TRAINING

FIWARE NGSI:

Managing Context Information at large scale

Fermín Galán Orion Context Broker dev team

FIWARE

Open APIs for Open Minds

FIWARE NGSI: Managing Context Information at large scale

Fermín Galán Márquez

Technological Expert. Orion Context Broker Development Lead

fermin.galanmarquez@telefonica.com

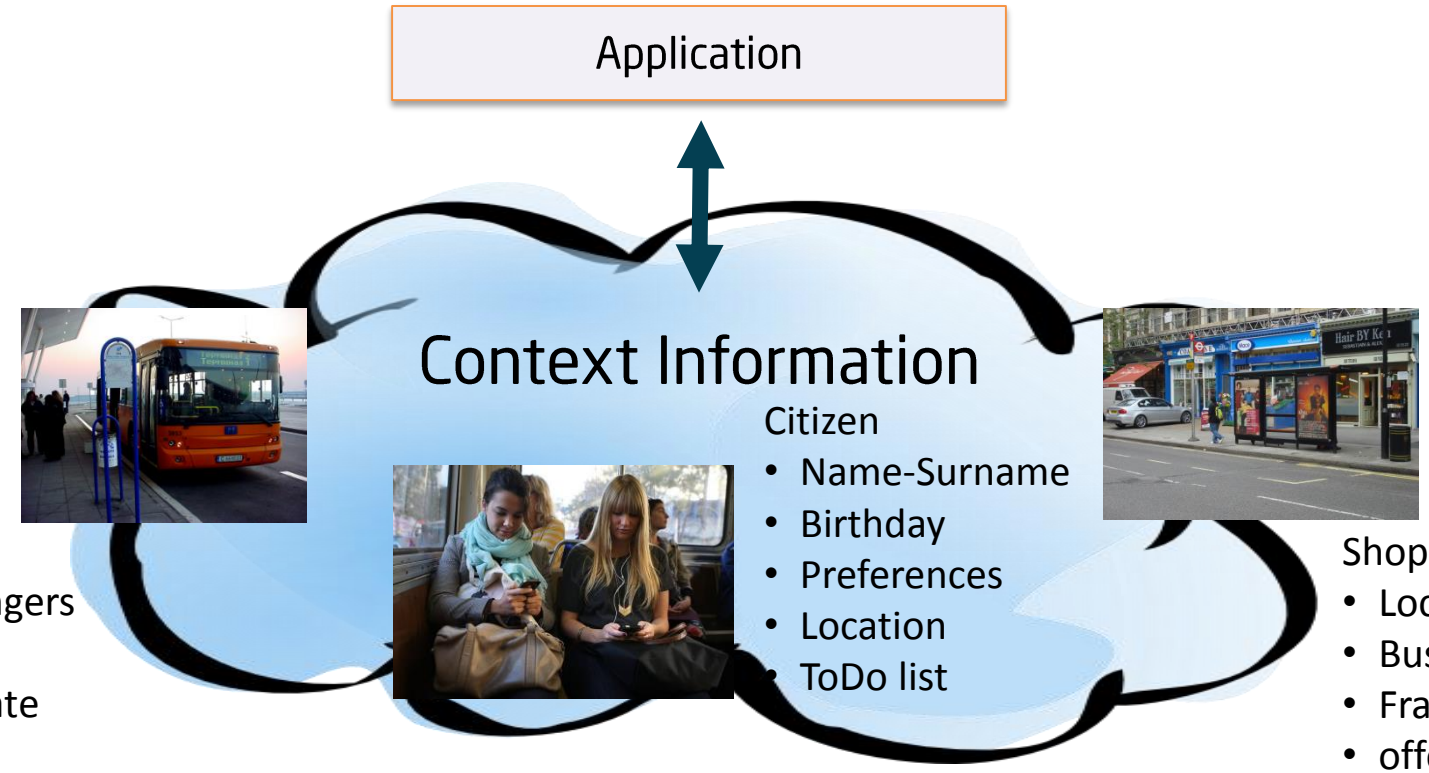
FIWARE

Outline

- Context Management in FIWARE
- Orion Context Broker
- Creating and pulling data
- Pushing data and notifications
- Batch operations
- Advanced functionality

Being "Smart" requires first being "Aware"

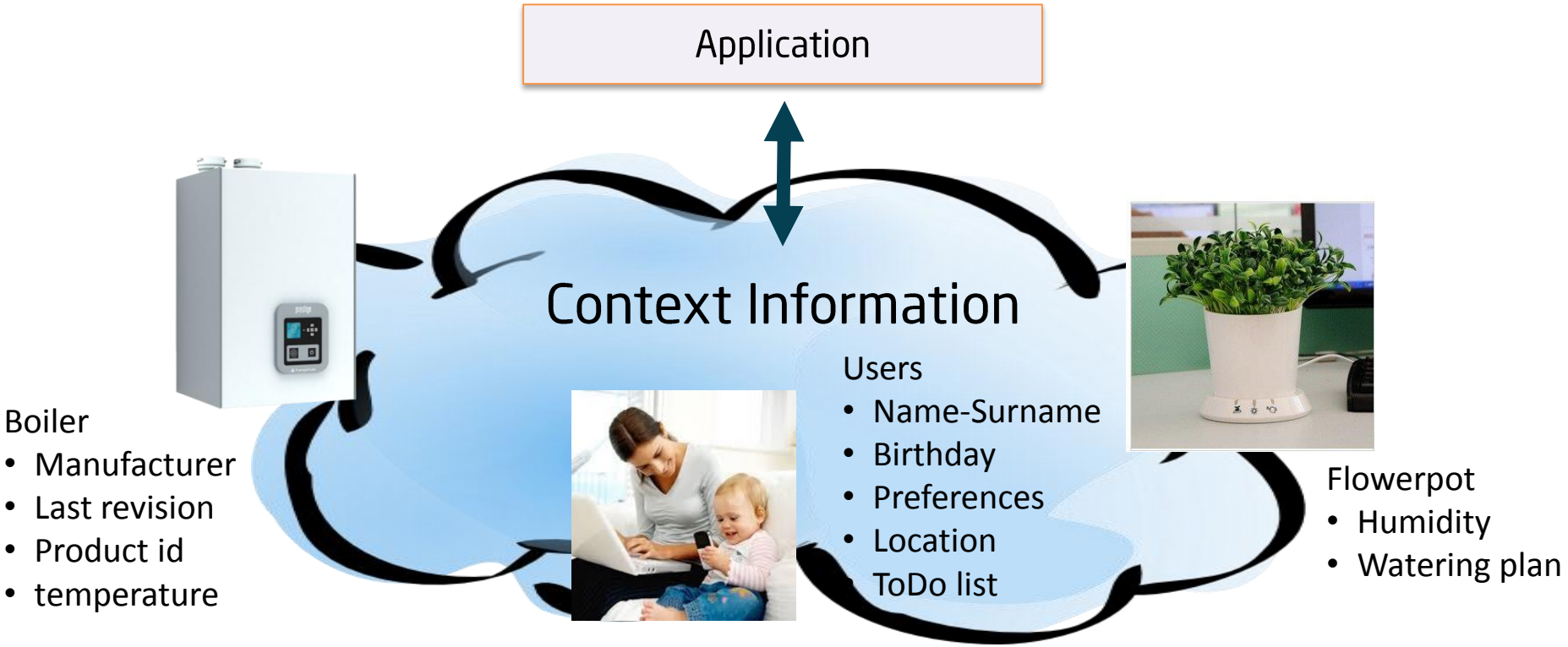
- Implementing a Smart Application requires gathering and managing context information
- Context information refers to the values of attributes characterizing entities relevant to the application



FIWARE

Being "Smart" requires first being "Aware"

- Implementing a Smart Application requires gathering and managing context information
- Context information refers to the values of attributes characterizing entities relevant to the application



FIWARE

Different sources of context need to be handle

- Context information may come from many sources:
  - Existing systems
  - Users, through mobile apps
  - Sensor networks (IoT Devices)
- Source of info for a given entity.attribute may vary over time

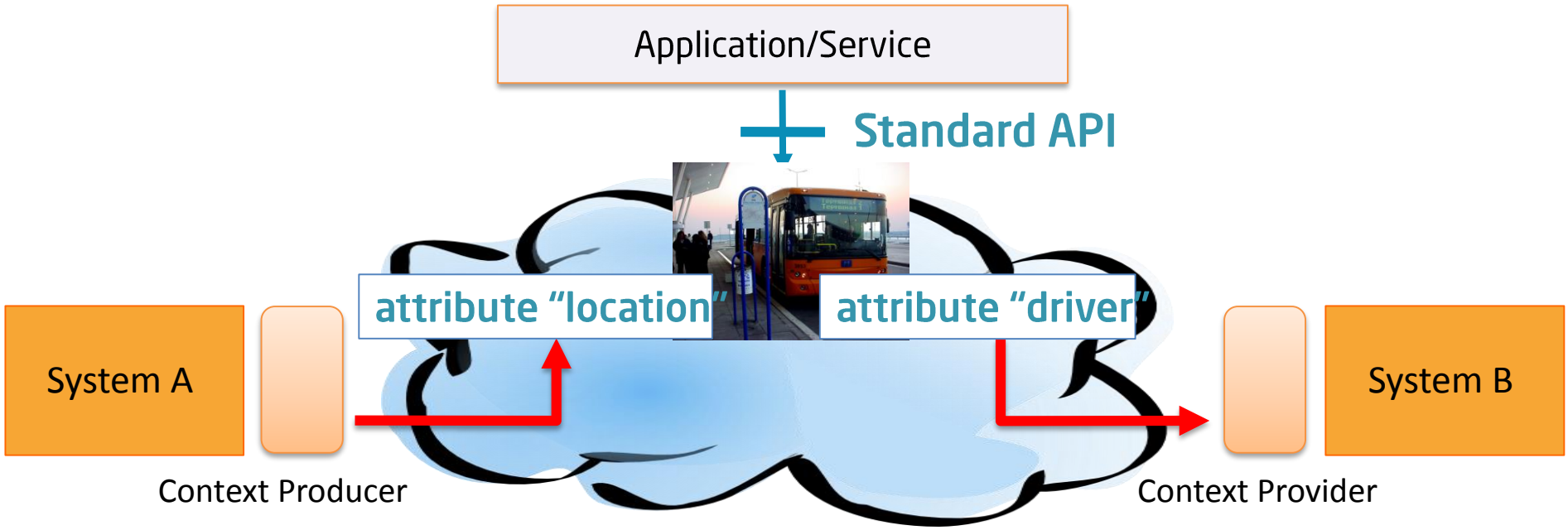
A sensor in a pedestrian street

A person from his smartphone

The Public Bus Transport Management system

A non-intrusive approach is required

- Capable to integrate with existing or future systems dealing with management of municipal services without impact in their architectures
- Info about attributes of one entity may come from different systems, which work either as Context Producers or Context Providers
- Applications rely on a single model adapting to systems of each city

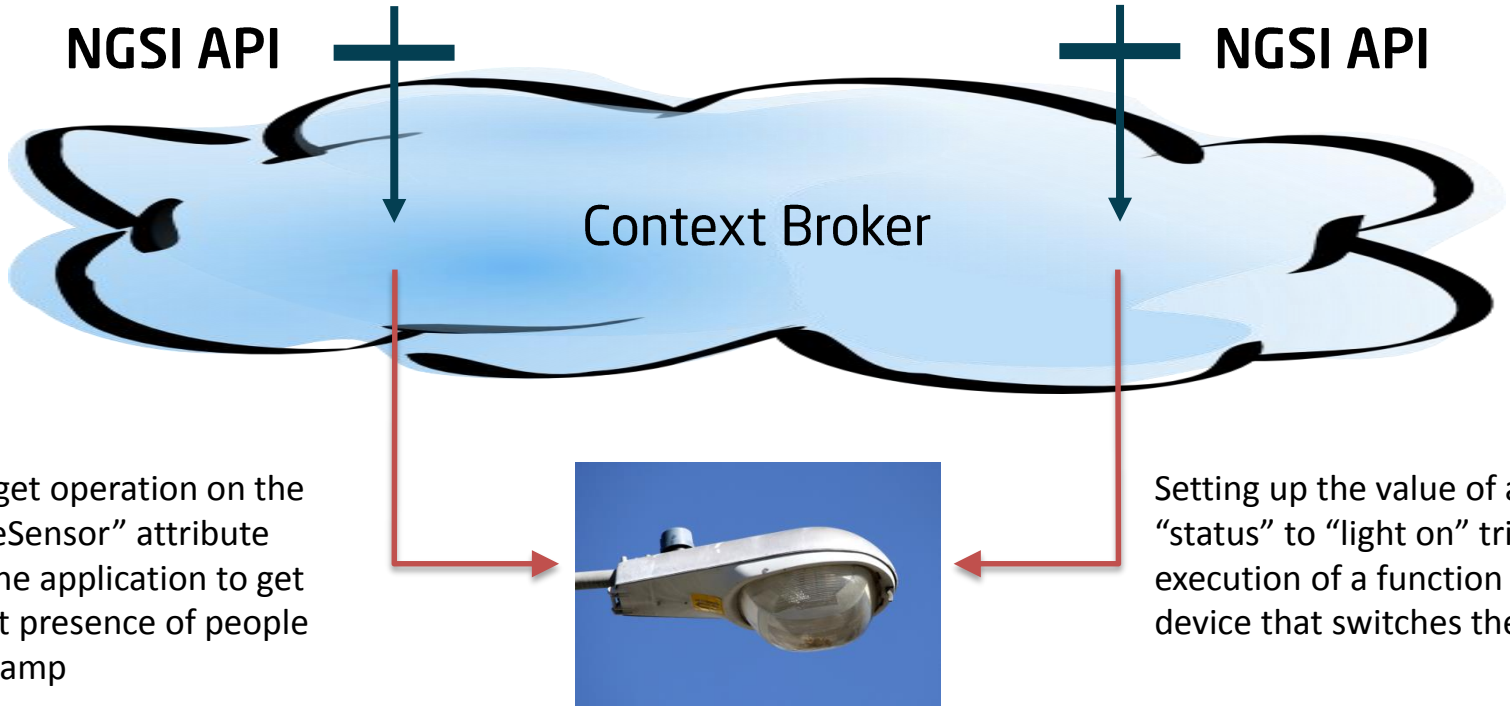


FIWARE NGSI: "The SNMP for IoT"

- Capturing data from, or Acting upon, IoT devices becomes as easy as to read/change the value of attributes linked to context entities using a Context Broker

GET <Oauth token>
/v2/entities/lamp1/attrs/presenceSensor

PUT <Oauth token>
/v2/entities/lamp1/attrs/status/value
"light on"



Issuing a get operation on the "presenceSensor" attribute enables the application to get info about presence of people near the lamp

Setting up the value of attribute "status" to "light on" triggers execution of a function in the IoT device that switches the lamp on

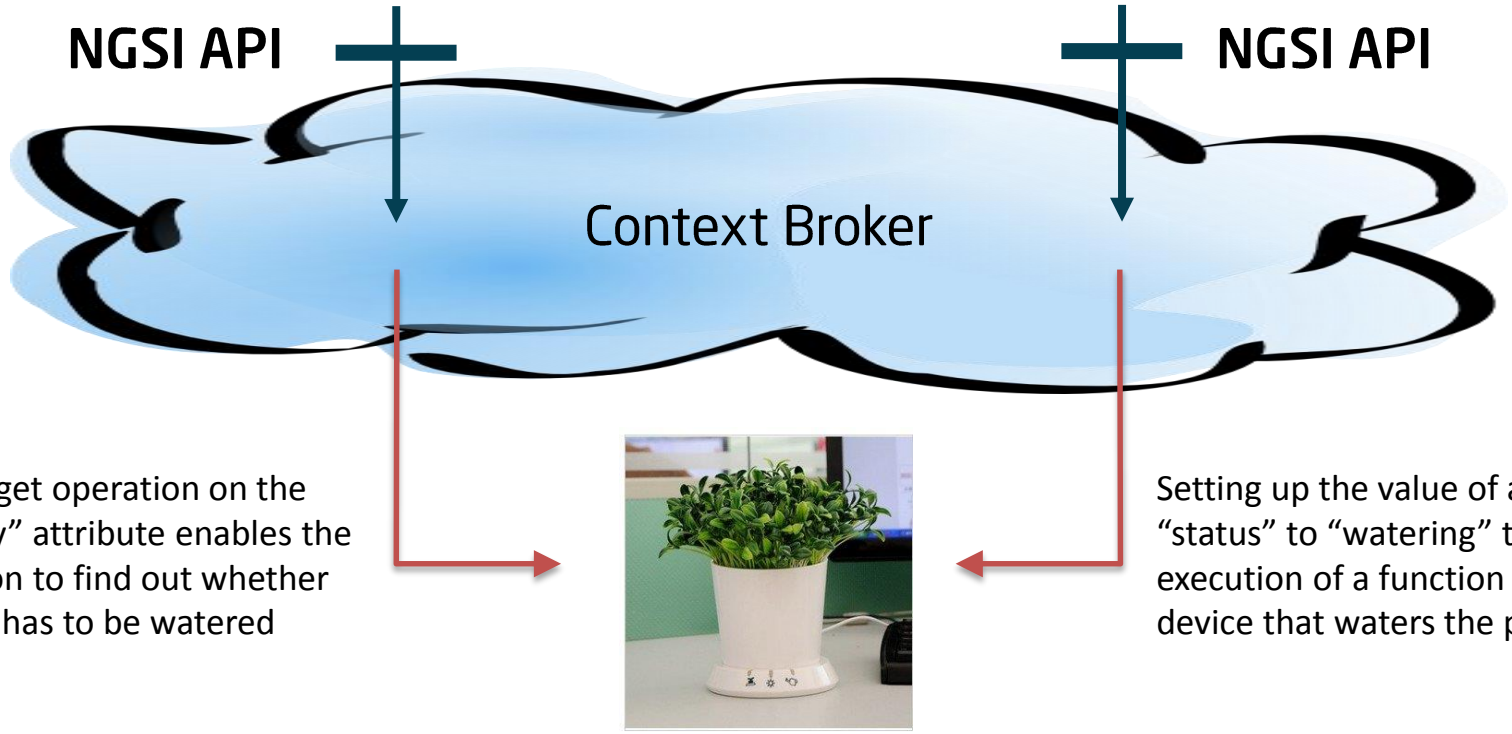
FIWARE

Connecting to the Internet of Things

- Capturing data from, or Acting upon, IoT devices becomes as easy as to read/change the value of attributes linked to context entities using a Context Broker

GET <Oauth token>
/v2/entities/lamp1/attrs/humidity

PUT <Oauth token>
/v2/entities/lamp1/attrs/status/value
"watering"



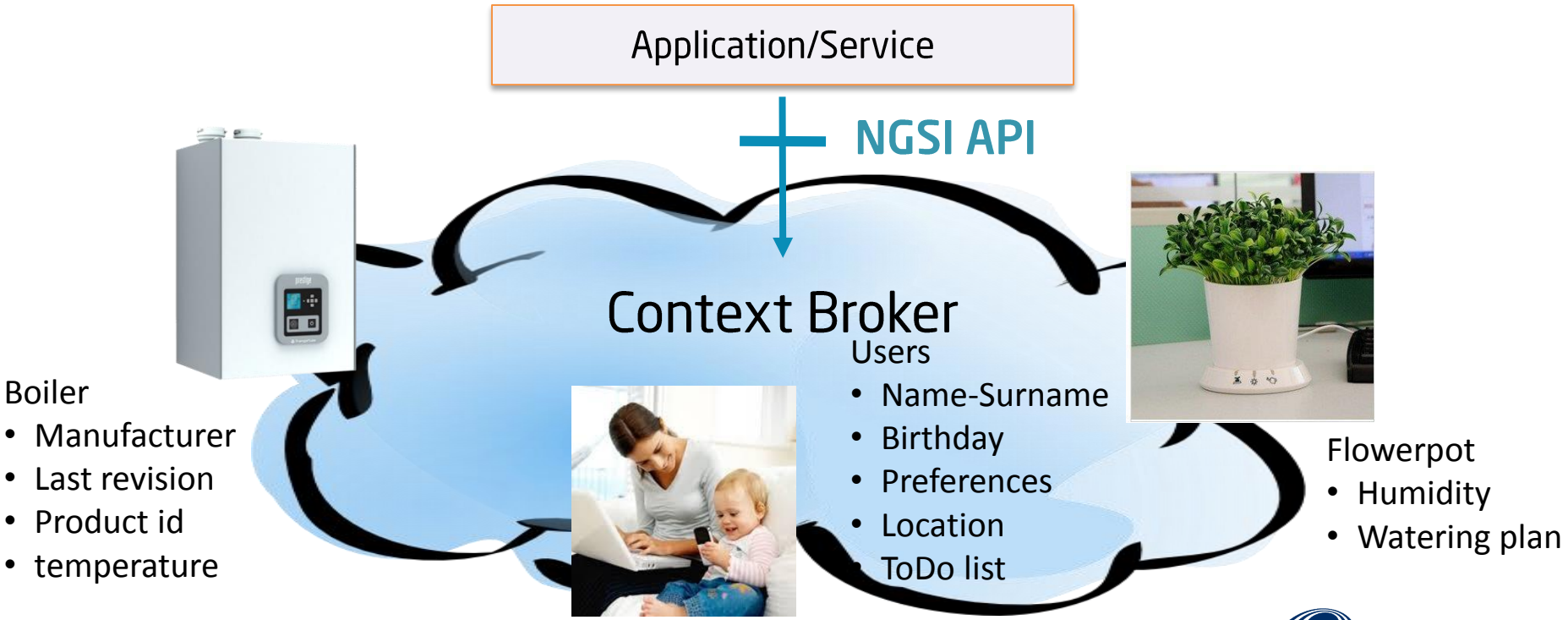
Issuing a get operation on the "humidity" attribute enables the application to find out whether the plant has to be watered

Setting up the value of attribute "status" to "watering" triggers execution of a function in the IoT device that waters the plant

FIWARE

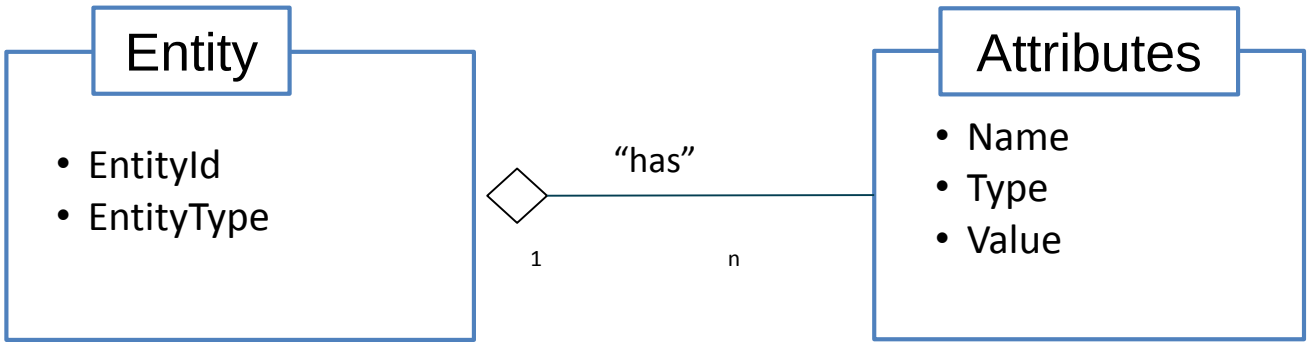
Context Management in FIWARE

- The FIWARE Context Broker GE implements the OMA NGSI-9/10 API: a simple yet powerful standard API for managing Context information complying with the requirements of a smart city
- The FIWARE NGSI API is Restful: any web/backend programmer gets quickly used to it



Orion Context Broker

- Main functions:
  - Context management
  - Context availability management (advanced topic)
- HTTP and REST-based
  - JSON payload support
- Context in NGSI is based in an **entity-attribute** model:



Two “flavors” of NGSI API

- NGSIv1
  - Original NGSI RESTful binding of OMA-NGSI
  - Implemented in 2013
  - Uses the **/v1** prefix in resource URL
- NGSIv2
  - A revamped, simplified binding of OMA-NGSI
    - Simple things must be easy
    - Complex things should be possible
    - Agile, implementation-driven approach
    - Make it as developer-friendly as possible (RESTful, JSON, …)
  - Enhanced functionality compared with NGSIv1 (eg. filtering)
  - Stable, ready for production, version already available
    - Current NGSIv2 version is **Release Candidate 2016.10** http://telefonicaid.github.io/fiware-orion/api/v2/stable
    - New features coming (http://telefonicaid.github.io/fiware-orion/api/v2/stable)
  - Uses the **/v2** prefix in resource URL
- Introduction to NGSIv2
  - https://docs.google.com/presentation/d/1_fv9dB5joCsOCHIb4Ld6A-QmeIYhDzHgFHUWreGmvKU/edit#slide=id.g53c31d7074fd7bc7_0

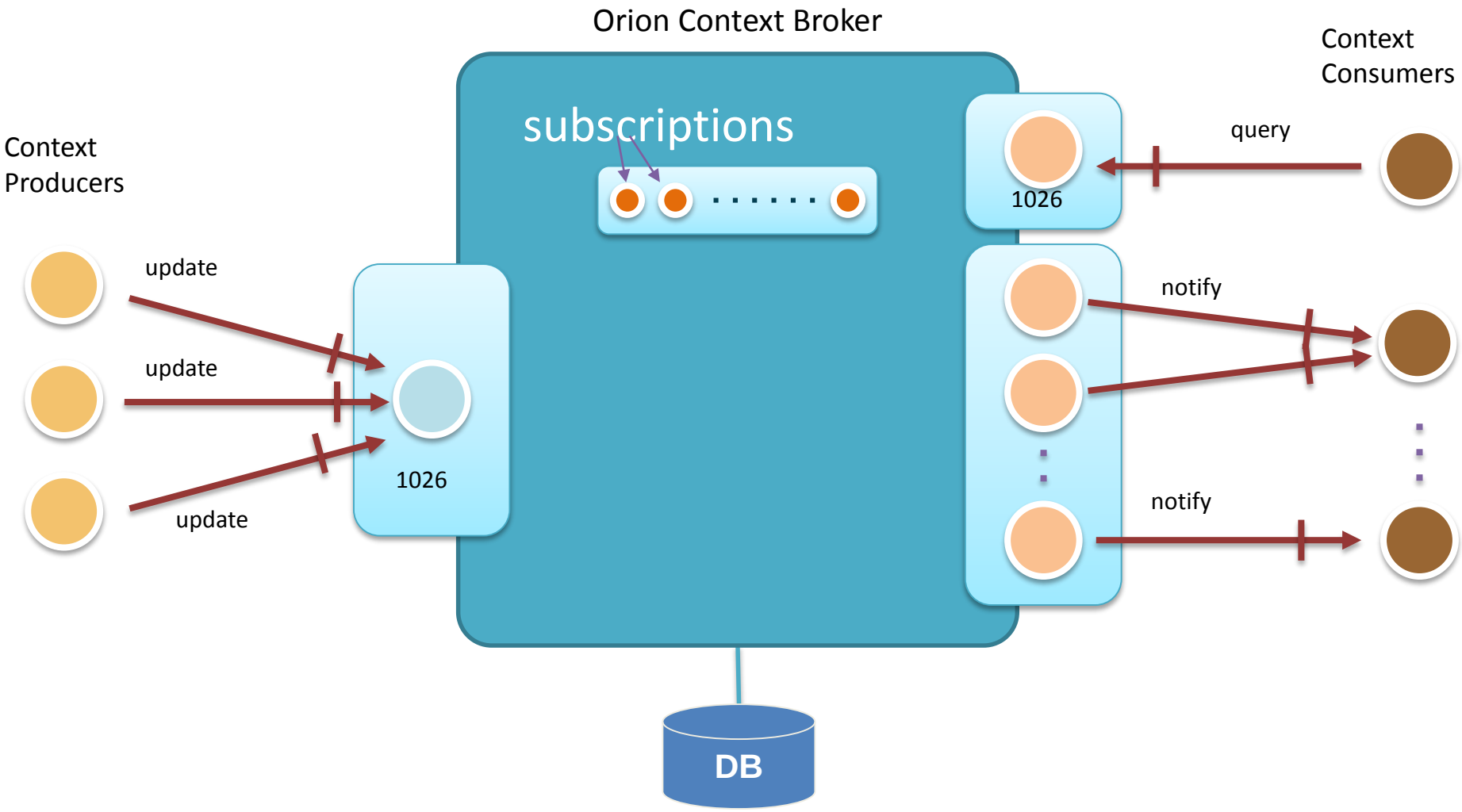
NGSIv2 status (AKA the “NGSIv2 disclaimer”)

- NGSIv2 is in “release candidate” status
  - By "release candidate" we mean that the specification is quite stable, but changes may occur with regard to new release candidates or the final version. In particular changes may be of two types:
    - Extensions to the functionality currently specified by this document. Note that in this case there isn't any risk of breaking backward compatibility on existing software implementations.
    - Slight modifications in the functionality currently specified by this document, as a consequence of ongoing discussions. Backward compatibility will be taken into account in this case, trying to minimize the impact on existing software implementations. In particular, only completely justified changes impacting backward compatibility will be allowed and "matter of taste" changes will not be allowed.

So… when should I use NGSIv1 or NGSIv2?

- In general, **it is always preferable to use NGSIv2**
- However, you would need to use NGSIv1 if
  - You need register/discovery operations (context management availability functionality)
    - Not yet implemented in NGSIv2 (in roadmap)
  - Zero tolerance to changes in software interacting with Orion
- Even if you use NGSIv1, you can still use NGSIv2 advanced functionality
  - See "Considerations on NGSIv1 and NGSIv2 coexistence" section at Orion manual
- For a NGSIv1-based version of this presentation have a look to
  - http://bit.ly/fiware-orion-ngsiv1

Orion Context Broker in a nutshell



Orion Context Broker – check health

GET <cb_host>:1026/version


```
{
  "orion" : {
    "version" : "1.6.0",
    "uptime" : "7 d, 21 h, 33 m, 39 s",
    "git_hash" : "aee96414cc3594bba161afb400f69d101978b39c",
    "compile_time" : "Mon Dec 5 08:38:58 CET 2016",
    "compiled_by" : "fermin",
    "compiled_in" : "centollo"
  }
}
```


FIWARE

Orion Context Broker Basic Operations

Entities

- GET /v2/entities
  - Retrieve all entities
- POST /v2/entities
  - Creates an entity
- GET /v2/entities/{entityID}
  - Retrieves an entity
- [PUT|PATCH|POST] /v2/entities/{entityID}
  - Updates an entity (different “flavors”)
- DELETE /v2/entities/{entityID}
  - Deletes an entity

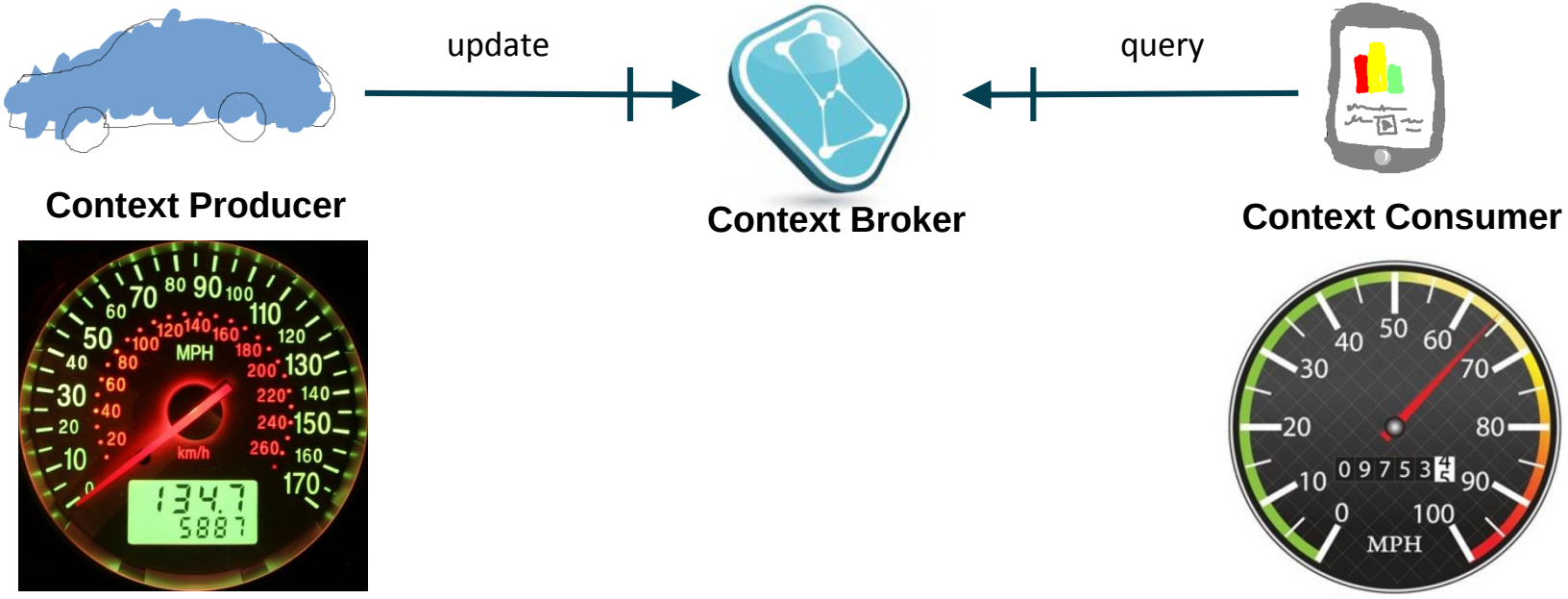
Orion Context Broker Basic Operations

Attributes

- GET /v2/entities/{entityID}/attrs/{attrName}
  - Retrieves an attribute's data
- PUT /v2/entities/{entityID}/attrs/{attrName}
  - Updates an attribute's data
- DELETE /v2/entities/{entityID}/attrs/{attrName}
  - Deletes an attribute
- GET /v2/entities/{entityID}/attrs/{attrName}/value
  - Retrieves an attribute's value
- PUT /v2/entities/{entityID}/attrs/{attrName}/value
  - Updates an attribute's value

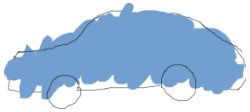
Context Broker operations: create & pull data

- **Context Producers** publish data/context elements by invoking the **update** operations on a Context Broker.
- **Context Consumers** can retrieve data/context elements by invoking the **query** operations on a Context Broker



Quick Usage Example: Car Create

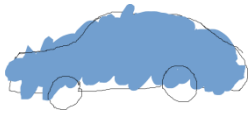
```
POST <cb_host>:1026/v2/entities
Content-Type: application/json
...

{
  "id": "Car1",
  "type": "Car",
  "speed":  {
    "type": "Float",
    "value": 98
  }
}
```



201 Created

FIWARE

Quick Usage Example: Car Speed Update (1)



```
PUT <cb_host>:1026/v2/entities/Car1/attrs/speed
Content-Type: application/json
...

{
   "type": "Float",
   "value": 110
}
```

In the case of id ambiguity, you can use "?type=Car" to specify entity type

```
204 No Content
...
```

FIWARE

Quick Usage Example: Car Speed Query (1)

GET <cb_host>:1026/v2/entities/Car1/attrs/**speed**

200 OK
Content-Type: application/json
...

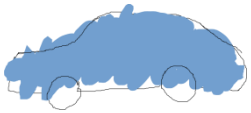
```
{
   "type": "Float",
   "value": 110,
   "metadata": {}
}
```

You can get all the attributes of the entity using the entity URL:
GET/v2/entities/Car1/attrs

FIWARE

Quick Usage Example: Car Speed Update (2)

```
PUT <cb_host>:1026/v2/entities/Car1/attrs/speed/value
Content-Type: text/plain
...

115
```




```
204 No Content
...
```

FIWARE

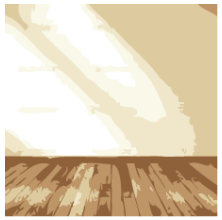
Quick Usage Example: Car Speed Query (2)

```
GET <cb_host>:1026/v2/entities/Car1/attrs/speed/value
Accept: text/plain
```

```
200 OK
Content-Type: text/plain
...

115.000000
```

Quick Usage Example: Room Create (1)

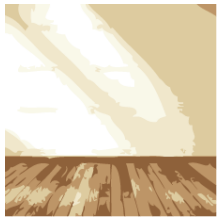
```
POST <cb_host>:1026/v2/entities
Content-Type: application/json
...

{
  "id": "Room1",
  "type": "Room",
  "temperature": {
    "type": "Float",
    "value": 24
  },
  "pressure": {
    "type": "Integer",
    "value": 718
  }
}
```



```
201 Created
...
```

Quick Usage Example: Room Update (1)

```
PATCH <cb_host>:1026/v2/entities/Room1/attrs
Content-Type: application/json
...

{
  "temperature“: {
    "type": "Float",
    "value": 25
  },
  "pressure": {
    "type": "Integer",
    "value":  720
  }
}
```



```
204 No Content
...
```

FIWARE

Quick Usage Example: Room Query (1)

GET <cb_host>:1026/v2/entities/**Room1**/attrs

```
200 OK
Content-Type: application/json
...

{
   "pressure": {
      "type": "Integer",
      "value": 720,
      "metadata": {}
   },
   "temperature": {
      "type": "Float",
      "value": 25,
      "metadata": {}
   }
}
```

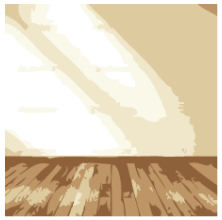
Quick Usage Example: Room Query (2)

GET <cb_host>:1026/v2/entities/Room1/attrs?**options=keyValues**

```
200 OK
Content-Type: application/json
...

{
   "pressure": 720,
   "temperature": 25
}
```

Quick Usage Example: Room Create (2)

```
POST <cb_host>:1026/v2/entities
Content-Type: application/json
...

{
  "id": "Room2",
  "type": "Room",
  "temperature": {
    "type": "Float",
    "value": 29
  },
  "pressure": {
    "type": "Integer",
    "value": 730
  }
}
```



```
201 Created
...
```

Quick Usage Example: Filters (1)

```
GET <cb_host>:1026/v2/entities?options=keyValues&q=temperature>27
```

```
200 OK
Content-Type: application/json
...

[
   {
      "id": "Room2",
      "pressure": 730,
      "temperature": 29,
      "type": "Room"
   }
]
```

Quick Usage Example: Filters (2)

```
GET <cb_host>:1026/v2/entities?options=keyValues&q=pressure==715..725
```

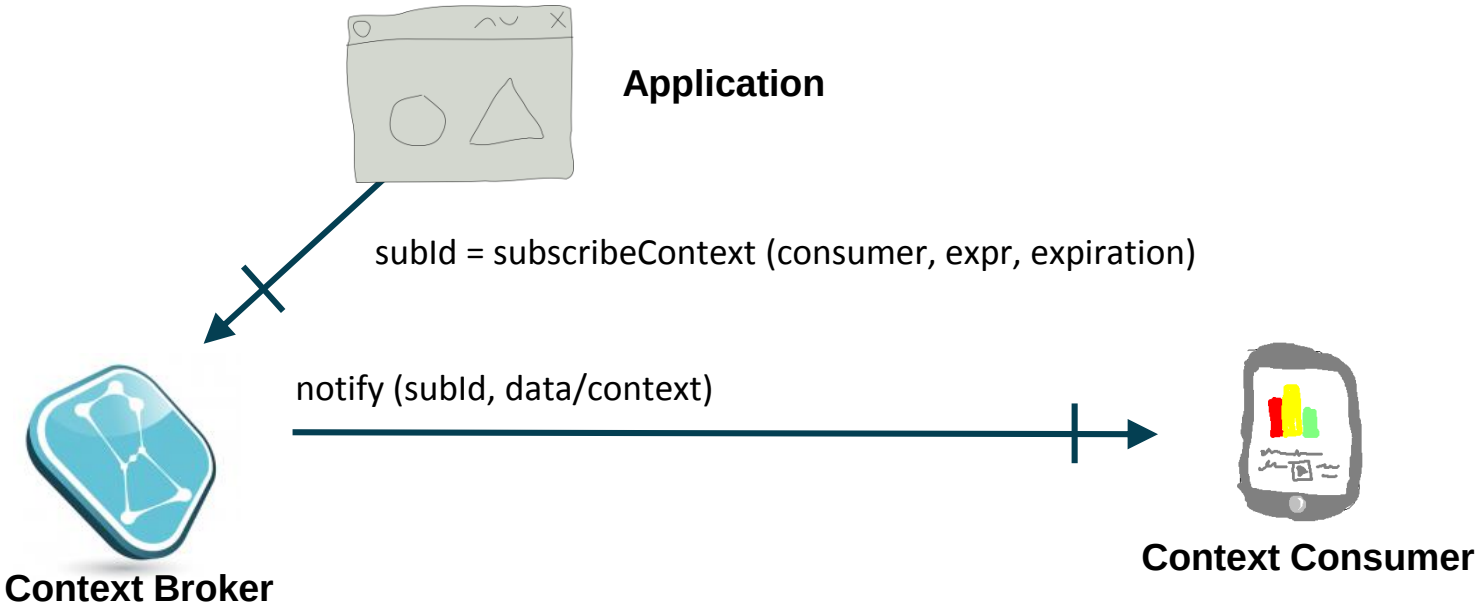
The full description of the Simple Query Language for filtering can be found in the NGSIv2 Specification document

```
200 OK
Content-Type: application/json
...

[
   {
      "id": "Room1",
      "pressure": 720,
      "temperature": 25,
      "type": "Room"
   }
]
```

Context Broker operations: push data

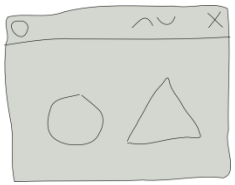
- **Context Consumers** can subscribe to receive context information that satisfy certain conditions using the **subscribe** operation. Such subscriptions may have an expiration time.

- The Context Broker notifies updates on context information to subscribed Context Consumers by invoking the **notify** operation they export



Quick Usage Example: Subscription

```
POST <cb_host>:1026/v2/subscriptions
Content-Type: application/json
...

{
  "subject": {
    "entities": [
      {
        "id": "Room1",
        "type": "Room"
      }
    ],
    "condition": {
      "attrs": [ "temperature"  ]
      }
    },
   "notification": {
    "http": {
      "url": "http://<host>:<port>/publish"
    },
    "attrs": [  "temperature"   ]
  },
  "expires": "2026-04-05T14:00:00.00Z"
}
```



```
201 Created
Location: /v2/subscriptions/51c0ac9ed714fb3b37d7d5a8
...
```

Quick Usage Example: Notification

25

19

FIWARE

Quick Usage Example: Notification

```
POST /publish HTTP/1.1
Content-type: application/json; charset=utf-8
Ngsiv2-AttrsFormat: normalized
…

{
   "subscriptionId": "574d720dbef222abb860534a",
   "data": [
      {
         "id": "Room1",
         "type": "Room",
         "temperature": {
            "type": "Float",
            "value": 19,
            "metadata": {}
         }
      }
   ]
}
```

List existing subscriptions

GET <cb_host>:1026/v2/subscriptions

The full description of the subscription object (including all its fields) can be found in the NGSIv2 Specification

```
200 OK
Content-Type: application/json
...
[{
   "id": " 51c0ac9ed714fb3b37d7d5a8 ",
   "expires": "2026-04-05T14:00:00.00Z",
   "status": "active",
   "subject": {
      "entities": [{
         "id": "Room1",
         "type": "Room"
      }],
      "condition": {
         "attrs": ["temperature"]
      }
   },
   "notification": {
      "timesSent": 3,
      "lastNotification": "2016-05-31T11:19:32.00Z",
      "lastSuccess": "2016-05-31T11:19:32.00Z",
      "attrs": ["temperature"],
      "attrsFormat": "normalized",
      "http": {
         "url": "http://localhost:1028/publish"
      }
   }
}]
```

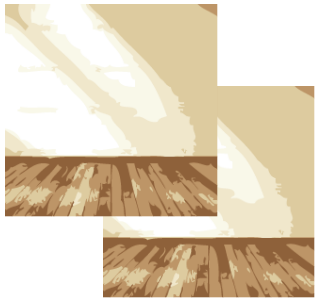
Orion Context Broker batch operations

- Batch query and batch update
- They are equivalent in functionality to previously described RESTful operations
- All them use **POST** as verb and the ***/v2/op*** URL prefix, including operation parameters in the JSON payload
- They implement extra functionality that cannot be achieved with RESTful operations, e.g. to create several entities with the same operation
- They are not a substitute but a complement to RESTful operations

Batch Operation Example: Create Several Rooms

```
POST <cb_host>:1026/v2/op/update
Conten-Type: application/json
...

{
  "actionType": "APPEND",
  "entities": [
    {
      "type": "Room",
      "id": "Room3",
      "temperature": {
        "value": 21.2,
        "type": "Float"
      },
      "pressure": {
        "value": 722,
        "type": "Integer"
      }
    },
...
```

```
...
    {
      "type": "Room",
      "id": "Room4",
      "temperature": {
        "value": 31.8,
        "type": "Float"
      },
      "pressure": {
        "value": 712,
        "type": "Integer"
      }
    }
  ]
}
```



```
201 Created
...
```

How to get Orion? (Virtual Machines)

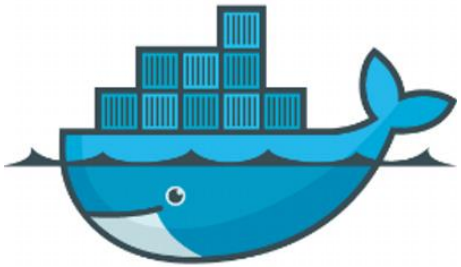
- FIWARE Lab image
  - Image: **orion-psb-image-R<x>.<y>**
- VirtualBox image
  - http://bit.ly/fiware-orion024-vbox (it's big!)
  - User/pass:
    - fiware/fiware
    - root/fiware
- **Hint**: update Orion package once the VM is deployed

How to get Orion? (Docker containers)

- Assuming docker is installed in your system
- Documentation in https://github.com/telefonicaid/fiware-orion/tree/develop/docker
- Quick guide

```
git clone https://github.com/telefonicaid/fiware-orion.git
cd fiware-orion/docker
sudo docker-compose up
```

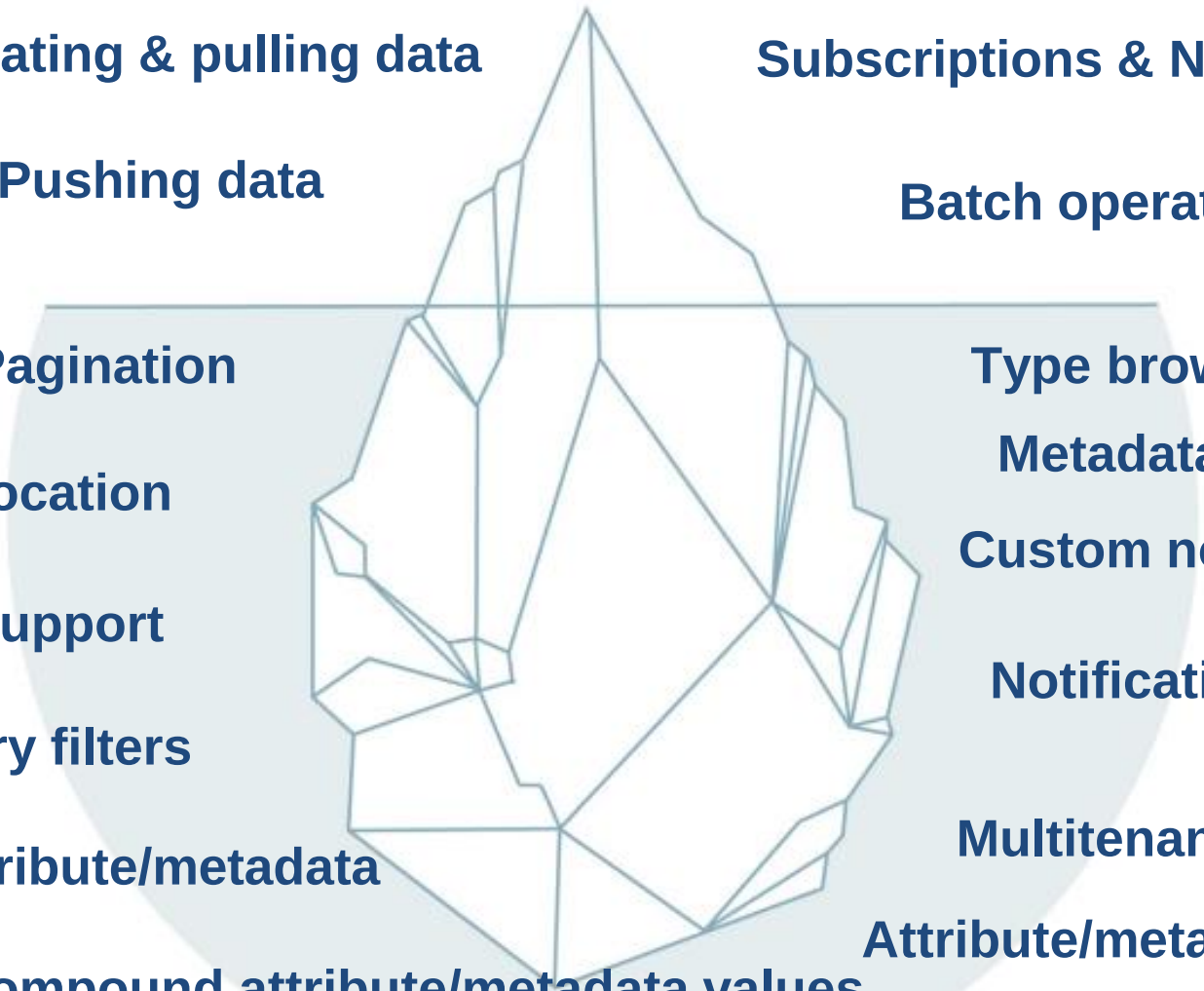
- That's all!
  - curl localhost:1026/version



Would you like to play with this?

- Have a look to the FIWARE Reference Tutorial application
  - git clone https://github.com/Fiware/tutorials.TourGuide-App.git
  - cd tutorials.TourGuide-App/
  - docker-compose up orion
  - curl localhost:1026/version
- Self-explanatory README.md at root directory
- Open a Postman session and rock and roll
  - Postman collection: https://github.com/Fiware/tutorials.TourGuide-App/blob/develop/contrib/CampusParty2016.postman_collection

Orion advanced functionality



Creating & pulling data

Pushing data

Pagination

Geo-location

DateTime support

Query filters

Special attribute/metadata

Compound attribute/metadata values

Registrations & context providers

Subscriptions & Notifications

Batch operations

Type browsing

Metadata

Custom notifications

Notification status

Multitenancy

Attribute/metadata filtering

FIWARE

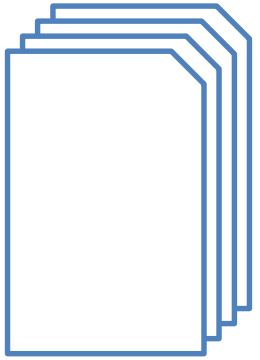
Geo-location

- Entities may have a location
- Queries/subscriptions may use the location as search criteria
- Specific session on this **tomorrow 12:45-13:30**
  - NGSI: geoqueries and Carto integration, Fermín Galán & Francisco Romero (Data Team)

FIWARE

Pagination

- Pagination helps clients organize query and discovery requests with a large number of responses.
- Three URI parameters:
  - limit
    - Number of elements per page (default: 20, max: 1000)
  - offset
    - Number of elements to skip (default: 0)
  - count (option)
    - Returns total elements (default: not return)



Pagination

- Example, querying the first 100 entries:
  - GET <orion_host>:1026/v2/entities?limit=100&options=count
- The first 100 elements are returned, along with the following header in the response:
  - Fiware-Total-Count: 322
- Now we now there are 322 entities, we can keep querying the broker for them:
  - GET <orion_host>:1026/v2/entities?offset=100&limit=100
  - GET <orion_host>:1026/v2/entities?offset=200&limit=100
  - GET <orion_host>:1026/v2/entities?offset=300&limit=100

Pagination

- By default, results are ordered by entity creation date
- This behavior can be overridden using **orderBy** URI parameter
  - A comma-separated list of attributes. Results are ordered by the first attribute. On ties, the results are ordered by the second attribute and so on. A "!" before the attribute name means that the order is reversed.
- Example: get the first 10 entities ordered by temp in ascending order, then humidity in descending order

  GET <orion_host>:1026/v2/entities?limit=20&offset=0&orderBy=temp,!humidity
- **dateCreated** and **dateModified** can be used to ordering by entity creation and modification date, respectively

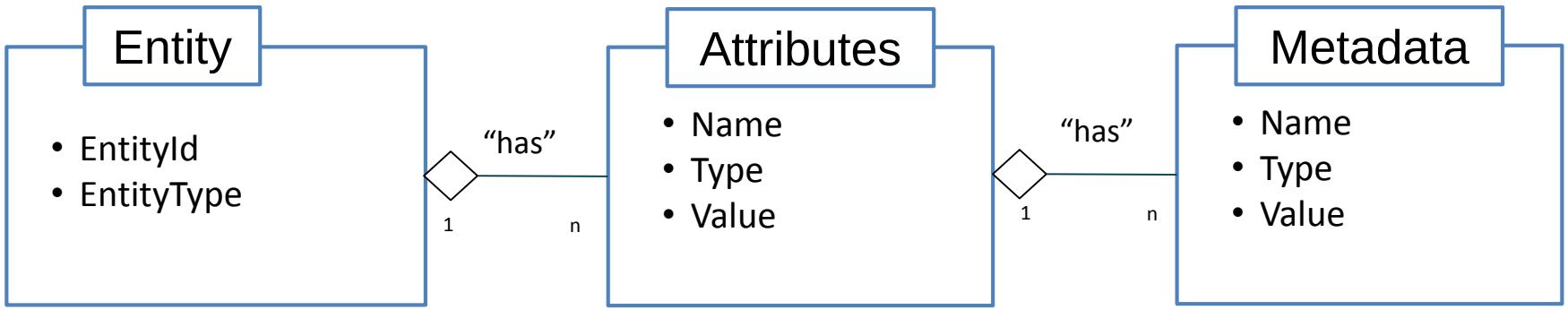
Metadata

- Users may attach metadata to attributes
- Reserved metadata: ID, location, dateCreated, dateModified, previousValue, actionType
- Examples:

```
…
"temperature": {
    "type": "Float",
    "value": 26.5,
    "metadata": {
    {
     "accuracy": {
      "type": "Float",
      "value": 0.9
     }
    }
}
…
```

```
…
"temperature": {
    "type": "Float",
    "value": 26.5,
    "metadata": {
    {
     “average": {
      "type": "Float",
      "value": 22.4
     }
    }
}
…
```

Complete NGSI Model



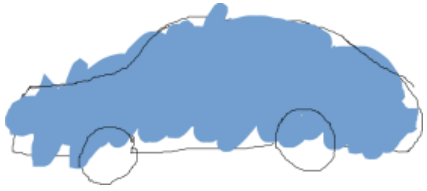
Compound Attribute/Metadata Values

- Attributes and metadata can have a structured value. Vectors and key-value maps are supported.
- It maps directly to JSON's objects and arrays.

Compound Attribute/Metadata Values

- Example: we have a car whose four wheels' pressure we want to represent as a compound attribute for a car entity. We would create the car entity like this:

```
{
   "type": "Car",
   "id": "Car1",
   "tirePressure": {
      "type": "kPa",
      "value": {
         "frontRight": "120",
         "frontLeft": "110",
         "backRight": "115",
         "backLeft": "130"
      }
   }
}
```



Type Browsing

- GET /v2/types
  - Retrieve a list of all entity types currently in Orion, including their corresponding attributes and entities count
- GET /v2/types/{typeID}
  - Retrieve attributes and entities count associated to an entity type

PRO TIP

GET /v2/contextTypes?options=values

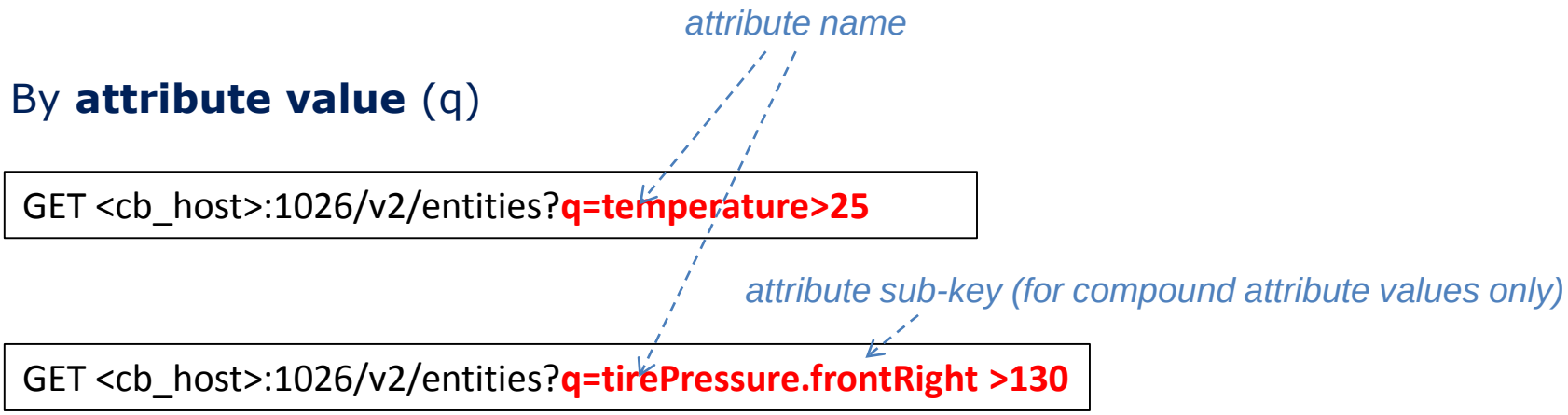
Retrieves just a list of all entity types without any extra info

Query filters

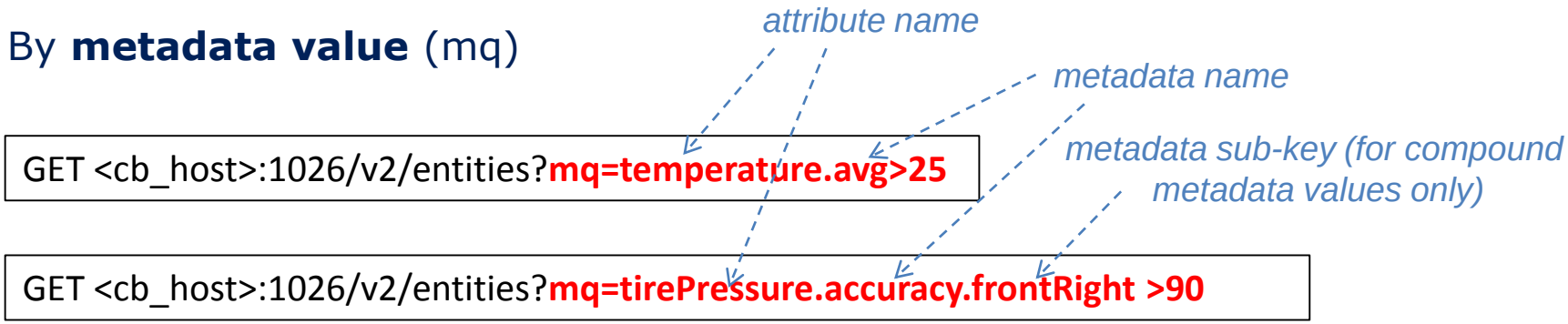
- For the **GET /v2/entities** operation
- By **entity type** GET <cb_host>:1026/v2/entities?**type=Room**
- By **entity id list** GET <cb_host>:1026/v2/entities?**id=Room1,Room2**
- By **entity id pattern** (regex) GET <cb_host>:1026/v2/entities?**idPattern=^Room[2-5]**
- By **entity type pattern** (regex) GET <cb_host>:1026/v2/entities?**typePattern=T[ABC]**
- By **geographical location**
  - Described in detail in previous slides
- Filters can be used simultaneously (i.e. like **AND** condition)

Query filters

- By **attribute value** (q)



- By **metadata value** (mq)



- See full details about **q** and **mq** query language in NGSIv2 specification

FIWARE

Query filters

- Filters can be also used in subscriptions
  - id
  - type
  - id pattern
  - type pattern
  - attribute values
  - metadata value
  - geographical location

```
POST <cb_host>:1026/v2/subscriptions
...
{
  "subject": {
    "entities": [
      {
        "id": “Car5",
        "type": “Car"
      },
      {
        "idPattern": “^Room[2-5]",
        "type": "Room"
      },
      {
        "id": “D37",
        "typePattern": "Type[ABC]"
      },
    ],
    "condition": {
      "attrs": [ "temperature"  ],
      "expression":  {
        "q": "temperature>40",
        "mq": "humidity.avg==80..90",
        "georel": "near;maxDistance:100000",
        "geometry": "point",
        "coords": "40.418889,-3.691944"
      }
      }
    },
   ...
}
```

Datetime support

- Orion implements date support
  - Based on ISO ISO8601 format, including partial representations and timezones
    - See https://fiware-orion.readthedocs.io/en/master/user/ngsiv2_implementation_notes/index.html#datetime-support for syntax details
  - Use reserved attribute type **DateTime** to express a date
  - Date-based filters are supported

Datetime support

```
POST /v2/entities
...
{
  "id": "John",
  "birthDate": {
    "type": "DateTime",
    "value": "1979-10-14T07:21:24.238Z"
  }
}
```

Example: create entity John, with *birthDate* attribute using type DateTime

- Attribute value arithmetic filters can be used with dates as if they were numbers

```
GET /v2/entities?q=birthDate<1985-01-01T00:00:00
```

- Entity **dateModified** and **dateCreated** special attributes, to get entity creation and last modification timestamps
  - They are shown in query responses using **attrs=dateModified,dateCreated**
- Entity **dateModified** and **dateCreated** special metadata, to get attribute creation and last modification timestamps
  - They are shown in query responses using **metadata=dateModified,dateCreated**

Custom notifications

- Apart from the standard formats defined in the previous slides NGSIv2 allows to re-define **all** the notification aspects
- **httpInfo** is used instead of **http**, with the following subfields
  - URL query parameters
  - HTTP method
  - HTTP headers
  - Payload (not necessarily JSON!)
- A simple macro substitution language based on **${..}** syntax can be used to "fill the gaps" with entity data (id, type or attribute values)
  - Exception: this cannot be used in HTTP method field

Custom notifications

```
PUT /v2/entities/DC_S1-D41/attrs/temp/value?type=Room
...
23.4
```

update

notificaiton

```
PUT http://foo.com/entity/DC_S1-D41?type=Room
Content-Type: text/plain
Content-Length: 31

The temperature is 23.4 degrees
```

```
...
"httpCustom": {
 "url": "http://foo.com/entity/${id}",
 "headers": {
  "Content-Type": "text/plain"
 },
 "method": "PUT",
 "qs": {
  "type": "${type}"
 },
 "payload": "The temperature is ${temp} degrees"
 }
...
```

Custom notification configuration

Notification status

- Status **failed** means that last attempt to notify failed
  - E.g. the endpoint is not reachable
- Detailed information in the **notifications** element
  - **timesSent**: total number of notifications attempts (both successful and failed)
  - **lastSuccess**: last time that notification was successfully sent
  - **lastFailure**: last time that notification was tried and failed
  - **lastNotification**: last time the notification was sent (either success or failure)
    - Corollary: lastNotification value is the same than either lastFailure or lastSuccess

```
200 OK
Content-Type: application/json
...
[{
   "id": " 51c0ac9ed714fb3b37d7d5a8 ",
   "expires": "2026-04-05T14:00:00.00Z",
   "status": "failed",
   "subject": { ... },
   "notification": {
      "timesSent": 3,
      "lastNotification": "2016-05-31T11:19:32.00Z",
      "lastSuccess": "2016-05-31T10:07:32.00Z",
      "lastFailure": "2016-05-31T11:19:32.00Z",
      ...
   }
}]
```

Attribute filtering and special attributes

- By default all attribute are included in query responses or notifications
- The **attrs** field (as parameter in GET operations and as **notification** sub-field in subscriptions) can be used to specify a filtering list
- The **attrs** field can be also used to explicitly include some special attributes (not included by default)
  - **dateCreated**, **dateModified**: described in previous slide
- The "*" can be used as an alias of "all the (regular) attributes"

Attribute filtering and special attributes

- Examples
  - Include only attributes temp and lum
    - In queries: GET /v2/entities?attrs=temp,lum
    - In subscriptions: "attrs": ["temp", "lum"]
  - Include dateCreated and not any other attribute
    - In queries: GET /v2/entities?attrs=dateCreated
    - In subscriptions: "attrs": ["dateCreated"]
  - Include dateModified and all the other (regular) attributes
    - In queries: GET /v2/entities?attrs=dateModified,*
    - In subscriptions: "attrs": ["dateModified", "*"]
  - Include all attributes (same effect that not using attrs, not very interesting)
    - In queries: GET /v2/entities?attrs=*
    - In subscriptions: "attrs": ["*"]

Metadata filtering and special attributes

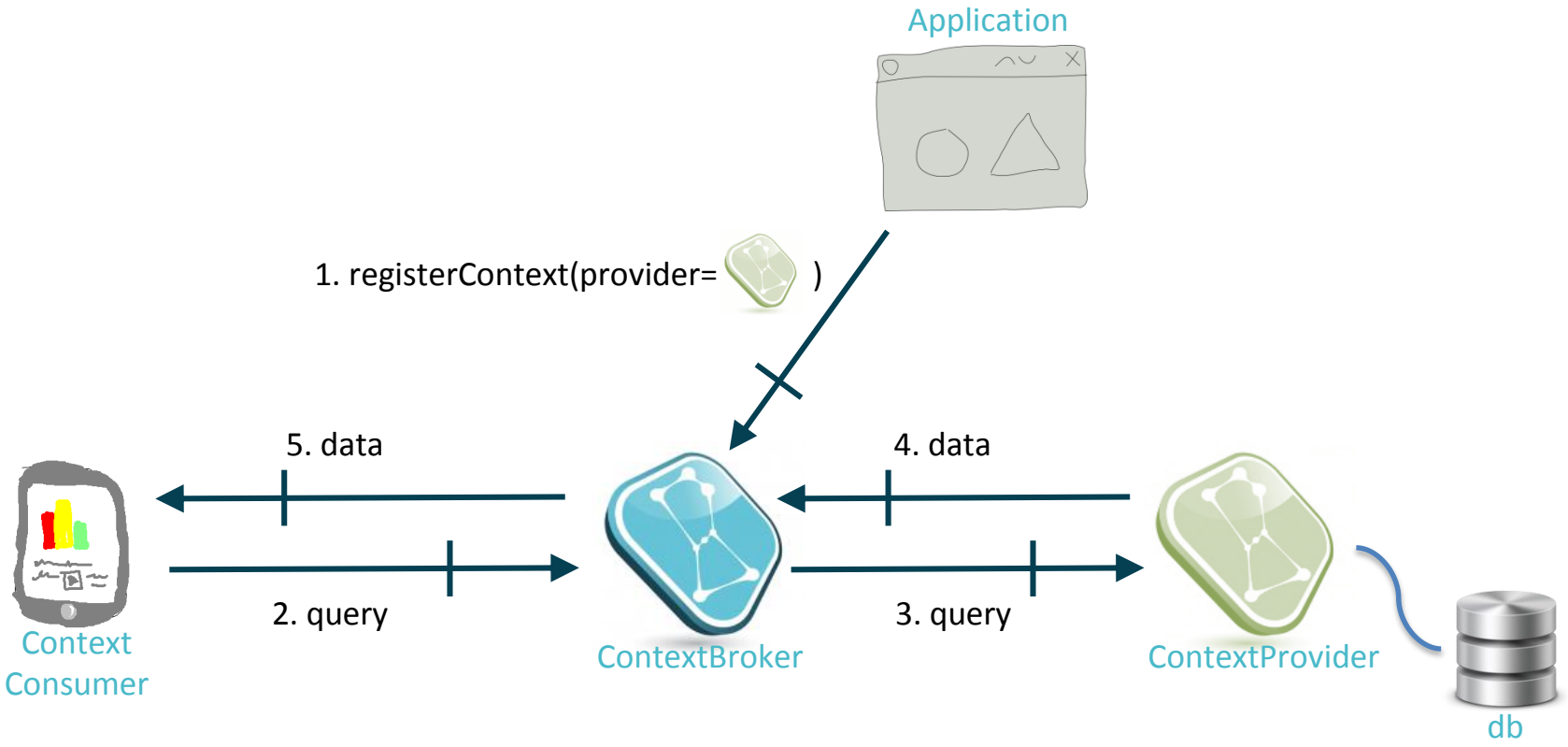
- By default all attribute metadata are included in query responses and notifications
- The **metadata** field (as parameter in GET operations and as **notification** sub-field in subscriptions) can be used to specify a filtering list
- The **metadata** field can be also used to explicitly include some special metadata (not included by default)
  - **dateCreated, dateModified:** described in previous slide
  - **actionType**: which value is the action type corresponding to the update triggering the notification: "update", "append" or "delete"
  - **previousValue**: which provides the value of the attribute previous to processing the update that triggers the notification
- The "*" can be used as an alias of "all the (regular) metadata"

Metadata filtering and special attributes

- Examples
  - Include only metadata MD1 and MD2
    - In queries: GET /v2/entities?metadata=MD1,MD2
    - In subscriptions: "metadata": ["MD1", "MD2"]
  - Include previousValue and not any other metadata
    - In queries: GET /v2/entities?metadata=previousValue
    - In subscriptions: "attrs": ["previousValue"]
  - Include actionType and all the other (regular) metadata
    - In queries: GET /v2/entities?metadata=actionType,*
    - In subscriptions: "attrs": ["actionType", "*"]
  - Include all metadatata (same effect that not using metadata, not very interesting)
    - In queries: GET /v2/entities?metadata=*
    - In subscriptions: "metadata": ["*"]

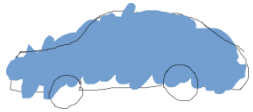
Registration & Context Providers

- Uncached queries and updates



Registration & Context Providers

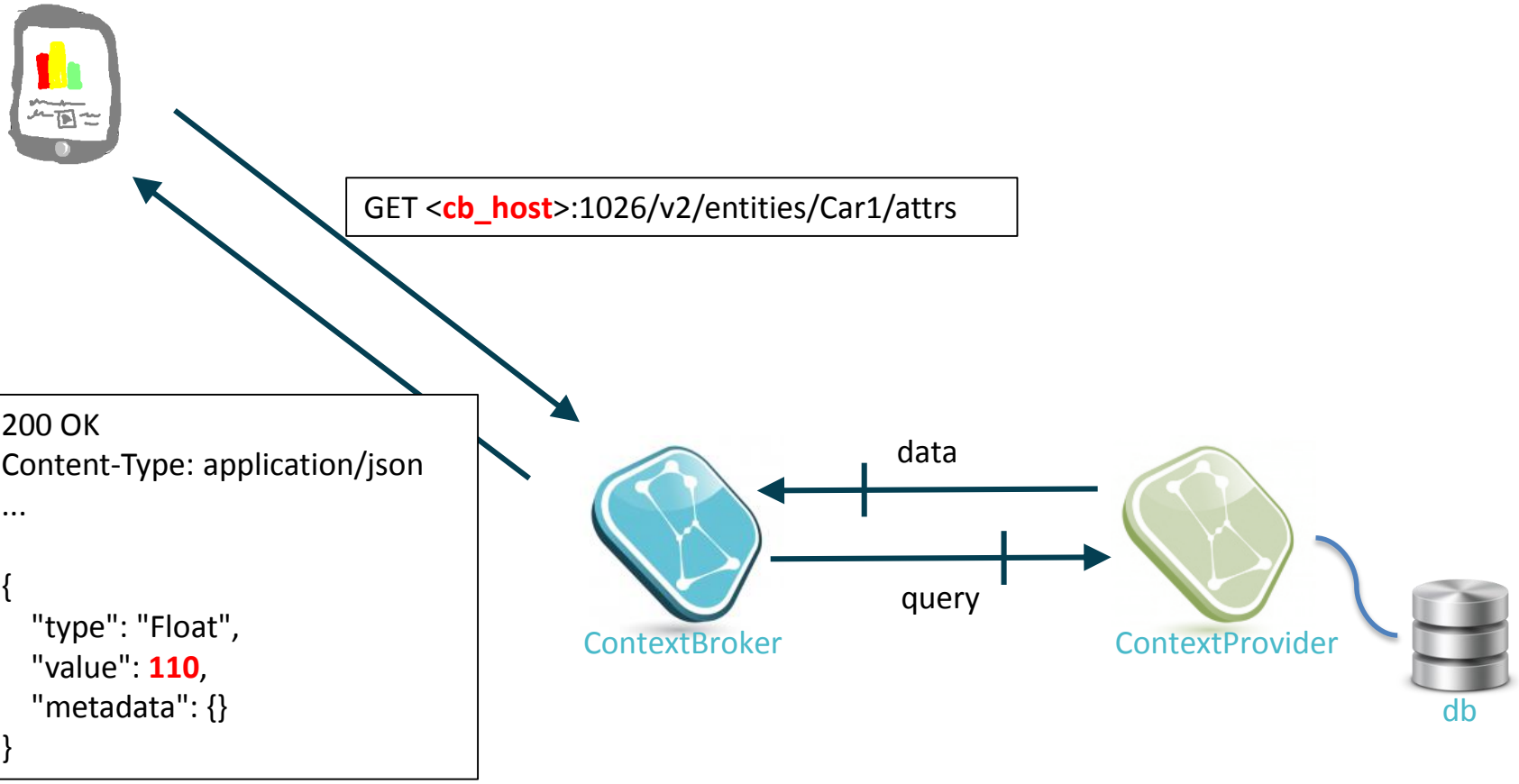
```
POST <cb_host>:1026/v1/registry/registerContext
...
{
   "contextRegistrations": [
      {
         "entities": [
            {
               "type": "Car",
               "isPattern": "false",
               "id": "Car1"
            },
            "attributes": [
            {
               "name": "speed",
               "type": "float",
               "isDomain": "false"
            }
         ],
         "providingApplication": "http://contextprovider.com/Cars"
      }
   ],
   "duration": "P1M"
}
```



```
200 OK
...
{
  "duration" : "P1M",
  "registrationId" : "52a744b011f5816465943d58"
}
```

Context management availability functionality not yet specified in NGSIv2. Thus, a NGSIv1 operation is used to create the registration.

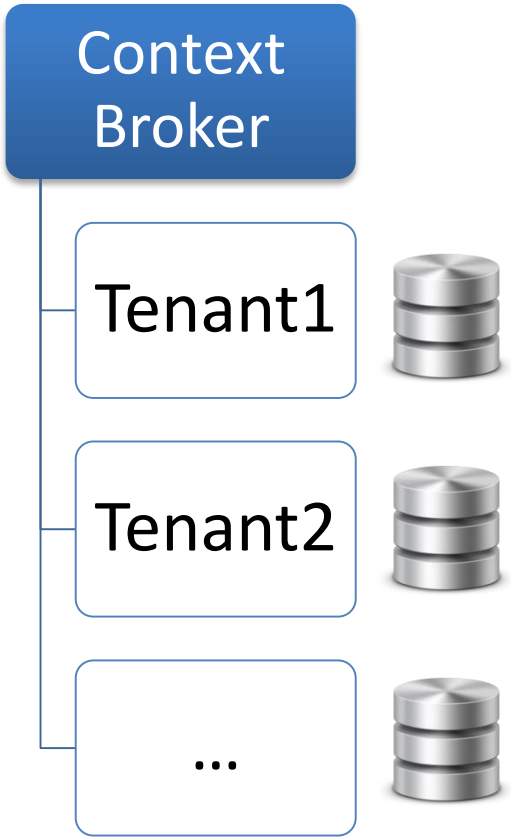
Registration & Context Providers



Multitenancy

- Simple multitenant model based on logical database separation.
- It eases tenant-based authorization provided by other components.
- Just use an additional HTTP header called "Fiware-Service", whose value is the tenant name. Example:

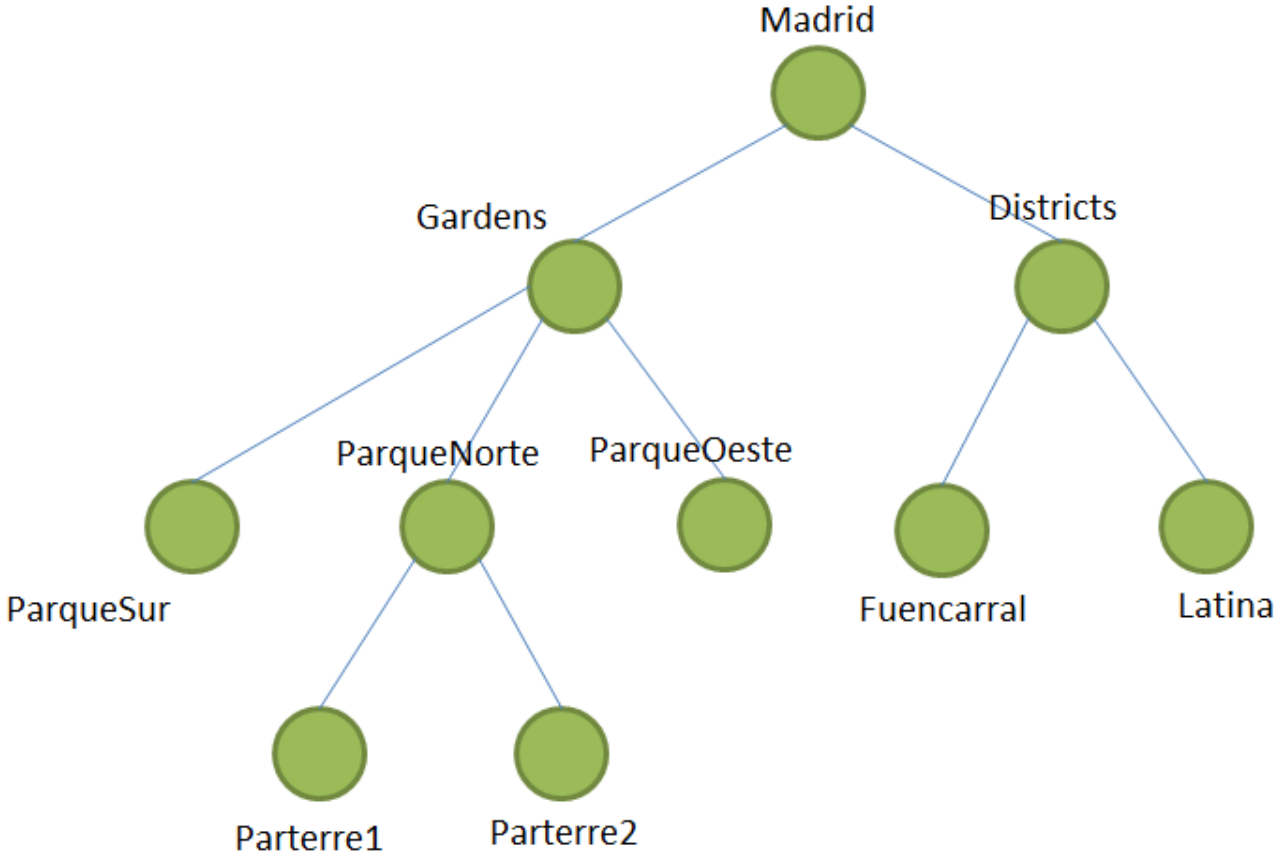
  Fiware-Service: Tenant1



FIWARE

Service Paths

- A service path is a hierarchical scope assigned to an entity at creation time (with POST /v2/entities).



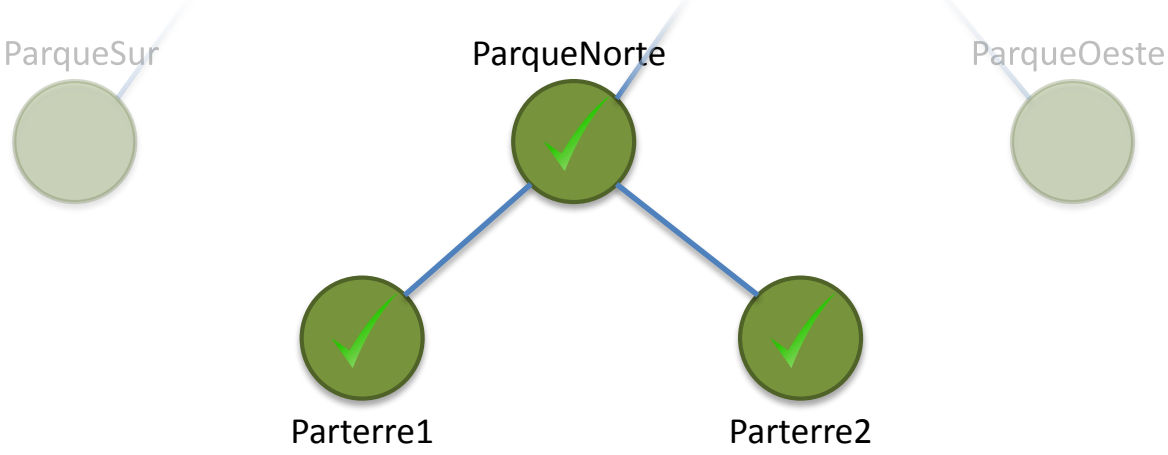
FIWARE

Service Paths

- In order to use a service path we put in a new HTTP header called "Fiware-ServicePath". For example:

  Fiware-ServicePath: /Madrid/Gardens/ParqueNorte/Parterre1

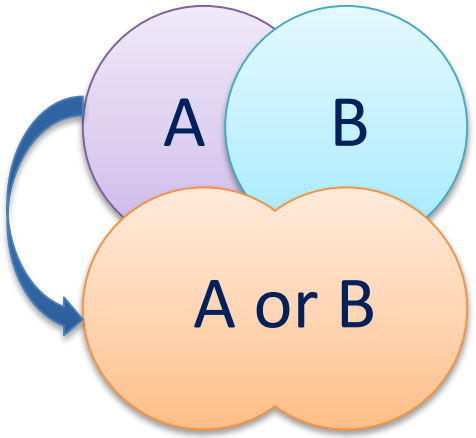
- Properties:
  - A query on a service path will look only into the specified node
  - Use "ParentNode/#" to include all child nodes
  - Queries without Fiware-ServicePath resolve to "/#"
  - Entities will fall in the "/" node by default

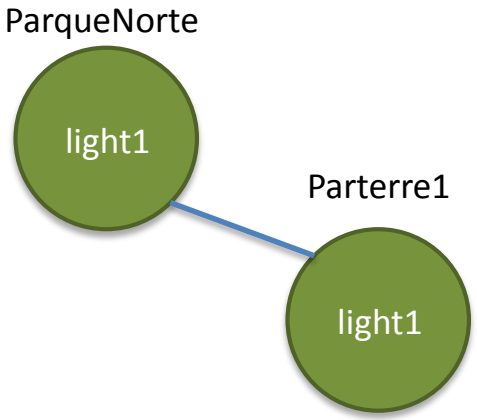


Service Paths

- Properties (continued):
  - You can OR a query using a comma (,) operator in the header
    - For example, to query all street lights that are either in ParqueSur or in ParqueOeste you would use:

      ServicePath: Madrid/Gardens/ParqueSur, Madrid/Gardens/ParqueOeste
    - You can OR up to 10 different scopes.
  - Maximum scope levels: 10
    - Scope1/Scope2/.../Scope10
  - You can have the same element IDs in different scopes (be careful with this!)
  - You can't change scope once the element is created
  - One entity can belong to only one scope
  - It works not only with queries, but also with subscriptions/notifications
  - It works not only in NGSI10, but also with registrations/discoveries (NGSI9)





Where (and when ☺) to go after this talk?

- Interesting NGSI&Orion-related stuff during FIWARE Summit
  - **Wednesday 14th 12:45-13:30:** "NGSI: geoqueries and Carto integration", Fermín Galán & Francisco Romero (Data Team)
  - **Wednesday 14th, 15:45-16:30:** "Creating context historic using Cygnus", Francisco Romero (Data team)
  - **Wednesday 14th 18:15-18:45** "NGSIv2-Overview-for-Developers-That-Already-Know-NGSIv1", Fermín Galán (Orion Context Broker Development Lead)
  - **Thursday 15th 12:15-13:00**, "Hands-on FIWARE Context Provider Simulator Tutorial", German Toro (Data team)

Would you like to know more?

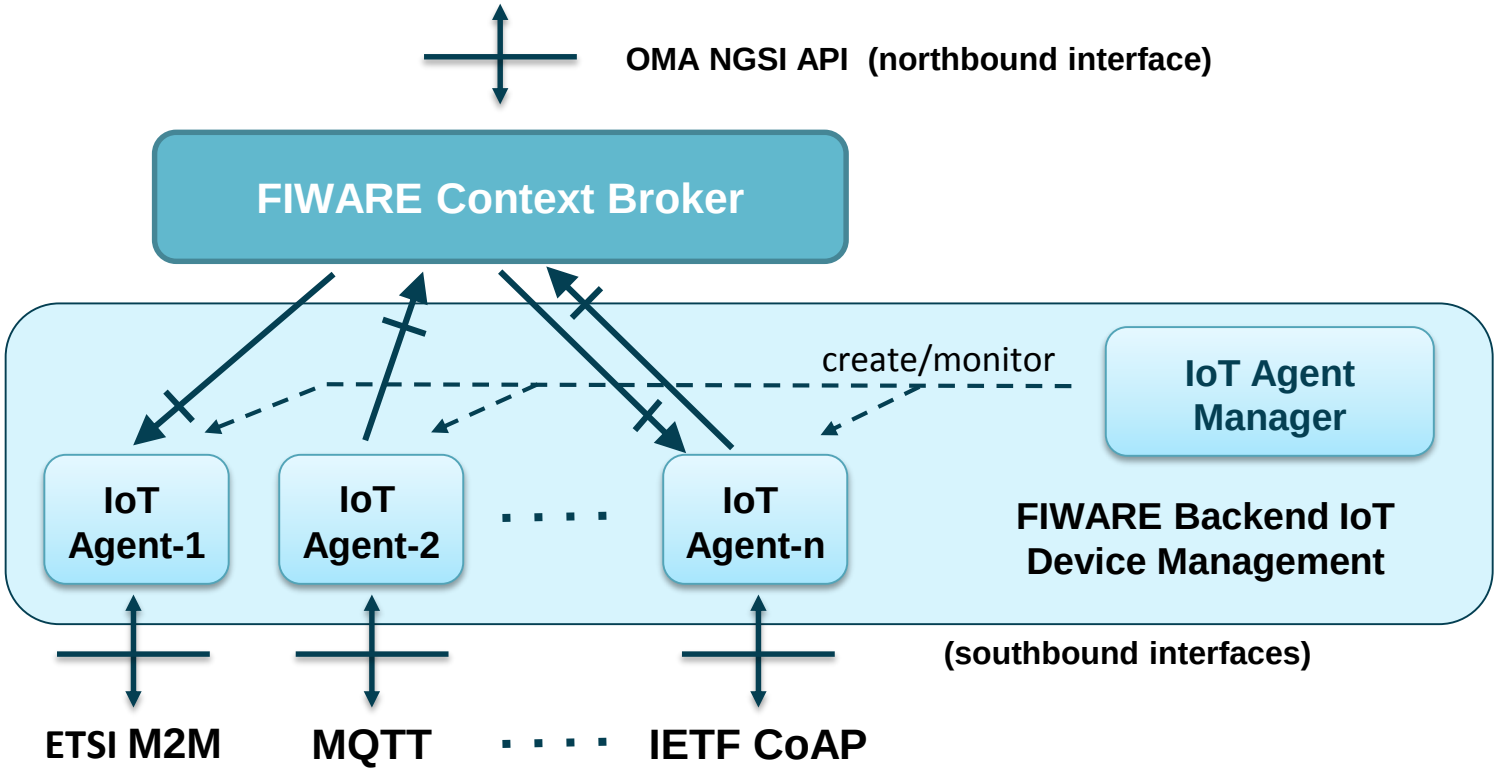
- The easy way
  - This presentation: google for "fermingalan slideshare" and search the one named "Managing Context Information at large scale"
  - Orion User Manual: google for "Orion FIWARE manual" and use the first hit
  - Orion Catalogue page: google for "Orion FIWARE catalogue" and use the first hit
- References
  - NGSIv2 Specification
    - http://fiware.github.io/specifications/ngsiv2/stable
    - http://fiware.github.io/specifications/ngsiv2/latest
  - NGSIv2 for NGSIv1 developers
    - http://bit.ly/ngsiv2-vs-ngsiv1
  - This presentation
    - http://www.slideshare.net/fermingalan/fiware-managing-context-information-at-large-scale
  - Orion Catalogue:
    - http://catalogue.fiware.org/enablers/publishsubscribe-context-broker-orion-context-broker
  - Orion support trhough StackOverflow
    - Ask your questions using the "fiware-orion" tag
    - Look for existing questions at http://stackoverflow.com/questions/tagged/fiware-orion

Thank you!

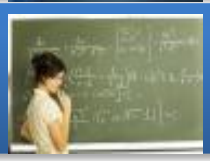
http://fiware.org
Follow @FIWARE on Twitter

Integration with sensor networks

- The backend IoT Device Management GE enables creation and configuration of NGSI IoT Agents that connect to sensor networks
- Each NGSI IoT Agent can behave as Context Consumers or Context Providers, or both

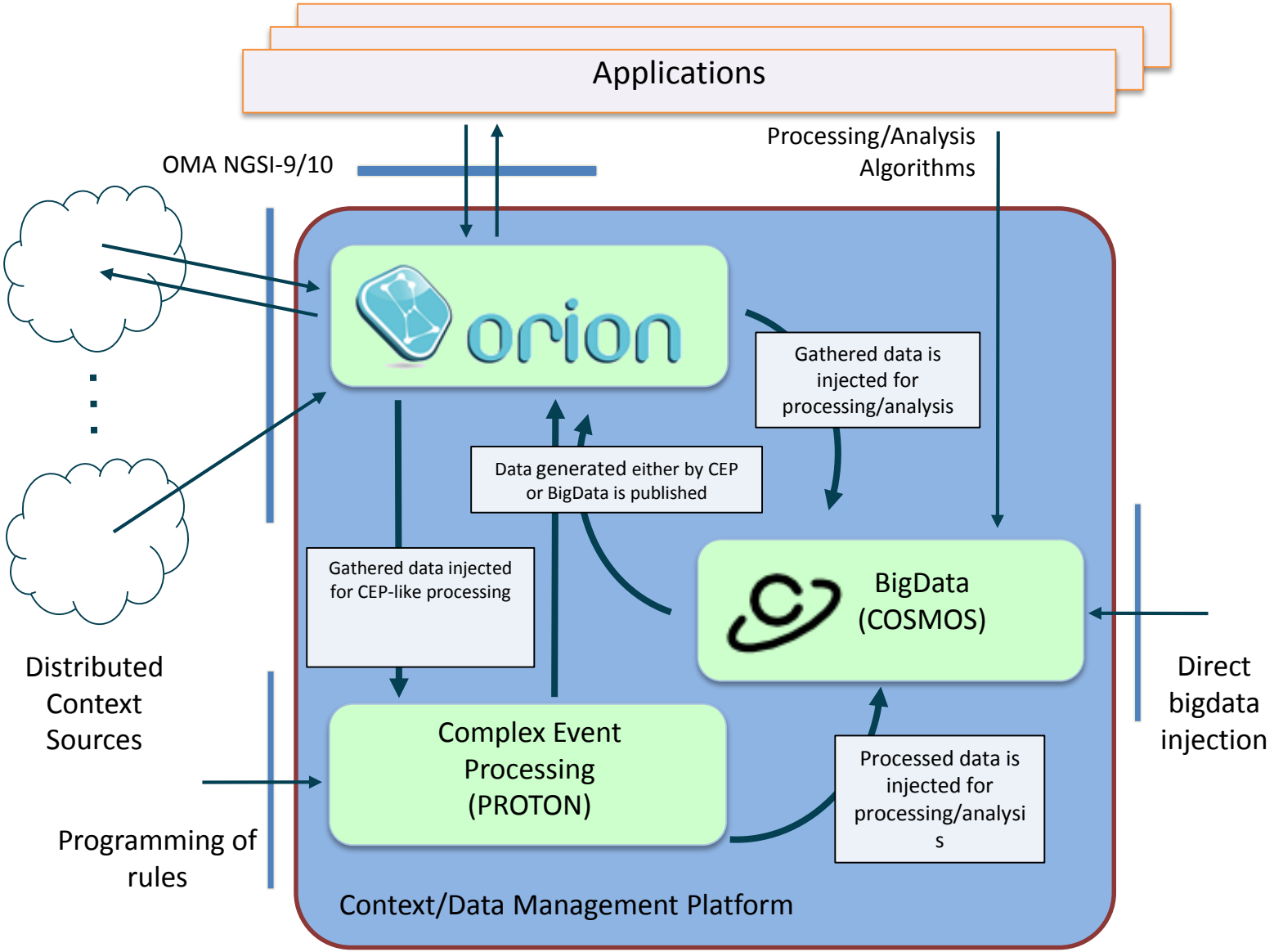


Context Management in FIWARE

Cloud		• Federation of infrastructures (private/public regions) • Automated GE deployment
Data		• Complete Context Management Platform • Integration of Data and Media Content
IoT		• Easy plug&play of devices using multiple protocols • Automated Measurements/Action ←→Context updates
Apps		• Visualization of data (operation dashboards) • Publication of data sets/services
Web UI		• Easy support of UIs with advanced web-based 3D and AR capabilities • Visual representation of context information.
I2ND		• Advanced networking capabilities (SDN) and Middleware • Interface to robots
Security		• Security Monitoring • Built-in Identity/Access/Privacy Management

FIWARE

FI-WARE Context/Data Management Platform



Special update action types

- Used by **/v2/op/update** (batch operation)
- Conventional actionTypes
  - APPEND: append (or update if the attribute already exists)
  - UPDATE: update
  - DELETE: delete
- Special actionTypes
  - APPEND_STRICT: strict append (returns error if some of the attributes to add already exists)
  - REPLACE: delete all the entity attributes, next append the ones in the update request