NGSI-LD – Subscriptions

Subscriptions are mechanisms that allow users **to receive real-time notifications when information associated with entities meets certain specified conditions**. A Subscription defines “what to get notified for” and “where and in what format to send the notification”. The notification are sent by CB **via HTTP or MQTT Protocols**.


Update Entity (temperature = 20)
NGSI-LD Context Broker
http request
mqtt
DSS
Subscribed for temperature > 15
Backend
Subscribed for temperature = 23
Real Time Event Processing
Subscribed for temperature < 40

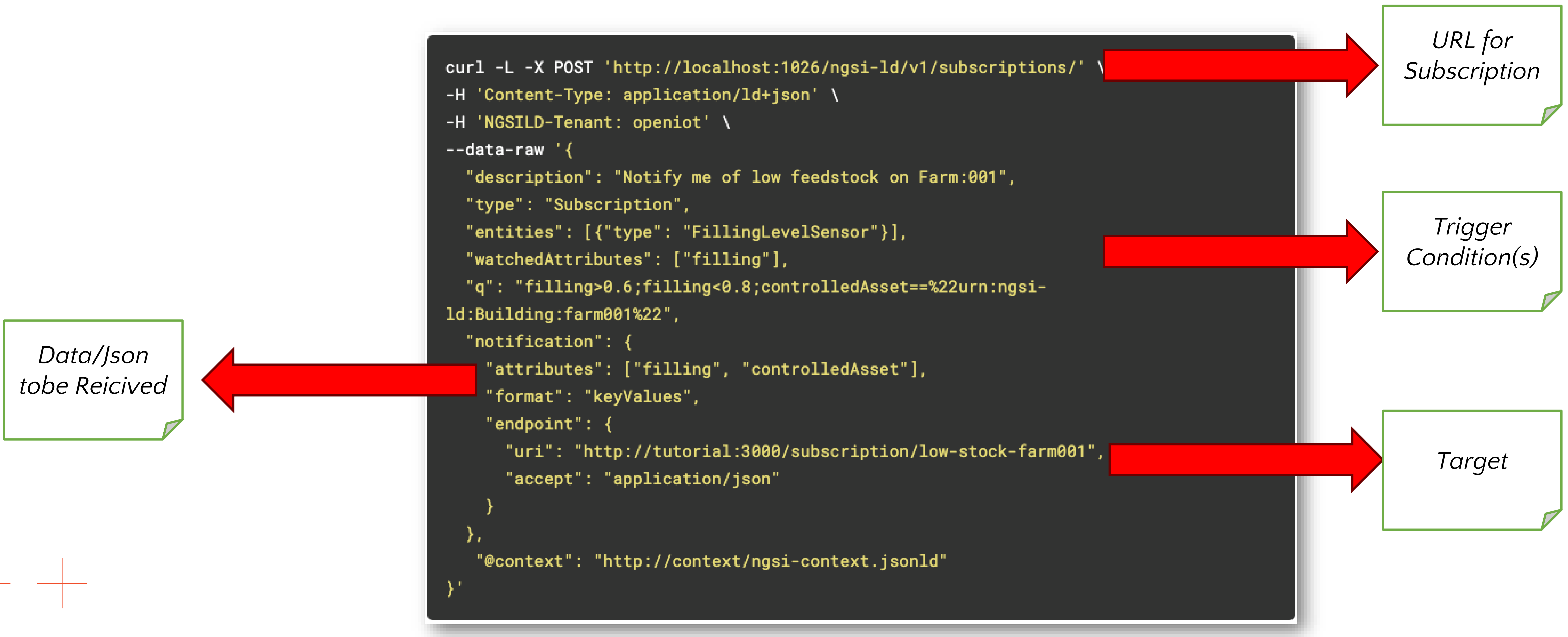
NGSI-LD Subscription Parmeters

A subscription in NGSI-LD is characterized by:

- **Watched Attributes and Conditions**: when you want to receive notification (example temperature greater than 20)
- **Entities**: which entity you want to monitor (obviously in the example before they must have the temperature attribute)
- **Notification**: here you can specify the endpoint (HTTP or MQTT) to which you want to receive the notification and the data format

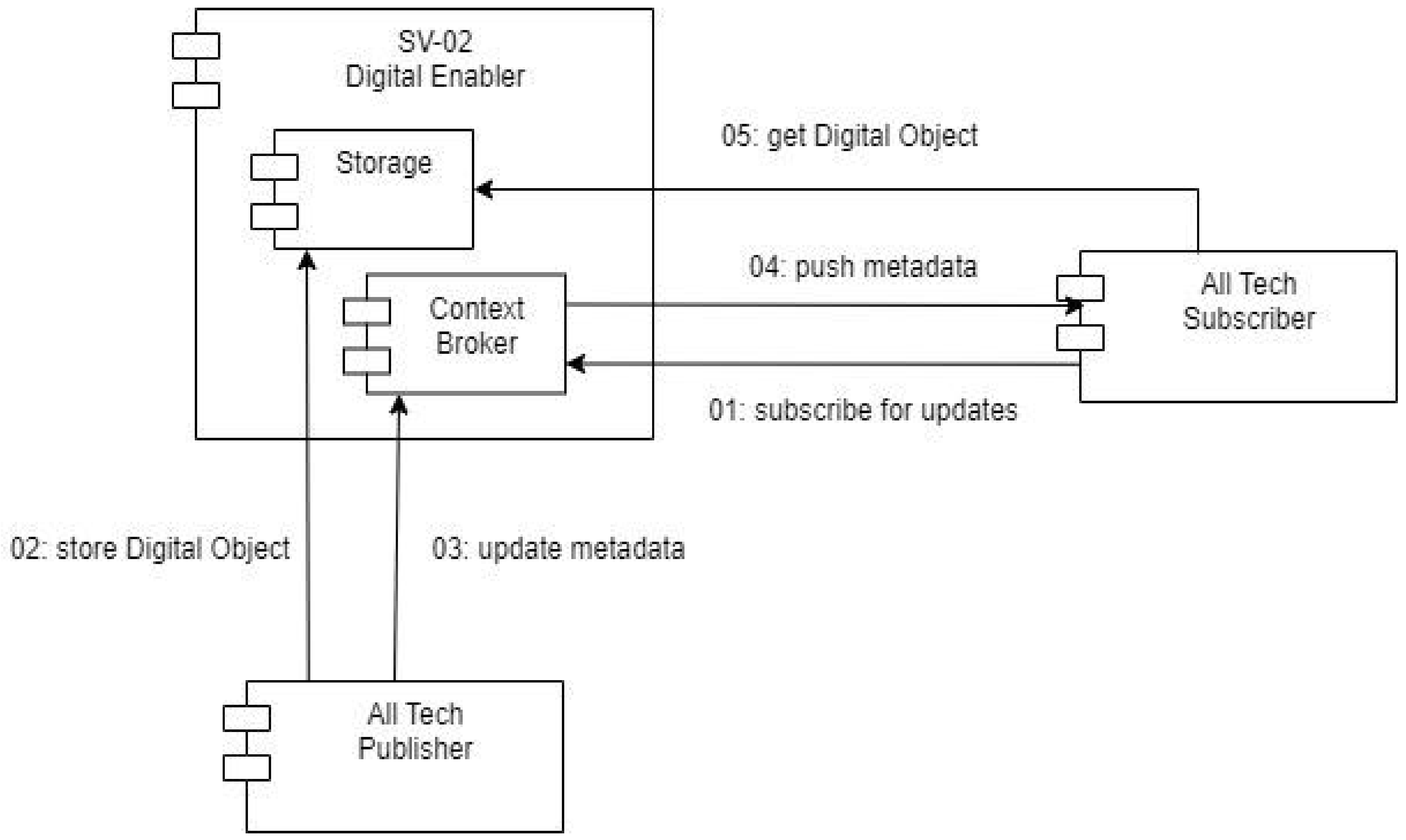
See example in the next slide.

NGSI-LD – Subscriptions (Example)



```
curl -L -X POST 'http://localhost:1026/ngsi-ld/v1/subscriptions/' \
-H 'Content-Type: application/ld+json' \
-H 'NGSILD-Tenant: openiot' \
--data-raw '{
  "description": "Notify me of low feedstock on Farm:001",
  "type": "Subscription",
  "entities": [{"type": "FillingLevelSensor"}],
  "watchedAttributes": ["filling"],
  "q": "filling>0.6;filling<0.8;controlledAsset==%22urn:ngsi-
ld:Building:farm001%22",
  "notification": {
    "attributes": ["filling", "controlledAsset"],
    "format": "keyValues",
    "endpoint": {
      "uri": "http://tutorial:3000/subscription/low-stock-farm001",
      "accept": "application/json"
    }
  },
   "@context": "http://context/ngsi-context.jsonld"
}'
```

How to use Context Broker in TEMA



Storage: MINIO (02)

CREATE MINIO CLIENT

To interface with Minio, you need to create a MINio Client.
The required parameters are:

- url minio
- client_id
- client_secret

Example code:

```
MinioClient minioClient =
    MinioClient.builder()
        .endpoint("https://play.min.io")
        .credentials("Q3AM3UQ867SPQQA43P2F",
"zuf+tfteSlswRu7BJ86wekitnifILbZam1KYY3TG")
        .build();
```

TEMA

Storage: MINIO (02)

UPLOAD OBJECT:
After the creation of Minio Client, it's possible to uploade contents from a file as object in bucket.

Upload a video file.

```
minioClient.uploadObject(
    UploadObjectArgs.builder()
        .bucket("my-bucketname")
        .object("my-objectname")
        .filename("my-video.avi")
        .contentType("video/mp4")
        .build());
```

Upload an JSON file.

```
minioClient.uploadObject(
    UploadObjectArgs.builder()
        .bucket("my-
bucketname").object("my-
objectname").filename("person.j
son").build());
```

Create/Update metadata (03)

POST API:
url: http://temacontextbroker.eu/ngsi-ld/v1/entities/
body:

```
            {
                    “id”:”urn:ngsi-ld:TemaEvent:Sardinia:26bc1c18-6cc3-11ee-b962-0242ac120002”,
                    “type”: “TemaEvent”,
                    “creator”: “Team Sardinia”,
                    “title”: “Drone image”
          “description” : “Fire occurred in Sardinia at 11pm”,
          “img_url”:“http://temastorage.eu/Tema/Sardinia/FireImages/651d7e8370125df5812ebb1d.jpg”

      }
```

TEMA

Context broker push metadata (04)

The payload data of a notification contains information about the subscription that triggered the notification and the entities that provoked it.

```
{
"id": "notification identifier",
"type": "Notification",
"subscriptionId": "urn:ngsi-ld:Subscription:Sardinia:TemaEvent",
"notifiedAt": "DateTime Timestamp corresponding to the instant when
the notification was generated",
"data": [
{ Entity 1 },
{ Entity 2 },
...
{ Entity N} ]
}
```

get Digital Object (05)

GET DIGITAL OBJECT:
After the creation of Minio Client, it’s possible to get and download contents from minio

```
// get object given the bucket and object
name
try (InputStream stream =
minioClient.getObject(
  GetObjectArgs.builder()
  .bucket("my-bucketname")
  .object("my-objectname")
  .build())) {
  // Read data from stream
}
```

```
// Download object given the bucket,
object name and output file name
minioClient.downloadObject(
  DownloadObjectArgs.builder()
  .bucket("my-bucketname")
  .object("my-objectname")
  .filename("my-object-file")
  .build());
```

TEMA

NGSI-LD vs NGSI-v2

NGSI-LD		NGSIv2
It uses the JSON-LD (Linked Data) serialisation format to represent data. This enables a semantic representation of the data by incorporating context information within the JSON payload.	***Data Representation***	Only JSON
Adopts a Linked Data based data model, providing richer semantics and greater interoperability between IoT applications.	***Data Model***	Has a simpler data model than NGSI-LD and does not exploit the Linked Data concept.
Introduces the NGSI-LD Query Language, which allows advanced queries on entity attributes, supporting logical operators and complex conditions.	***Query Language***	Uses a simpler and more straightforward query syntax, without the support of advanced features such as those found in the NGSI-LD Query Language.
Allows greater flexibility in attribute management, including the ability to define nested attributes (subAttribute) and specify metadata for attributes.	***Attribute Management***	Has simpler attribute management than NGSI-LD.
It adopts a more advanced semantic approach through the use of Linked Data, enabling better semantic interoperability between heterogeneous systems.	***Semantic Approach***	It focuses on a lighter and more direct approach without fully adopting the concepts of Linked Data.
Aligned with the standardisation principles of the World Wide Web Consortium (W3C) for linked data.	***Standard***	Based on specifications developed by FIWARE, a consortium of companies and organisations promoting open solutions for data management in the IoT.

TEMA

Thanks!

Q&A

and

DEMO

TEMA

TRUSTED
EXTREMELY PRECISE
MAPPING AND PREDICTION
FOR EMERGENCY
MANAGEMENT

Thank you for your attention!

Matteo Basile (matteo.basile@eng.it)

European Commission

This Project has received funding from the European union's
HE Research and Innovation programme under grant agreement 101093003