Extensions – Linked Data Notifications and NGSI-LD

SEPA training

24th Conference of the Open Innovations Association FRUCT

Moscow, Russia

April 9th-10th, 2019

Luca Roffia

Research follow, Adjunct Professor

Department of Computer Science and Engineering, University of Bologna

luca.roffia@unibo.it

https://site.unibo.it/wot/en

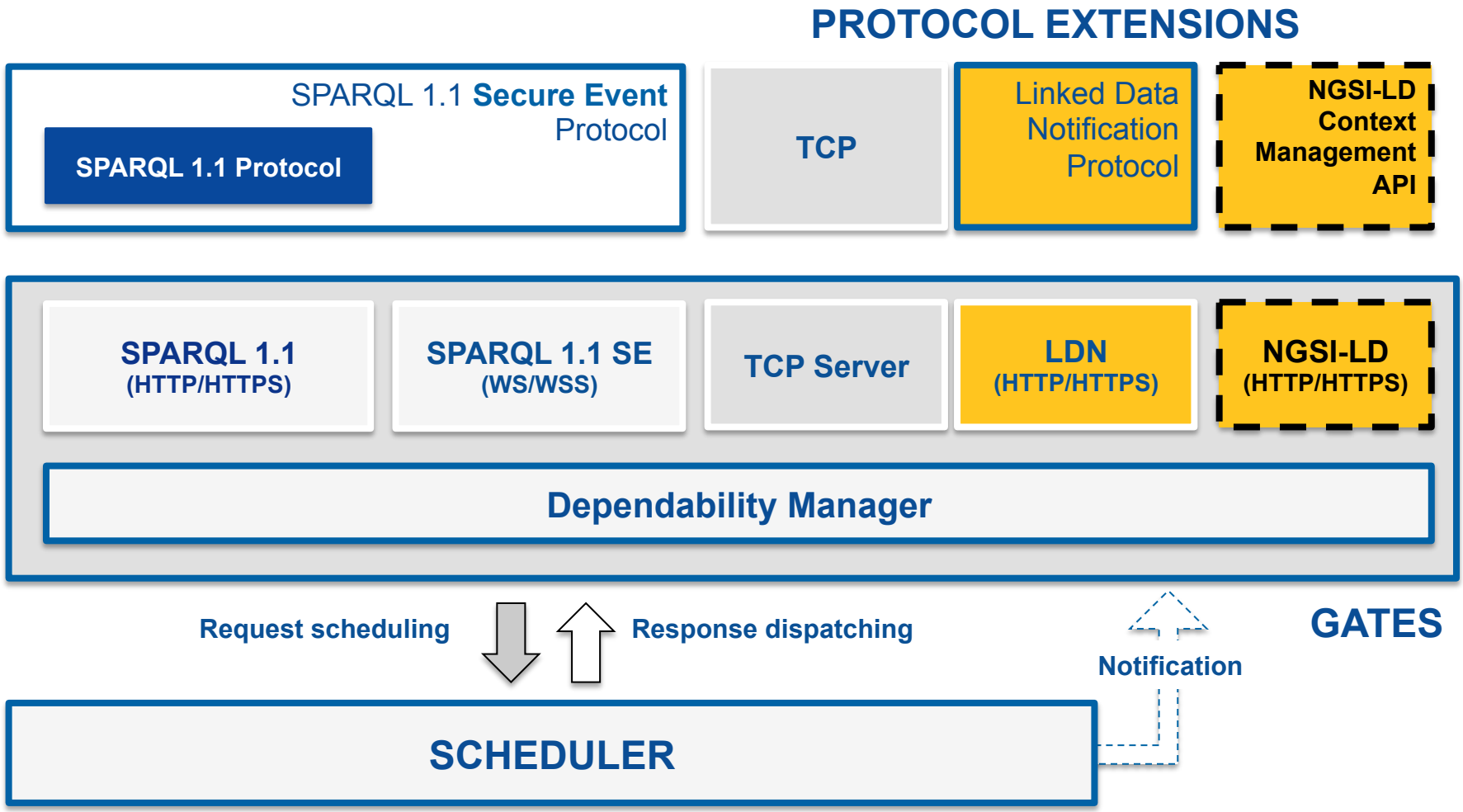
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA ARCES

IEEE

IEEE ComSoc IEEE Communications Society

IEEE Italy Section

SEPA gates


PROTOCOL EXTENSIONS
SPARQL 1.1 Secure Event Protocol
SPARQL 1.1 Protocol
TCP
Linked Data Notification Protocol
NGSI-LD Context Management API
SPARQL 1.1 (HTTP/HTTPS)
SPARQL 1.1 SE (WS/WSS)
TCP Server
LDN (HTTP/HTTPS)
NGSI-LD (HTTP/HTTPS)
Dependability Manager
Request scheduling
Response dispatching
Notification
GATES
SCHEDULER

Linked Data Notifications

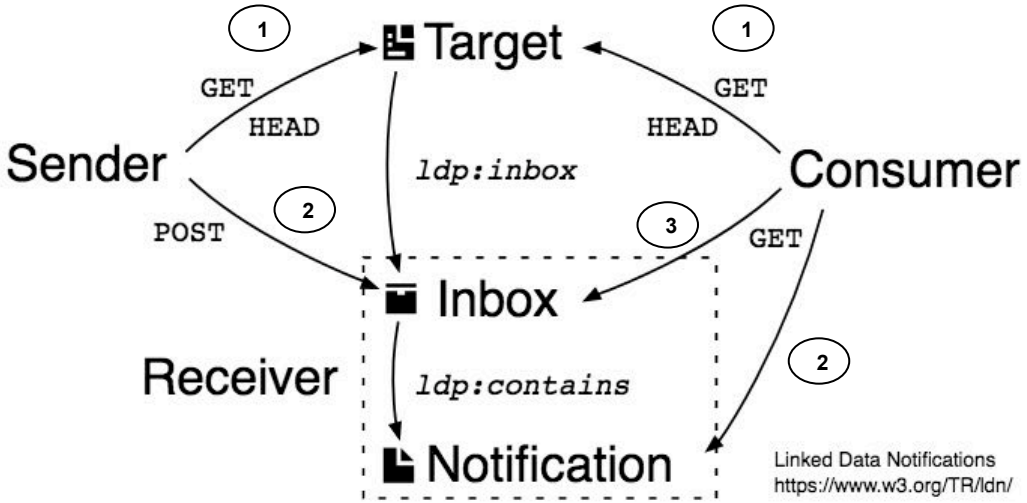
Linked Data Notifications
W3C Recommendation 2 May 2017

https://www.w3.org/TR/ldn/

Notification: an individual entity with its own URI. As such, notifications can be retrieved and reused.

1) The **sender** (***consumer***) chooses a **target resource** to send notifications to (*to receive notifications from*). The **sender** (***consumer***) then discovers the location of the target's **Inbox**
2) The **sender** (***consumer***) sends (*gets*) the **notification** there. Any resource can advertise an Inbox.

3) The **receiver** exposes the notification data (according to appropriate access control) for use by consumers.



Overview of Linked Data Notifications

Inbox discovery (sender/consumer)

Senders and consumers do the following to discover the Inbox URL:

- make an HTTP **HEAD** request on the target URL, and use the **Link** header with a **rel** value of http://www.w3.org/ns/ldp#inbox.

```
HEAD /article HTTP/1.1
Host: example.org
Accept: application/ld+json

HTTP/1.1 200 OK
Link: <http://example.org/inbox/>; rel="http://www.w3.org/ns/ldp#inbox"
```

- make an HTTP **GET** request on the target URL to retrieve an RDF representation, whose encoded RDF graph contains a relation of type http://www.w3.org/ns/ldp#inbox. The subject of that relation is target and the **object** is the **Inbox**.

```
GET /profile HTTP/1.1
Host: example.org
Accept: application/ld+json

HTTP/1.1 200 OK
Content-Type: application/ld+json

{
  "@context": "http://www.w3.org/ns/ldp",
  "@id": "http://example.org/profile",
  "inbox": "http://example.org/inbox/"
}
```

Sending notifications (sender)

Senders who want to send notifications must deliver them through a **POST** request to the Inbox URL. Senders can expect a 201 Created (with a Location Link header) or a 202 Accepted in response to a successful request.

If the request was queued to be processed asynchronously

SENDING EXAMPLE REQUEST

```
POST /inbox/ HTTP/1.1
Host: example.org
Content-Type: application/ld+json;profile="https://www.w3.org/ns/activitystreams"
Content-Language: en

{
  "@context": "https://www.w3.org/ns/activitystreams",
  "@id": "",
  "@type": "Announce",
  "actor": "https://rhiaro.co.uk/#me",
  "object": "http://example.net/note",
  "target": "http://example.org/article",
  "updated": "2016-06-28T19:56:20.114Z"
}
```

JSON-LD notification body

SENDING EXAMPLE RESPONSE

```
HTTP/1.1 201 Created
Location: http://example.org/inbox/5c6ca040
```

the URL from which the notification data can be retrieved

Making Inbox contents available to consumers (consumer)

A successful **GET** request on the **Inbox** must return a HTTP 200 OK with the **URIs of notifications**, subject to the requester's access (returning 4xx error codes as applicable). Receivers may list only URIs of notifications in the Inbox that the consumer is able to access. The Inbox URL must use the http://www.w3.org/ns/ldp#contains predicate to refer to the notifications.

Each notification must be an RDF source.

```
GET /inbox/ HTTP/1.1
Host: example.org
Accept: application/ld+json

HTTP/1.1 200 OK
Content-Type: application/ld+json

{
  "@context": "http://www.w3.org/ns/ldp",
  "@id": "http://example.org/inbox/",
  "contains": [
    "http://example.org/inbox/5c6ca040",
    "http://example.org/inbox/92d72f00"
  ]
}
```

URIs of notifications

GET →

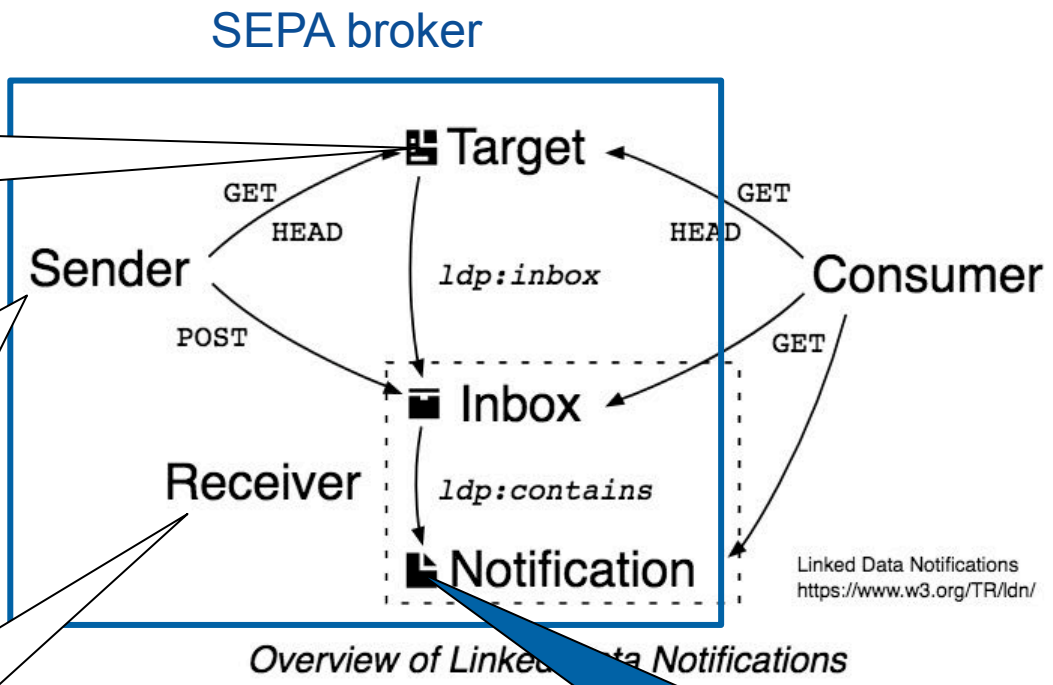
```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "@id": "",
  "@type": "Announce",
  "actor": "https://rhiaro.co.uk/#me",
  "object": "http://example.net/note",
  "target": "http://example.org/article",
  "updated": "2016-06-28T19:56:20.114Z"
}
```

SEPA & LDN

Target: the URL of the LDN resource exposed by a SEPA broker (e.g., /ldn).

Sender: intercepts the notifications sent from the scheduler to the gates derived from a CONSTRUCT based subscription and builds the actual results (removes and adds results).

Receiver: the SEPA broker itself could store notifications in a RDF graph(s).



Overview of Linked ta Notifications

Notification: the updated results of subscriptions based on CONSTRUCT queries ➔ RDF graph ➔ JSON-LD

JSON-LD

JSON-LD 1.0 A JSON-based Serialization for Linked Data
W3C **Recommendation** 16 January 2014 https://www.w3.org/TR/json-ld/

JSON-LD 1.0 Processing Algorithms and API
W3C **Recommendation** 16 January 2014 https://www.w3.org/TR/json-ld-api/

JSON-LD 1.1 A JSON-based Serialization for Linked Data
W3C **Working Draft** 14 December 2018 https://www.w3.org/TR/json-ld11/
W3C **Editor's Draft** 28 March 2019 https://w3c.github.io/json-ld-syntax/

JSON-LD 1.1 Processing Algorithms and API
W3C **Working Draft** 14 December 2018 https://www.w3.org/TR/json-ld11-api/

JSON-LD is a lightweight syntax to serialize **Linked Data** in JSON [RFC8259]. In addition to all the features JSON provides, JSON-LD introduces:

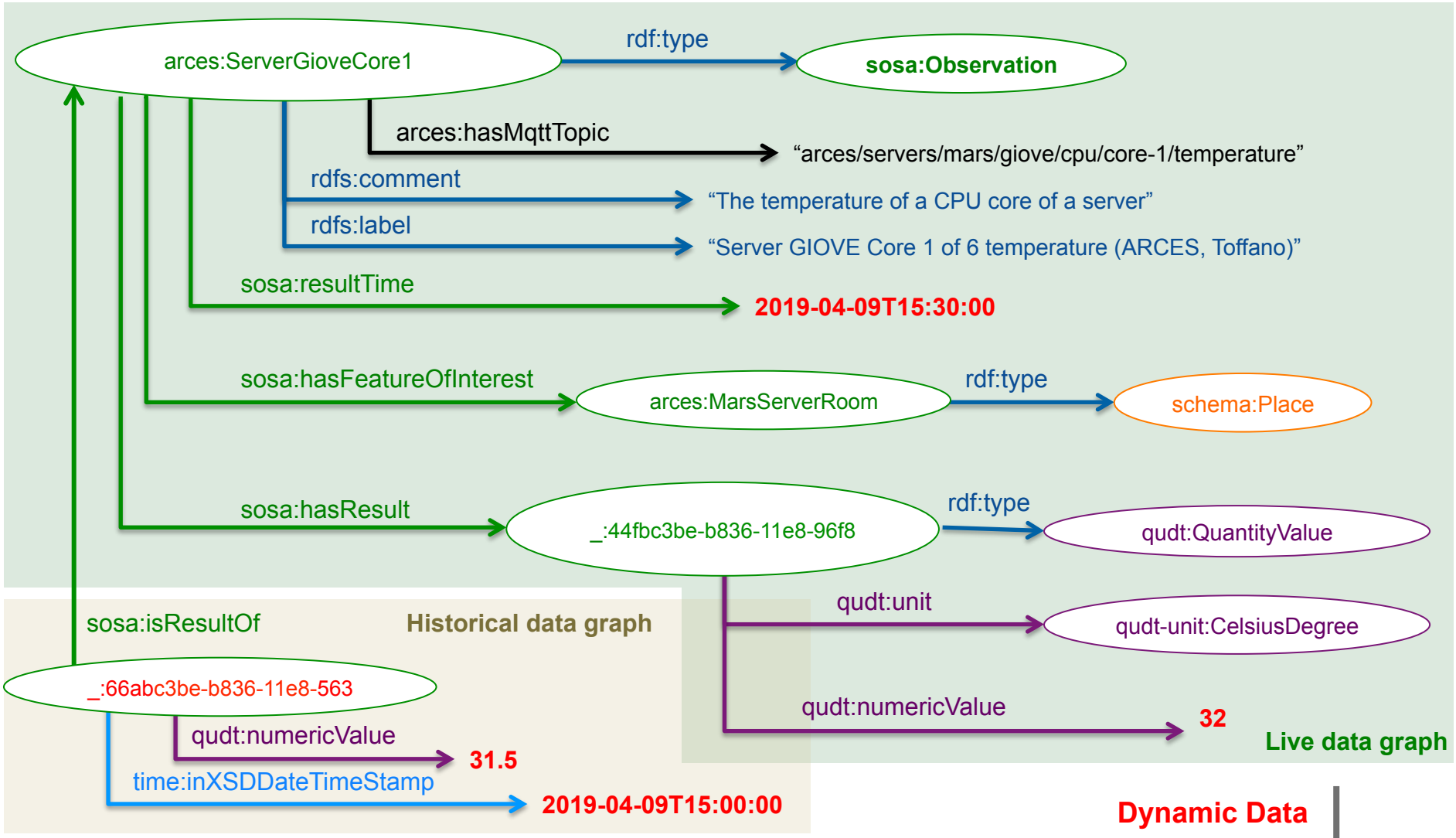
- a universal identifier mechanism for JSON objects via the use of **IRIs**,
- a way to disambiguate **keys** shared among different JSON documents by mapping them to IRIs via a **context**,
- and a facility **to express one or more directed graphs**, such as a social network, in a single document.

JSON-LD document forms

- **Expanded:** expansion is the process of taking a JSON-LD document and applying a context so that the @context is no longer necessary.
- **Compacted**: compaction is the process of applying a provided context to an existing JSON-LD document.
- **Flattened**: flattening is the process of extracting embedded nodes to the top level of the JSON tree, and replacing the embedded node with a reference, creating blank node identifiers as necessary.
- **Framed**: framing is used to shape the data in a JSON-LD document, using an example frame document which is used to both match the flattened data and show an example of how the resulting data should be shaped.

From RDF…

```
"arces": "http://wot.arces.unibo.it/monitor#"
"rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
"rdfs": "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
"sosa": "http://www.w3.org/ns/sosa/"
"qudt": "http://qudt.org/1.1/schema/qudt#"
"qudt-unit": "http://qudt.org/1.1/vocab/unit#"
"schema": "http://schema.org/"
"time": "http://www.w3.org/2006/time#"
```



...to JSON-LD

```
{
  "@context": {
    "arces": "http://wot.arces.unibo.it/monitor#",
    "qudt": "http://qudt.org/1.1/schema/qudt#",
    "qudt-unit": "http://qudt.org/1.1/vocab/unit#",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "schema": "http://schema.org/",
    "sosa": "http://www.w3.org/ns/sosa/",
    "time": "http://www.w3.org/2006/time#",
    "xsd": "http://www.w3.org/2001/XMLSchema#"
  },
  "@graph": [
    {
      "@id": "arces:ServerGioveCore1",
      "@type": "sosa:Observation",
      ”rdfs:comment": "The temperature of a CPU core of a server.",
      ”rdfs:label": "Server GIOVE Core 1 of 6 temperature (ARCES, Toffano)",
      "sosa:hasFeatureOfInterest": {
        "@id": "arces:MarsServerRoom"
      },
      "sosa:hasResult": {
        "@id": "_:44fbc3be-b836-11e8-96f8"
      },
      “sosa:resultTime” : “2019-04-09T15:30:00“
    },
    {
      "@id": "_:44fbc3be-b836-11e8-96f8",
      "@type": "qudt:QuantityValue",
      "qudt:numericValue": "32",
      "qudt:unit": {
        "@id": "qudt-unit:CelsiusDegree"
      }
    },
    {
      "@id": "_:66abc3be-b836-11e8-563",
      "@type": ”qudt:QuantityValue",
      ”sosa:isResultOf": {
        "@id": ”arces:ServerGioveCore1"
      },
      "qudt:numericValue": "31.5",
      "time:inXSDDateTimeStamp": "2019-04-09T15:00:00"
    }
  ]
}
```

NGSI-LD API

F. Viola, F. Antoniazzi, C. Aguzzi, C. Kamienski, L. Roffia, Mapping the NGSI-LD context model on top of a SPARQL event processing architecture: implementation guidelines.

Friday @ 11:00 Auditorium 1

Context Information			
Entitites Provisioning	Create[a]	POST	/entities
	Delete[a,b]	DELETE	/entities/*entityId*
Attributes Provisioning	Append[a]	POST	/entities/*entityId*/attrs
	Update[a]	PATCH	/entities/*entityId*/attrs
	Delete[a]	DELETE	/entities/*entityId*/attrs/*attrId*
	Partial update[a]	PATCH	/entities/*entityId*/attrs/*attrId*
Entitites Consumption	Retrieve[b]	GET	/entities/*entityId*
	Query[b]	GET	/entities
Subscription	Create[a,c]	POST	/subscriptions
	Query[b]	GET	/subscriptions
	Retrieve[b]	GET	/subscriptions/*subscriptionId*
	Update[a,c]	PATCH	/subscriptions/*subscriptionId*
	Delete[a,c]	DELETE	/subscriptions/*subscriptionId*
Context Source			
Registration	Register[a]	POST	/csource
	Update[a]	PATCH	/csource/*registrationId*
	Delete[a]	DELETE	/csource/*registrationId*
Discovery	Query[b]	GET	/csource
	Retrieve[b]	GET	/csource/*registrationId*
Subscription	Create[a,c]	POST	/csourceSubscriptions
	Query[b]	GET	/csourceSubscriptions
	Update[a,c]	PATCH	/csourceSubscriptions/*subscriptionId*
	Retrieve[b]	GET	/csourceSubscriptions/*subscriptionId*
	Delete[a,c]	DELETE	/csourceSubscriptions/*subscriptionId*

(a) Mapped as SPARQL 1.1 Update (b) Mapped as SPARQL 1.1 Query (c) Mapped as SPARQL 1.1 Subscribe

Source: Figure 4.2.3-1, ETSI GS CIM 009 V1.1.1 (2019-01).