FIWARE Global Summit

From Data to Value

OPEN SOURCE
OPEN STANDARDS
OPEN COMMUNITY

# Digital Twin Advanced Programming using NGSI-LD

Federation + Actuation, Multi-types, DatasetId, NGSI-Tenant, NGSI-Scope

Jason Fox, Technical Evangelist, FIWARE Foundation

**Vienna, Austria**
**12-13 June, 2023**
**#FIWARESummit**

FIWARE
Open APIs for Open Minds

# Learning Goals

- Overview advanced deployments
- **Multi-Attributes** with `datasetId` (alternative sources, 1-n relationships)
- **Scopes** (filter based on hierarchies)
- Distributed deployments based on **registrations** (IoT Agents, Context Brokers)
- **Federated deployments** (transparently give applications access to context information from different players running own Context Brokers)
- **Tenants** (use same Context Broker instance, but isolate databases)
- Outlook: New distributed deployment features in NGSI-LD v1.6.1

# Multi-attributes: Use of `datasetId`



FIWARE

# Multi-Attributes with `datasetId` (1)

## Properties

- There may be more than one source for the same information, e.g. speed of the same car from *speedometer*, *navigation system*, *external sensor*

```
"speed": {
    "type" : "Property",
    "value" : 100
}
```

```
"speed": {
    "type" : "Property",
    "value" : 95
}
```

```
"speed": {
    "type" : "Property",
    "value" : 93
}
```

- They all provide the *speed* of the *car* but they may be systematically different, e.g. the speedometer shall never show a speed lower than the actual speed  in practice it will often show a slightly higher speed
- If a single property value in a Context Broker was arbitrarily updated by the different sensors, such values would be quite volatile, especially when looking at a temporal evolution, even if the actual value was relatively stable

 **It is necessary to distinguish between the values from the different sources**

# Multi-Attributes with `datasetId` (2)

## Properties

- The datasetId has the purpose of distinguishing values from different sources

```
"speed": {
    "type" : "Property",
    "value" : 100,
    "datasetId": "urn:speedometer"
}
```

```
"speed": {
    "type" : "Property",
    "value" : 95,
    "datasetId": "urn:gps"
}
```

```
"speed": {
    "type" : "Property",
    "value" : 93,
    "datasetId": "urn:speedcamera"
}
```

- Property values with different **`datasetIds`** are considered distinct values and stored separately
- Property values with a **`datasetId`** can thus be updated independently without affecting values of the same property with a different **`datasetId`**
- Property values without a **`datasetId`** are considered to be the **default value**
- An update without a **`datasetId`** updates the **default value**
- A retrieve / query / notification returns all existing values for the same property – as JSON object if there is a single value, within a container, if there are multiple



FIWARE

# Multi-Attributes with `datasetId` (3)

- Create Entity

```
curl -iX POST \
 '.../ngsi-ld/v1/entities' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id": "urn:HDXX234",
    "type": "Vehicle",
    "speed": {
        "type": "Property",
        "value": 100,
        "unitCode": "KMH",
        "datasetId": "urn:speedometer"
    }
}'
```

… Add further speeds

```
curl -iX POST \
    '.../ngsi-ld/v1/entities/urn:HDXX234/attrs' \
--header 'Content-Type: application/json' \
--data-raw '{
    "speed": [
    {
        "type": "Property",
        "value" : 95,
        "unitCode": "KMH",
        "datasetId": "urn:gps"
    },
    {
        "type": "Property",
        "value": 92,
        "unitCode": "KMH",
        "datasetId": "urn:speedcamera"
    }
    ]
}'
```



FIWARE

# Multi-Attributes with `datasetId` (4)

- Update Speedometer Speed

```
curl --iX PATCH \
    '.../ngsi-ld/v1/entities/urn:HDXX234/attrs' \
--header 'Content-Type: application/JSON' \
--data-raw '{
    "id": "urn:HDXX234",
    "type": "Vehicle",
    "speed": {
        "type": "Property",
        "value": 103,
        "unitCode": "KMH"
        "datasetId": "urn:speedometer"
    }
}'
```

- Query Vehicle **`urn:HDXX234`**

```
curl -iX GET \
     '.../ngsi-ld/v1/entities/urn:HDXX234' \
--header 'Accept: application/json'
```

Result


```
{
    "id": "urn:HDXX234",
    "type": "Vehicle",
    "speed": [
        {
            "type": "Property",
            "datasetId": "urn:gps",
            "value": 95,
            "unitCode": "KMH"
        },
        {
            "type": "Property",
            "datasetId": "urn:speedcamera",
            "value": 92,
            "unitCode": "KMH"
        },
        {
            "type": "Property",
            "datasetId": "urn:speedometer",
            "value": 103,
            "unitCode": "KMH"
        }
    ]
}
```

# Attributes with fixed order values and common meta-data: `ListProperty` and `ListRelationship`

# Multi-Attribute **Relationships** with individual metadata using `datasetId`

## Relationships

- In addition to 1:1 relationships, there are also 1:n relationships
- Each element of a 1:n relationship can be represented with a **`datasetId`**
- Each relationship with a **`datasetId`** may have different sub-attributes

```
"friend": {
    "type" : "Relationship",
    "object" : "John“,
    "datasetId": "urn:001"
}
```

```
"friend": {
    "type" : "Relationship",
    "object" : "Paul",
    "datasetId": "urn:002",
}
```

```
"friend": {
    "type" : "Relationship",
    "object" : "David",
    "datasetId": "urn:003",
    "bestFriend": {
        "type" : "Property",
        "value" : true
    }
}
```

- If David becomes the new best friend, the information can be updated accordingly, using the **`datasetId`** as shown

# But Relationships in NGSI-LD - `datasetId` are currently overloaded

## A simple single Relationship - one metadata

- In the book *Murder on the Orient Express* , the murder victim is **Mr. Ratchett**

```
"victim":{
    "object": urn:sam-rachett, "type": "Relationship", "observedAt": "01-01-2023"
 }
```

## A Relationship, measured in different ways - each element can have different metadata

- In the game *Cluedo*, the murder victim is **Dr. Black** (Waddingtons - GB) or **Mr. Boddy** (Hasbro - US)

```
"victim": [
   { "object": "urn:dr-black", "type": "Relationship",
     "datasetId": "urn:waddingtons", "observedAt" : "01-01-2023"},
   {"object": "urn:mr-boddy", "type": "Relationship",
     "datasetId": "urn:hasbro", "observedAt" : "01-01-2023"}
]
```

## 1-N Relationships, with multiple objects - one common metadata and fixed order

- In the game *Cluedo*, the suspects are: **Miss Scarlett**, **Reverend Green**, **Colonel Mustard**, **Professor Plum**, **Mrs. Peacock** and **Mrs. White**

# Answer use - **ListRelationship** in various formats

- In the game *Cluedo*, the suspects are: **Miss Scarlett**, **Reverend Green**, **Colonel Mustard**, **Professor Plum**, **Mrs. Peacock** and **Mrs. White**

## **Normalized -** an ordered 1-N relationship

```
"suspects": {
   "objectList": ["urn:miss-scarlett","urn:rev-green","urn:col-mustard",
                  "urn:prof-plum", "urn:mrs-peacock","urn:mrs-white"],
   "type": "ListRelationship"
}
```

## **Concise -** presence of **`objectList`** is sufficient to indicate an ordered 1-N relationship

```
"suspects": {
   "objectList": ["urn:miss-scarlett","urn:rev-green","urn:col-mustard",
                  "urn:prof-plum", "urn:mrs-peacock","urn:mrs-white"]
}
```

## **Simplified -** list of URNs is just returned as an Array. **Lossy**

```
"suspects": ["urn:miss-scarlett","urn:rev-green","urn:col-mustard",
             "urn:prof-plum", "urn:mrs-peacock","urn:mrs-white"]
```

FIWARE

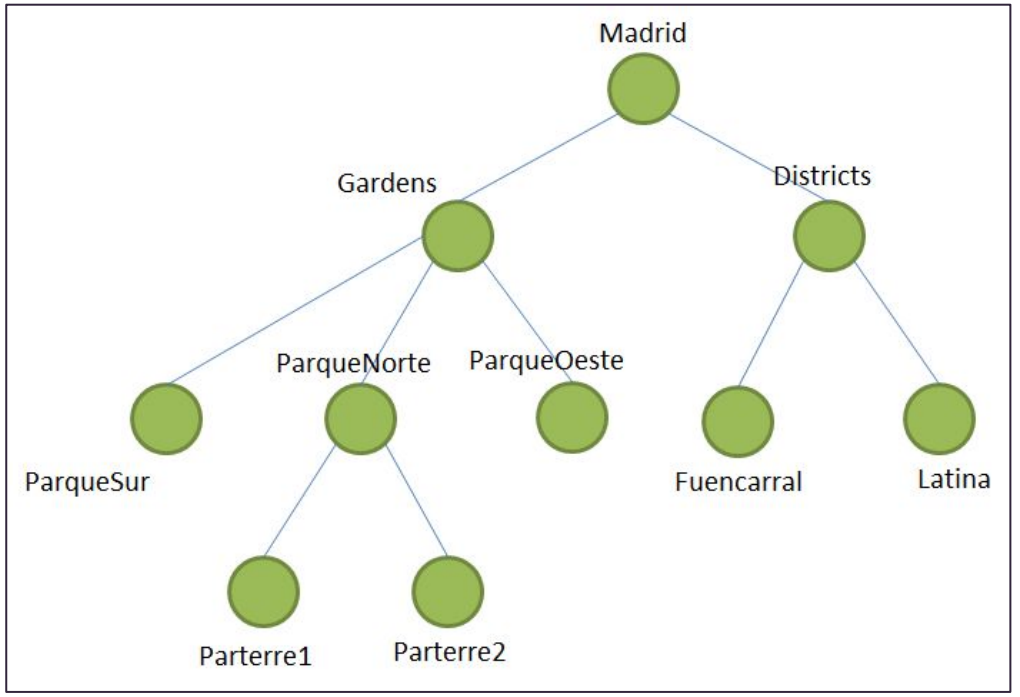
# Logical Data Separation of Entities: use of **scopeQ**



FIWARE

# NGSI-LD Scopes (1)

- **Hierarchical structures** play an important role for **structuring and organizing our world**, e.g. hierarchical location structures (example on the left) or organizational structures (e.g. of a company)
- The special ***scope*** property allows giving an entity a hierarchical scope, e.g. *ParqueSur* in the example on the left
- In queries and subscriptions, scopes can be used for filtering with the ***scopeQ*** parameter, e.g. `scopeQ="/Madrid/Gardens/ParqueSur"`, which would only match entities that have this particular scope.




**Note: NGSI-LD Scopes are the successor of NGSIv2 Fiware-ServicePath, but there are important differences**

# NGSI-LD Scopes (2)

- Entities can have **multiple** hierarchical scopes attached, e.g.
- Scopes are **optional** and **independent of the Entity ID**
- Scopes can be used to **scope queries and subscriptions**
- The scope query language (**scopeQ**) allows **wildcards**
  - **+** for an arbitrary value of one hierarchy level, e.g. `scopeQ="/CompanyA/+/HR"`, for matching the HR groups of all departments of CompanyA
  - **#** matches the given scope and the whole hierarchy of scopes below, e.g. `scopeQ="/Madrid/Gardens/#` matches all entities with scope ***/Madrid/Gardens*** or any of its direct or indirect sub-scopes
  - ***/#*** matches all entities that have any explicit scope attached
- The scope query language (**scopeQ**) also has logical *and* and *or* operators, e.g. `scopeQ="/Madrid/District/Centro,/Madrid/District/Cortes"` for all entities whose scope is ***/Madrid/District/Centro*** or ***/Madrid/District/Cortes***
- Scopes of entities can be updated
- Scopes can be registered as part of registrations

```
"scope":[
    "/Madrid/District/Latina",
    "/CompanyA/DepartmentX/UnitC"
]
```



FIWARE

# NGSI-LD Scopes (3)

```
curl -iX POST \
    '.../ngsi-ld/v1/entities' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id": "urn:HDMY185",
    "type": "Vehicle",
    "scope":[
        "/Germany/Heidelberg",
        "/CompanyA/DepartmentX/UnitC"
    ],
    "speed": {
        "type": "Property",
        "value": 32,
        "datasetId": "urn:speedometer"
    }
}'
```

```
curl -iX POST \
   '.../ngsi-ld/v1/entities' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id": "urn:BFF2016",
    "type": "Vehicle",
    "scope":[
        "/Germany/Berlin/Charlottenburg",
        "/CompanyA/DepartmentX"
    ],
    "speed": {
        "type": "Property",
        "value": 32,
        "datasetId": "urn:speedometer"
    }
}'
```



FIWARE

# NGSI-LD Scopes (4)

```
curl -iX GET \
'.../ngsi-ld/v1/entities?type=Vehicle&scopeQ=/CompanyA' \
--header 'Accept: application/json'
```

```
[]
```

```
curl -iX GET \
 '.../ngsi-ld/v1/entities?type=Vehicle&scopeQ=/CompanyA/%23' \
--header 'Accept: application/json' \
```

URL encoded Hash

```
[
    {
        "id": "urn:BFF2016",
        "type": "Vehicle",
        "scope": [
            "/Germany/Berlin/Charlottenburg",
            "/CompanyA/DepartmentX"
        ],
…
    },
    {
        "id": "urn:HDMY185",
        "type": "Vehicle",
        "scope": [
            "/Germany/Heidelberg",
            "/CompanyA/DepartmentX/UnitC"
        ],
…
    }
]
```

# Multi-tenancy: use of the `NGSILD-Tenant` header



FIWARE

# Tenants (1)

- If the same context broker instance is to be used for multiple different user (groups), their context information has to be isolated from each other
- The concept for this is called multi-tenancy and each user (group) is associated with one tenant
- NGSI-LD implementations can optionally support multitenancy. In all requests a tenant can optionally be specified.
- In the HTTP binding this is done through the HTTP header “NGSILD-Tenant”
- NGSILD-Tenant is the successor of Fiware-Service in NGSIv2

# Tenants (2)

- The support for *tenants* can be implicit, i.e. a tenant is created when first used in the context of a create (entity, subscription, registration) operation. There are currently no API operations for explicitly creating or deleting tenants.
- Principle: if no tenant is specified, there is always a "default tenant" (which does not have to be created and does not necessarily have to have any information to provide).
- How tenants are implemented, i.e. how isolation is achieved, is up to the implementation.
- Registrations target a tenant (if not, the "default tenant" is assumed) – if the same context source / broker is to be registered for multiple tenants, multiple registrations are needed.

# Accessing Different Tenants

## Default Tenant

```
curl -iX POST \
    '.../ngsi-ld/v1/entities' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id": "urn:SZZ546",
    "type": "Vehicle",
    "speed": {
        "type": "Property",
        "value": 32,
    }
}'
```

```
curl -iX GET \
    '.../ngsi-ld/v1/entities?type=Vehicle' \
--header 'Accept: application/json'
```

```
[{
    "id": "urn:SZZ546",
    "type": "Vehicle",
    "speed": {
       "type": "Property",
       "value": 32
    }
}]
```

## A named Tenant called "FIWARE"

```
curl -iX POST \
    '.../ngsi-ld/v1/entities' \
--header 'Content-Type: application/json' \
--header 'NGSILD-Tenant: FIWARE' \
--data-raw '{
    "id": "urn:MYY456",
    "type": "Vehicle",
    "speed": {
        "type": "Property",
        "value": 66
    }
}'
```

```
curl -iX GET \
    '.../ngsi-ld/v1/entities?type=Vehicle' \
--header 'Accept: application/json' \
--header 'NGSILD-Tenant: FIWARE' \
```

```
[{
    "id": "urn:MYY456",
    "type": "Vehicle",
    "speed": {
        "type": "Property",
        "value": 66
    }
}]
```

# Federations, Actuations and Advanced Deployments

## Including new Registration features in NGSI-LD 1.6.1

https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.06.01_60/gs_CIM009v010601p.pdf



FIWARE

# Overview: Advanced Deployments

- Simple deployments typically:
  - are controlled by a single operator that can guarantee consistency
  - handle a few hundred entities
  - are used by a single application or a small set of applications

… and can be handled by a single Broker / single database

- Advanced deployments may:
  - handle thousands or even millions of entities (scalability)
  - are controlled by multiple independent operators
  - have heterogeneous sources of overlapping information
  - require that only some of the information is shared
  - are used by many different applications

… and require interaction between many Brokers and/or aggregation of information from multiple data sources

**NGSI-LD has features to support advanced deployments**



FIWARE

# Distributed deployments

- NGSI-LD supports distributed deployments with multiple context sources, which can also be full context brokers
- Context brokers, but also IoT agents or other context sources (partially) implementing the NGSI-LD API, can register with the context registry what information they can provide
- The context broker uses the information from the context registry to access and aggregate information to be returned to the requesting
- As the next level of context brokers can again have context brokers registered, whole hierarchies of context brokers can be created, possibly reflecting company structures or geographical structures



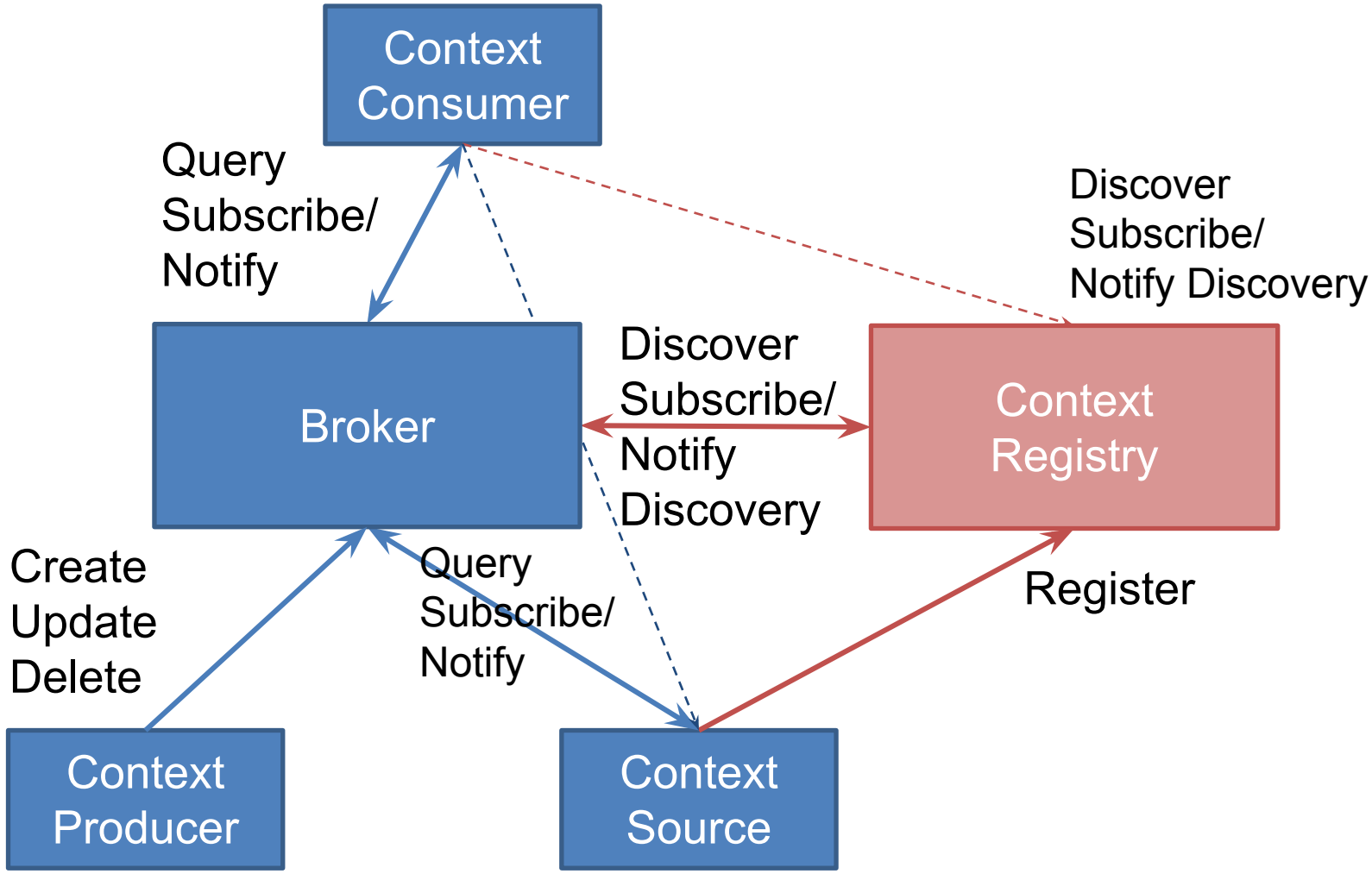
FIWARE

# Federated deployments

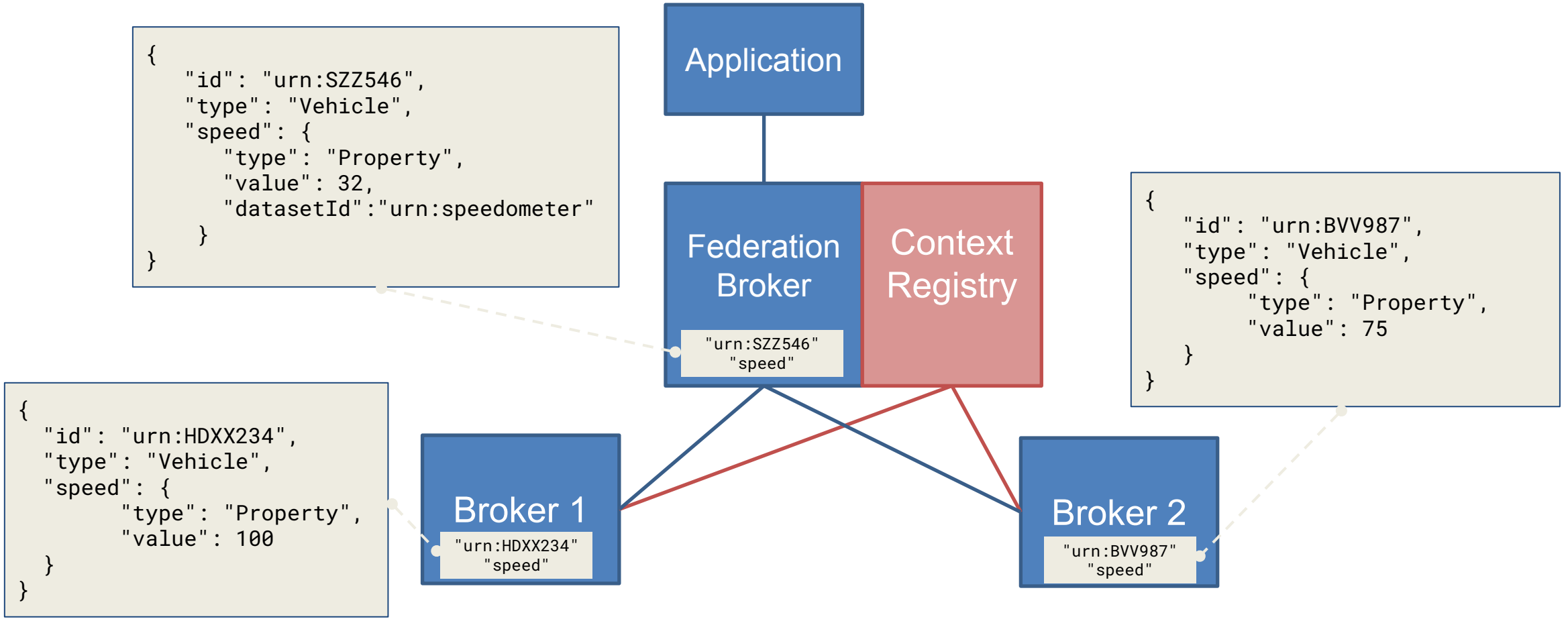
- Federated deployments are distributed deployments that transparently give applications access to context information from different players running own context brokers
  - a federated scenario is not technically different from other distributed scenarios, but the assumption is that it goes across administrative boundaries, there is no central control that can be assumed

- In federated scenarios, the focus typically is on accessing (and possibly aggregating) context information from multiple context brokers, while the management of the information (create, update, delete) is done locally in each domain.

# NGSI-LD Architectural Roles



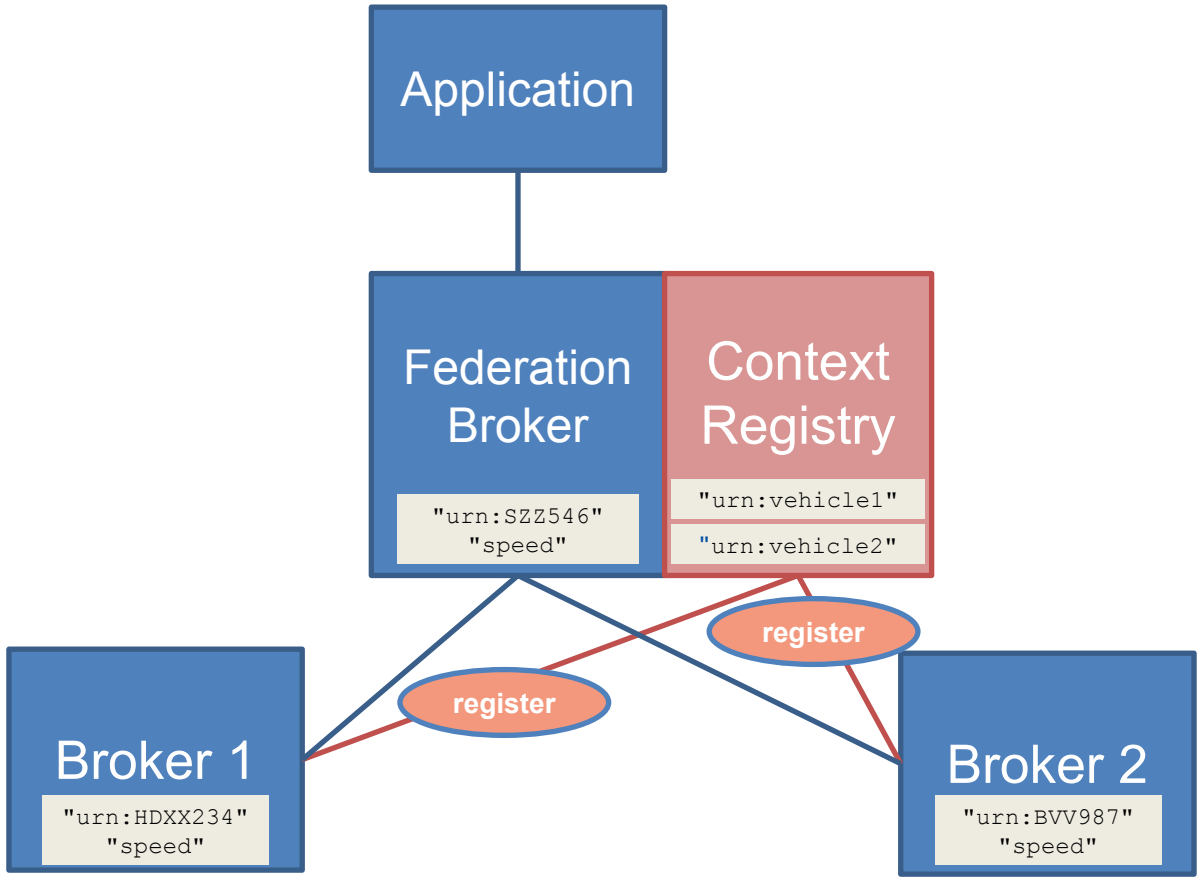
# Federation and Single Request for **different** Entities (1)



# Federation and Single Request for **different** Entities (2)

```
curl -iX POST \
    '.../ngsi-ld/v1/csourceRegistrations' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id": "urn:ngsi-ld:ContextSourceRegistration:vehicle1",
    "type": "ContextSourceRegistration",
    "information": [{
            "entities": [{
                "type": "Vehicle"
            }]
        }
    ],
    "endpoint": "http://broker1:9090"
}'
```

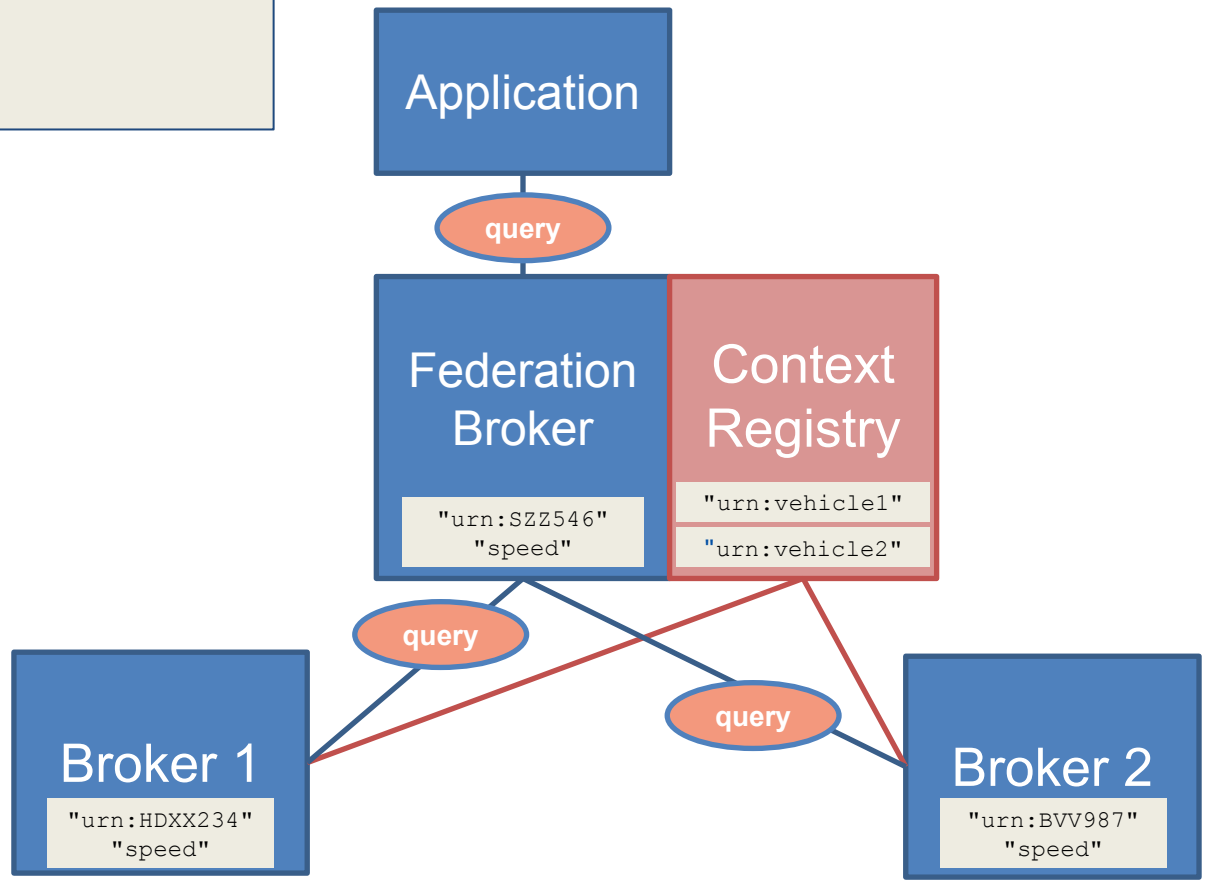
```
curl iX POST \
   '.../ngsi-ld/v1/csourceRegistrations' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id": "urn:ngsi-ld:ContextSourceRegistration:vehicle2",
    "type": "ContextSourceRegistration",
    "information": [{
            "entities": [{
                "type": "Vehicle"
            }]
        }
    ],
    "endpoint": "http://broker2:9090"
}'
```



# Federation and Single Request for **different** Entities (3)

```
curl -iX GET \
    '.../ngsi-ld/v1/entities?type=Vehicle' \
--header 'Accept: application/json'
```

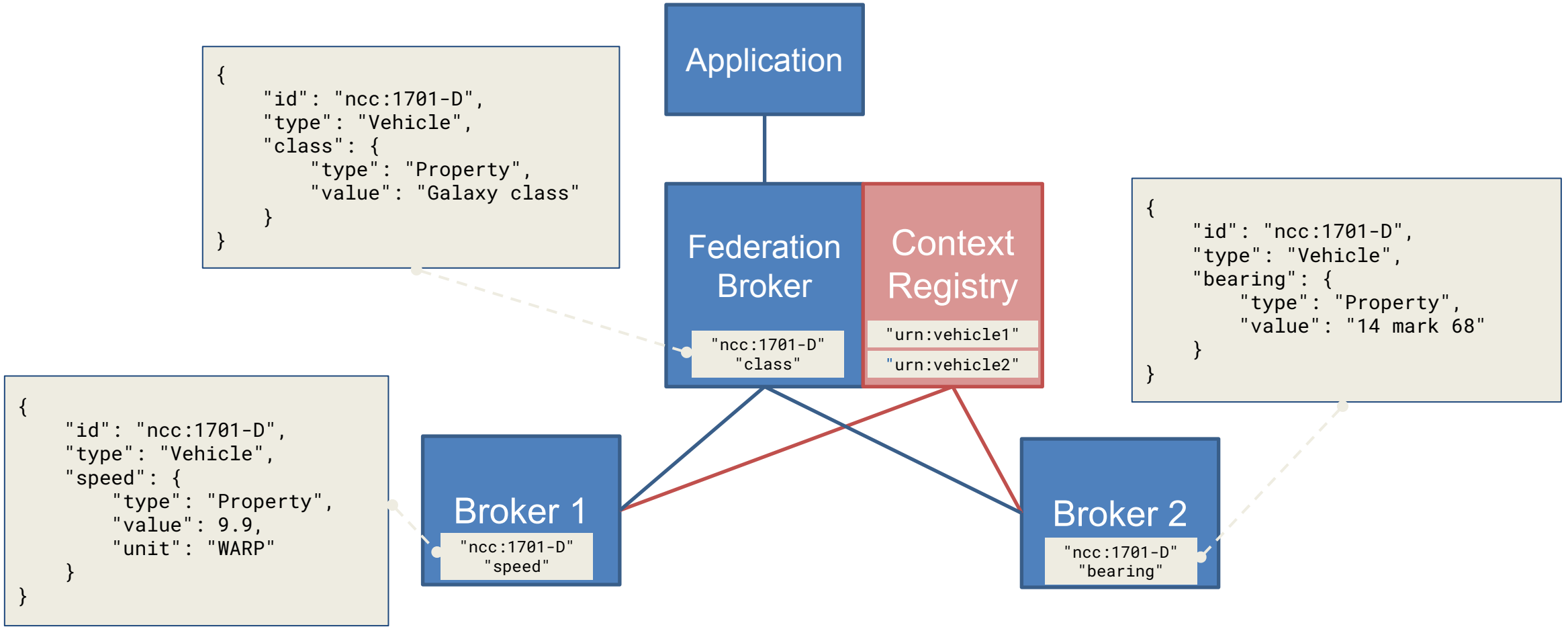
```
[
    {
        "id": "urn:BVV987",
        "type": "Vehicle",
        "speed": {
            "type": "Property",
            "datasetId": "urn:speedometer",
            "value": 75
        }
    },
    {
        "id": "urn:SZZ546",
        "type": "Vehicle",
        "speed": {
            "type": "Property",
            "datasetId": "urn:speedometer",
            "value": 32
        }
    },
    {
        "id": "urn:HDXX234",
        "type": "Vehicle",
        "speed": {
            "type": "Property",
            "value": 100
        }
    }
]
```






FIWARE

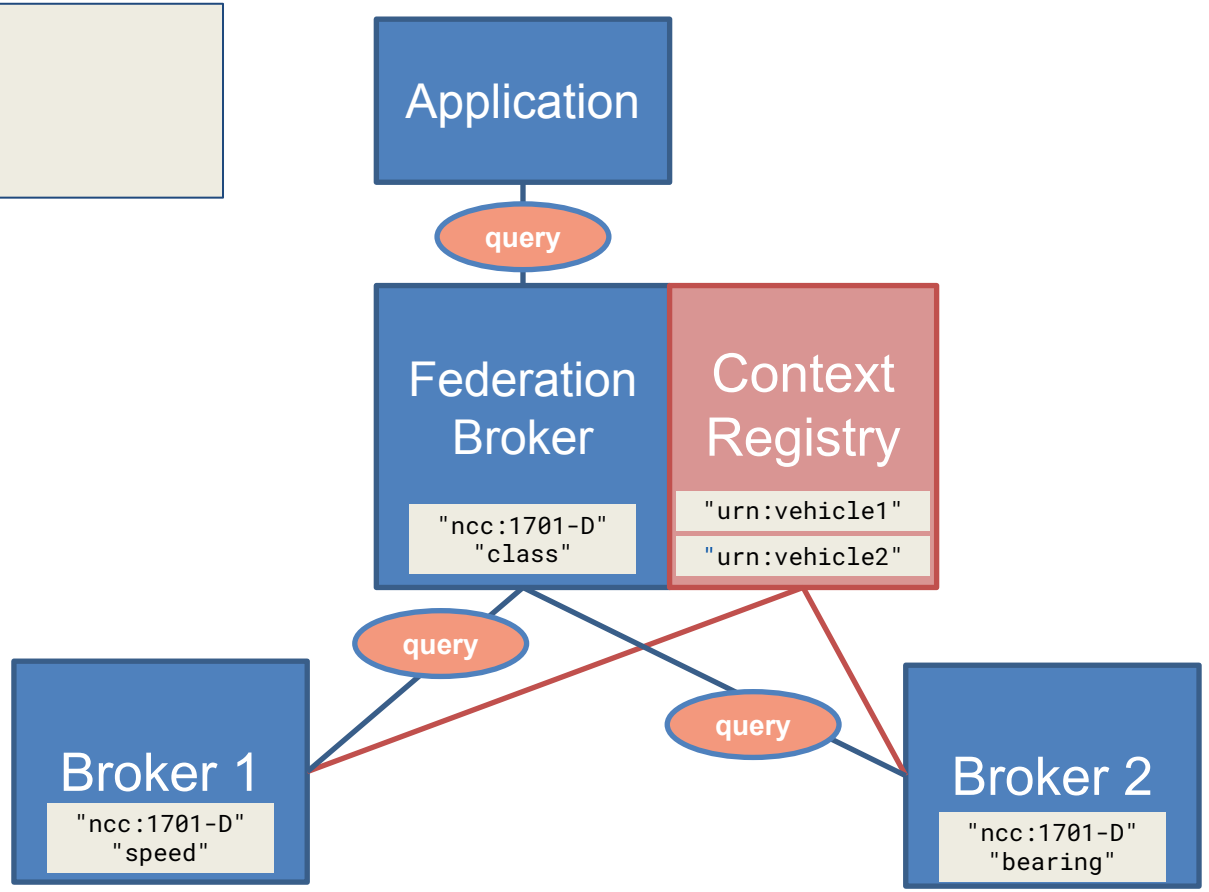
# Federation and Single Request for **the same** Entity (1)



# Federation and Single Request for **the same** Entity (2)

```
curl -iX GET \
   '.../ngsi-ld/v1/entities?type=Vehicle' \
--header 'Accept: application/json'
```


```
[
    {
        "id": "ncc:1701-D",
        "type": "Vehicle",
        "class": {
            "type": "Property",
            "value": "Galaxy class"
        },
        "bearing": {
            "type": "Property",
            "value": "14 mark 68"
        },
        "speed": {
            "type": "Property",
            "value": 9.9,
            "unitCode": "WARP"
        }
    }
]
```

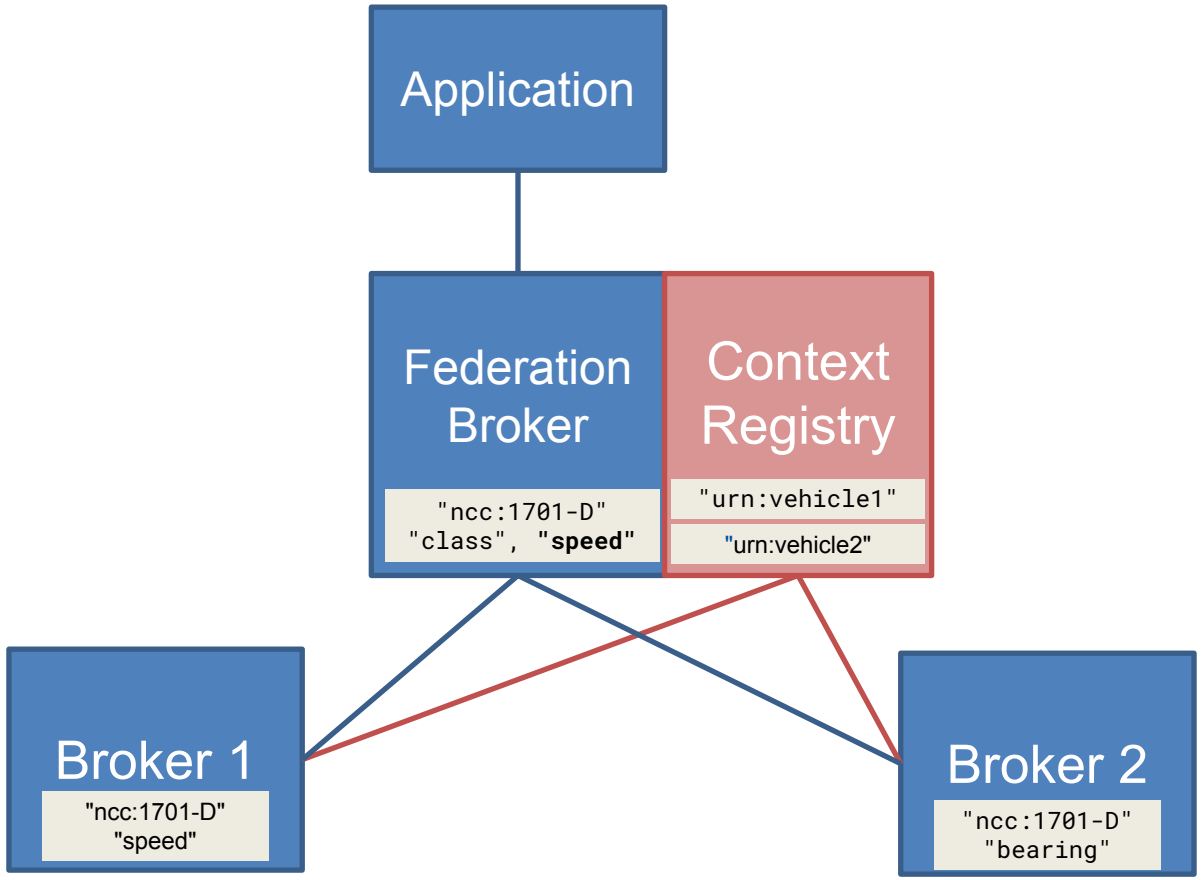



# Federation and Single Request for **the same** Entity (3)

Add more values to speed property in Federation Broker, using **datasetId**

```
curl --location --request POST 'http://localhost:909
0/ngsi-ld/v1/entities/ncc:1701-D/attrs' \
--header 'Content-Type: application/json' \
--data-raw '{
    "id": "ncc:1701-D",
    "type": "Vehicle",
    "speed": [{
        "type": "Property",
        "value": 21.468,
        "unit": "c",
        "datasetId": "urn:inspeedoflight"
    },
    {
        "type": "Property",
        "value": 6500000000,
        "unitCode": "KMH",
        "datasetId": "urn:inkmh"
    }]

}'
```



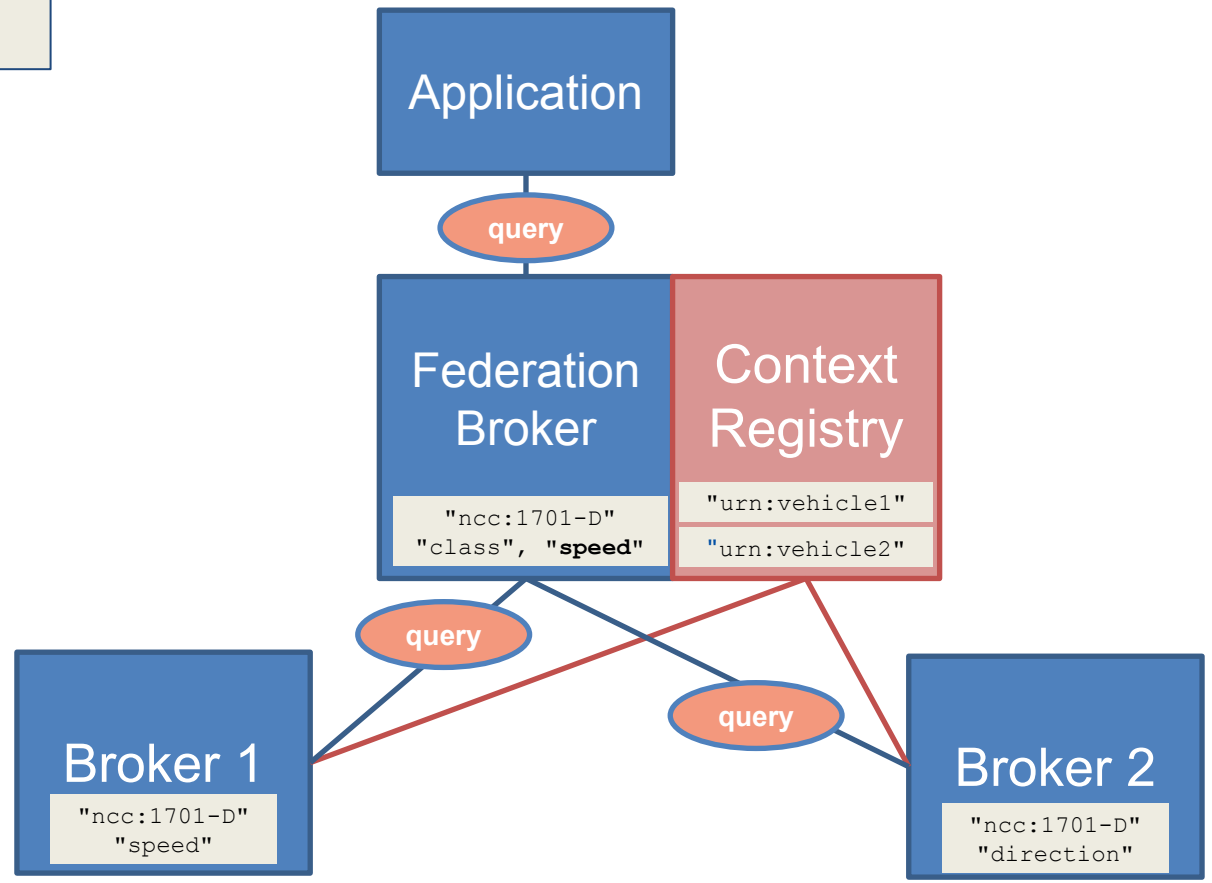
# Federation and Single Request for **the same** Entity (4)

```
curl --location --request \
GET 'http://localhost:9090/ngsi-ld/v1/entities?type
=Vehicle' \
--header 'Accept: application/json'
```

```
[
    {
        "id": "ncc:1701-D",
        "type": "Vehicle",
…
        "speed": [
            {
                "type": "Property",
                "value": 9.9,
                "unitCode": "WARP"
            },
            {
                "type": "Property",
                "datasetId": "urn:inspeedoflight",
                "value": 21.468,
                "unit": "c"
            },
            {
                "type": "Property",
                "datasetId": "urn:inkmh",
                "value": 6500000000,
                "unitCode": “KMH"
            }
        ]
    }
]
```

Broker 1

Federation Broker






FIWARE

# Tenants in Distributed/Federated Cases: Tenant Mapping in the Registry

- As for other operations, a tenant can be specified in registry operations
- Due to isolation between tenants, there are de facto separate registries per tenant (realization is implementation-dependent)
- The available tenants in registered Context Sources or Brokers may be different from those in the Broker to which the registry belongs therefore a mapping is required
- Thus: optional tenant information can be added to registrations (see next slide)
- Brokers then use this tenant information from the registrations when interacting with the registered Context Sources / Brokers (not the tenant information provided in the original request)

# Tenant Information in CsourceRegistrations

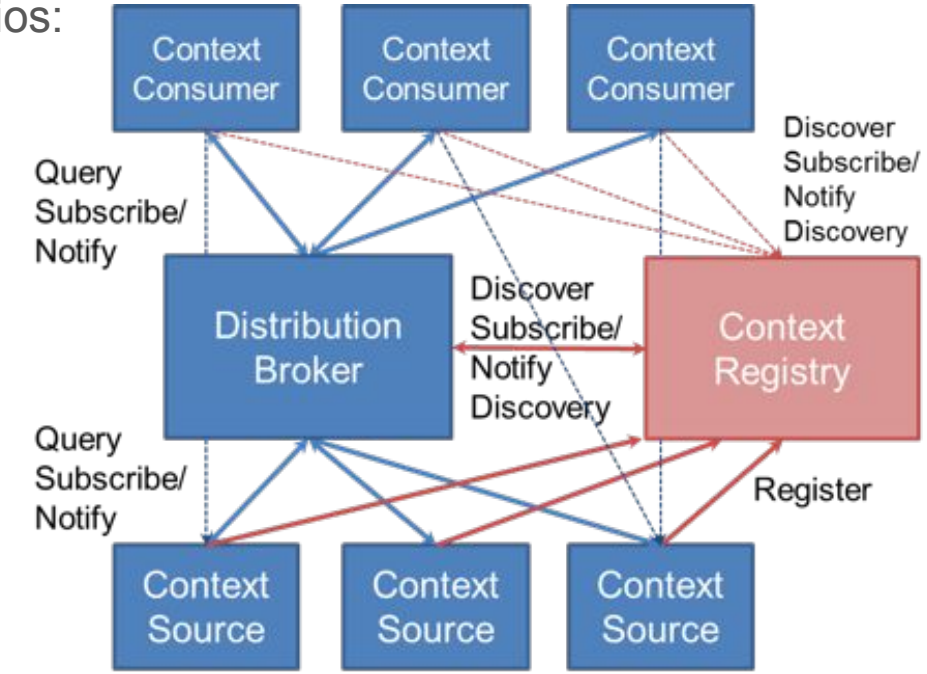
| Name | Data type | Restriction | Cardinality | Description |
|---|---|---|---|---|
| id | URI | At creation time, If it is not provided, it will be assigned during registration process and returned to client.<br>It cannot be later modified in update operations | 0..1 | Unique registration identifier. (JSON-LD @id). There may be multiple registrations per Context Source, i.e. the id is unique per registration |
| type | string | "ContextSource Registration" | 1 | JSON-LD @type<br>Use reserved type for identifying Context Source Registration |
| name | string | Non-empty string | 0..1 | A name given to this Context Source Registration |
| description | string | Non-empty string | 0..1 | A description of this Context Source Registration |
| Information | RegistrationInfo[] | See data type definition in clause 5.2.10. Empty array (0 length) is not allowed | 1 | Describes the Entities, Properties and Relationships for which the Context Source may be able to provide information |
| tenant | **string** | | **0..1** | **Identifies the tenant that has to be specified in all requests to the Context Source that are related to the information registered in this Context Source Registration. If not present, the default tenant is assumed. Should only be present in systems supporting multi-tenancy.** |
| […] | […] | […] | […] | […] |

# Beyond Federations: New distributed deployment features

Focus up to here has been on **federated cases** of context broker to context broker operations, which is defined as the **default** case for NGSI-LD

However, there are also other common distributed deployment scenarios:

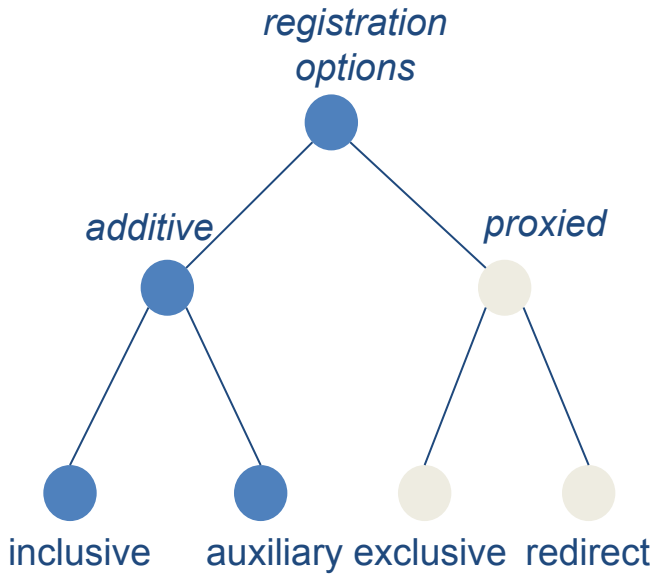
- Support for **actuation** (IoT Agents)
- Support for ***"lazy"* attributes** (IoT Agents)
- **Explicit distribution of context information**, e.g. one context source for vehicles, another for bicycles
- **Backup context sources** e.g. less frequently updated
- Complex Data Sharing scenarios




These require a different **subset of operations** to the standard federation case – a full context broker is frequently not required

- Security concerns may be explicitly deny certain operations
- In other cases, some entities or attributes may only be obtained from a **single specific context source** (e.g. device actuation)
- Others could be augmented from multiple sources with limited availability

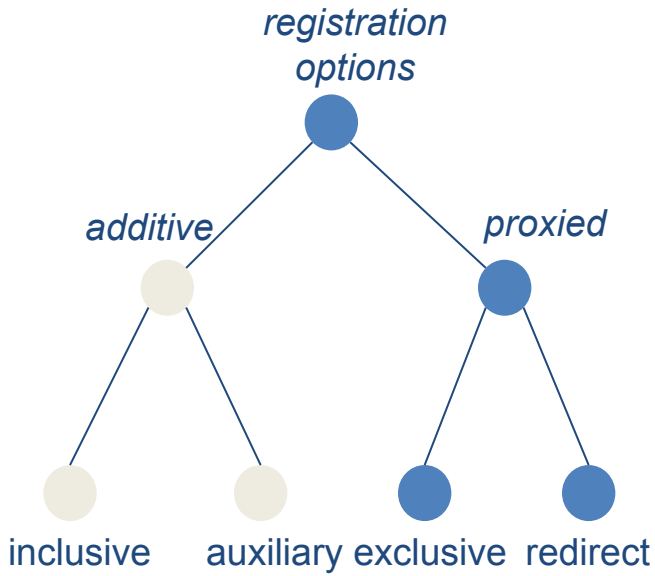
# Distributed Operation Modes (1)



# Distributed Operation Modes (2)

## Proxied Registrations

A Context Broker is **not permitted** to hold context data about the Entities and Attributes locally itself. All context data is obtained from the external registered sources.




- An **exclusive** Context Source Registration specifies that all of the registered context data is held in a single location external to the *Context Broker.* The *Context Broker* itself holds no data locally about the registered Attributes and no overlapping proxied Context Source Registrations shall be supported for the same combination of registered Attributes on the Entity.

  An exclusive registration **must be fully specified**. It always relates to specific Attributes found on a single Entity. It can be used for actuations

- A **redirect** Context Source Registration also specifies that the registered context data is held in a location external to the *Context Broker*, but potentially multiple distinct *redirect* registrations can apply at the same time.

# Limiting Distributed Operations: `local`

- In order to avoid cascading distributed operations, the query parameter **`local`** can be used to limit all operations to be executed on locally available information only. (Clause 6.3.18)

- Context Sources can also put **localOnly** in the *management* info, provided in the Context Source Registration to indicate that, for all requests forwarded to it, **`local`** should be set.

| Name | Data Type | Cardinality | Remarks |
|---|---|---|---|
| local | boolean | 0..1 | If local=true then no Context Source Registrations shall be considered as matching to avoid cascading distributed operations (see clause 4.3.6.4) |

# Context Source Registration

| | | | | |
|---|---|---|---|---|
| mode | String | It shall be one of: "inclusive", "exclusive", "redirect" or "auxiliary"<br><br>The mode is assumed to be "inclusive" if not explicitly defined | 0..1 | The definition of the mode of distributed operation (see clause 4.3.6) supported by the registered Context Source |
| operations | String[] | Entries are limited to the named API operations and named operation groups (see clause 4.20) | 0..1 | The definition limited subset of API operations supported by the registered Context Source<br><br>If undefined, the default set of operations is "federationOps" (see clause 4.20) |
| refreshRate | String | String representing a duration in ISO 8601 format [17] | 0..1 | An indication of the likely period of time to elapse between updates at this registered endpoint.<br><br>Brokers may optionally use this information to help implement caching. |
| management | Registration Management Info | See data type definition in clause 5.2.34 | 0..1 | Holds additional optional registration management information that can be used to limit unnecessary distributed operation requests. |

# Supported Distributed Operations in Registration

## Individual Operations

| | Operation name | Implements |
|---|---|---|
| Context Information Provision | createEntity | 5.6.1 Create Entity |
| | updateEntity | 5.6.2 Update Entity Attributes |
| | appendAttrs | 5.6.3 Append Entity Attributes |
| | updateAttrs | 5.6.4 Partial Attribute update |
| | deleteAttrs | 5.6.5 Delete Entity Attribute |
| | deleteEntity | 5.6.6 Delete Entity |
| | createBatch | 5.6.7 Batch Entity Creation |
| | upsertBatch | 5.6.8 Batch Entity Creation or Update (Upsert) |
| | updateBatch | 5.6.9 Batch Entity Update |
| | deleteBatch | 5.6.10 Batch Entity Delete |
| | upsertTemporal | 5.6.11 Create or Update Temporal Representation of an Entity |
| | appendAttrsTemporal | 5.6.12 Add Attributes to Temporal Representation of an Entity |
| | deleteAttrsTemporal | 5.6.13 Delete Attributes from Temporal Representation of an Entity |
| | updateAttrsTemporal | 5.6.14 Partial Update Attribute instance in Temporal Representation of an Entity |
| | deleteAttrInstanceTemporal | 5.6.15 Delete Attribute Instance from Temporal Representation of an Entity |
| | deleteTemporal | 5.6.16 Delete Temporal Representation of an Entity |
| | mergeEntity | 5.6.17 Merge Entity |
| | replaceEntity | 5.6.18 Replace Entity |
| | replaceAttrs | 5.6.19 Attribute Replace |
| | mergeBatch | 5.6.20 Batch Entity Merge |
| Context Information Consumption | retrieveEntity | 5.7.1 Retrieve Entity |
| | queryEntity | 5.7.2 Query Entities (excluding batch entity queries) |
| | queryBatch | 5.7.2 Query Entities (batch entity queries only) |
| | retrieveTemporal | 5.7.3 Retrieve Temporal Evolution of an Entity |
| | queryTemporal | 5.7.4 Query Temporal Evolution of Entities |
| | retrieveEntityTypes | 5.7.5 Retrieve Available Entity Types |
| | retrieveEntityTypeDetails | 5.7.6 Retrieve Details of Available Entity Types |
| | retrieveEntityTypeInfo | 5.7.7 Retrieve Available Entity Type Information |
| | retrieveAttrTypes | 5.7.8 Retrieve Available Attributes |
| | retrieveAttrTypeDetails | 5.7.9 Retrieve Details of Available Attributes |
| | retrieveAttrTypeInfo | 5.7.10 Retrieve Available Attribute Information |
| Context Information Subscription | createSubscription | 5.8.1 Create Subscription |
| | updateSubscription | 5.8.2 Update Subscription |
| | retrieveSubscription | 5.8.3 Retrieve Subscription |
| | querySubscription | 5.8.4 Query Subscription |
| | deleteSubscription | 5.8.5 Delete Subscription |

**NGSI-LD v1.6.1**

## Pre-defined Operation Groups

| Operation Group name | Implements |
|---|---|
| federationOps<br>**default** | • retrieveEntity<br>• queryEntity<br>• retrieveEntityTypes<br>• retrieveEntityTypeDetails<br>• retrieveEntityTypeInfo<br>• retrieveAttrTypes<br>• retrieveAttrTypeDetails<br>• retrieveAttrTypeInfo<br>• createSubscription<br>• updateSubscription<br>• retrieveSubscription<br>• querySubscription<br>• deleteSubscription |
| updateOps | • updateEntity<br>• updateAttrs<br>• replaceEntity<br>• replaceAttrs |
| retrieveOps | • retrieveEntity<br>• queryEntity |

| Operation Group name | Implements |
|---|---|
| redirectionOps | • createEntity<br>• updateEntity<br>• appendAttrs<br>• updateAttrs<br>• deleteAttrs<br>• deleteEntity<br>• mergeEntity<br>• replaceEntity<br>• replaceAttrs<br>• retrieveEntity<br>• queryEntity<br>• retrieveEntityTypes<br>• retrieveEntityTypeDetails<br>• retrieveEntityTypeInfo<br>• retrieveAttrTypes<br>• retrieveAttrTypeDetails<br>• retrieveAttrTypeInfo |

⬅ **Federations**

⬅ **Actuators**

⬅ ***"lazy"* Attributes**

⬅ **CRUD Proxies**



FIWARE

# Optional Registration Management Info

| Name | Data Type | Restrictions | Cardinality | Description |
|---|---|---|---|---|
| localOnly | boolean | | 0..1 | If **localOnly=true** then distributed operations associated to this Context Source Registration will act only on data held directly by the registered Context Source itself (see clause 4.3.6.4). |
| cacheDuration | String | String representing a duration in ISO 8601 format [17] | 0..1 | Minimal period of time which shall elapse between two consecutive context information consumption operations (as defined in clause 5.7) related to the same context data will occur.<br><br>If the cacheDuration latency period has not been reached, a cached value for the entity or its attributes shall be returned where available. |
| timeout | Number | Greater than 0 | 0..1 | Maximum period of time in milliseconds which may elapse before a forwarded request is assumed to have failed. |
| cooldown | Number | Greater than 0 | 0..1 | Minimum period of time in milliseconds which shall elapse before attempting to make a subsequent forwarded request to the same endpoint after failure.<br><br>If requests are received before the cooldown period has expired, a timeout error response for the registration is automatically returned. |

# Conclusions



FIWARE

# Summary (1)

- All NGSI-LD brokers (e.g Scorpio, Stellio, Orion-LD) are working towards a common NGSI-LD specification
- The NSGI-LD Specification defines many advanced features, that Context Brokers support are obliged to support to comply.
  - These features are essential for certain advanced use cases, but in general usage of advance features is not mandatory and usually will not be required when running simpler installations.
- Use **Multi-Attributes** with `datasetId` for alternative sources, 1-n relationships and sequential command actuations
- Use **Scopes** to logically separate data. `scopeQ` filters hierarchical queries
- Use **Tenants** and the `NGSILD-Tenant` header to isolate context data into separate databases.
- Use **Advanced Registration** features to alter the scope of distributed operations

# Summary (2)

Remember, specification runs before implementation:

- As of **September 2022**, The latest specification revision was **1.6.1** and can be found at: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.06.01_60/gs_CIM009v010601p.pdf.
- Basic operations (e.g. from **1.3**, **1.4**) are commonly available across all brokers
  - Federation has already been achieved between different brokers.
- Expect rollout of recently defined advanced distributed operation **1.6.1** features to arrive in brokers in the coming months based on commercial priorities.

# Summary (3)

- 4 distributed operation modes have been defined :
  - `inclusive` = federation (**default**)
  - `exclusive` = IoT devices
  - `redirect` = distributed databases
  - `auxiliary` = weak secondary sources
- The default list of supported distributed operations is explicitly defined as **Federation** (i.e. Query + Subscribe operations)
  - No changes required for existing federated environments
- For advanced operations and the use of limited partial brokers such as IoT Agents, switch to using registration `mode` and the new `operations` list as soon as available
  - Limit unnecessary and unsupported forwarded requests whenever possible.
- Registration management info (e.g `localOnly`) can be used to add further restrictions to avoid unnecessary traffic, but remember that support for these options is always defined as best effort only



FIWARE

FIWARE Global Summit

Find Us On

Stay up to date

JOIN OUR NEWSLETTER

Be certified and featured

FIWAREMarketplace

## Hosting Partner

City of Vienna

vienna business agency

## Keystone Sponsors

aws

ms.gis

Raseel

Red Hat

## Media Partners

bee smart city

CITIES FORUM

CitiesToday

Our Future Water

RM RENEWABLE MATTER

Smart Cities World

the smartcity journal

FIWARE
Global
Summit

# Thanks!

Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds