

Managing Context Information at large scale (Introduction)



www.fiware.org
[@Fiware](https://twitter.com/Fiware) 

Contact twitter
[@fermingalan](https://twitter.com/fermingalan)



Contact email
fermin.galanmarquez@telefonica.com
kengunnar.zangelin@telefonica.com

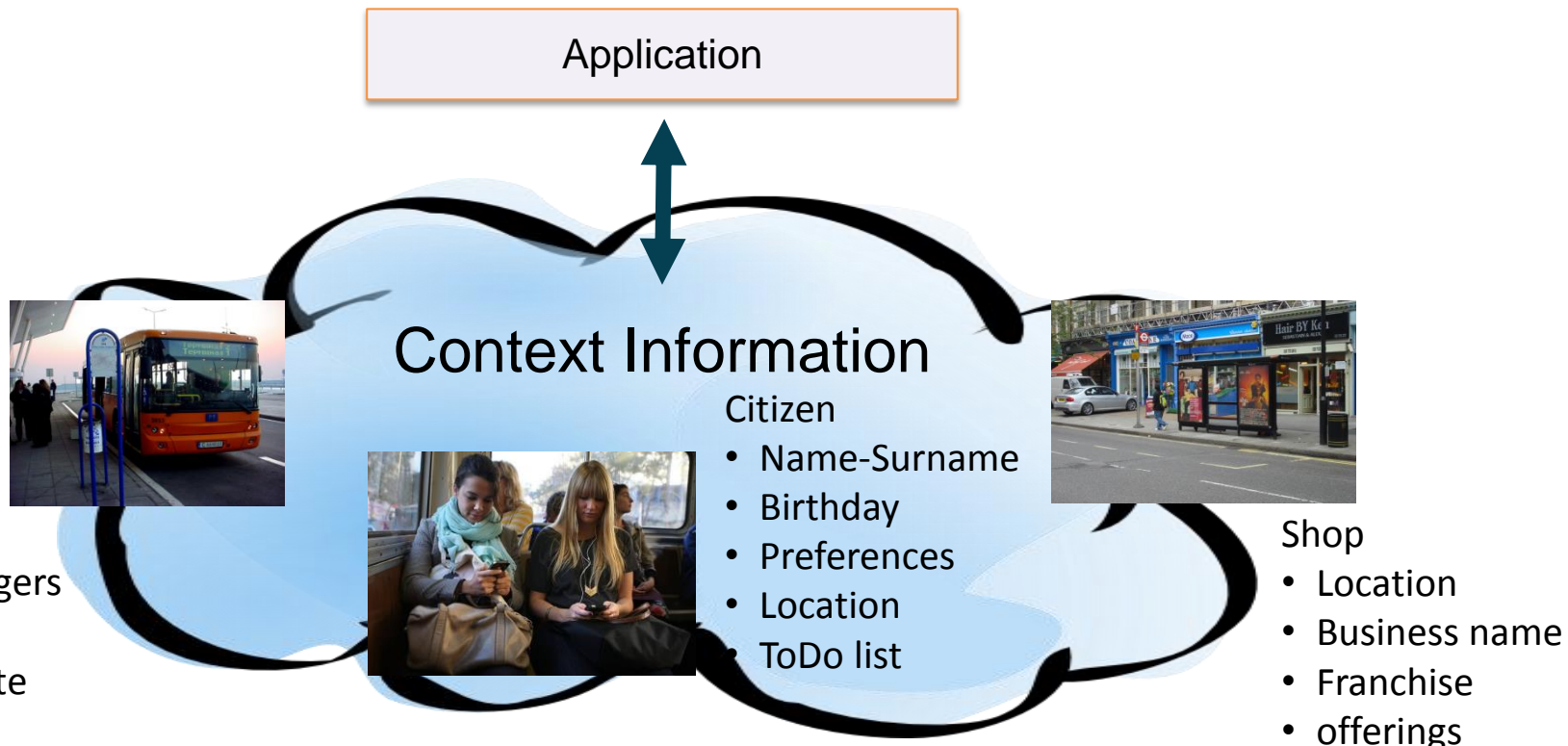
Introduction

Orion Context Broker

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

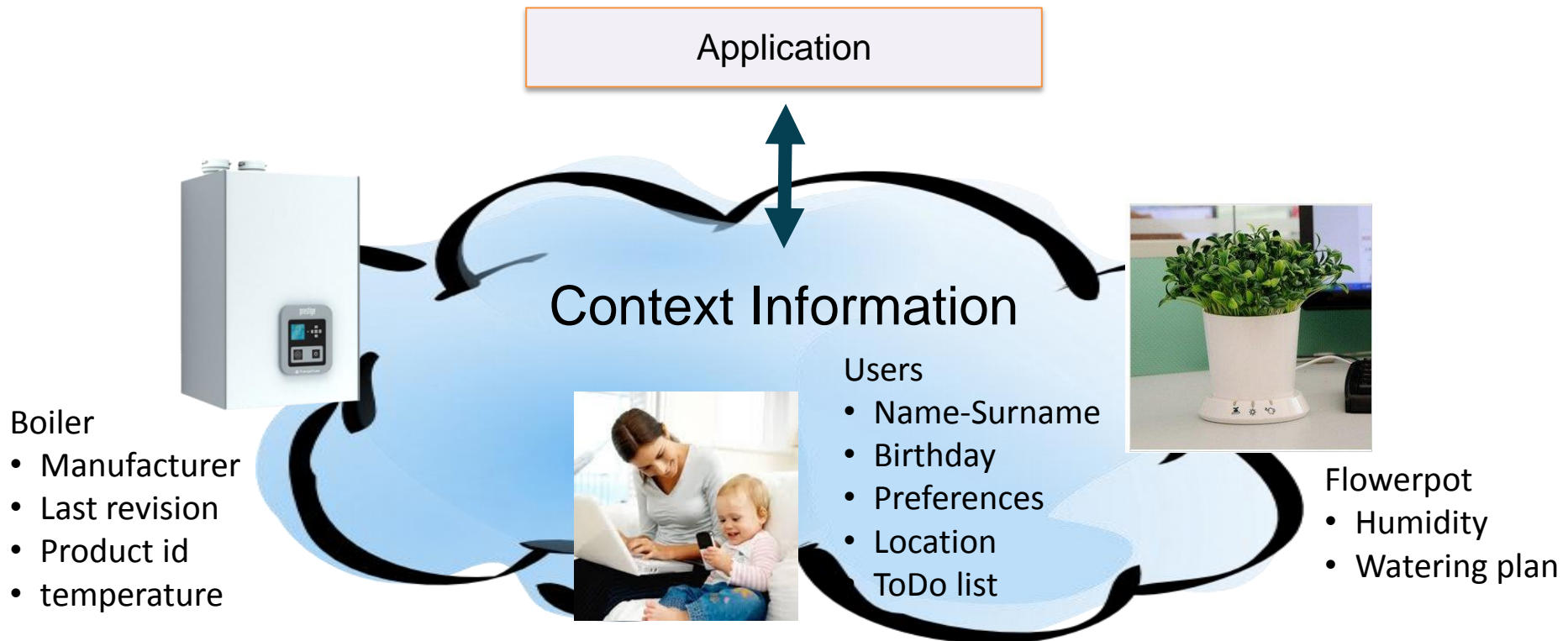
Being “Smart” requires first being “Aware”

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



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



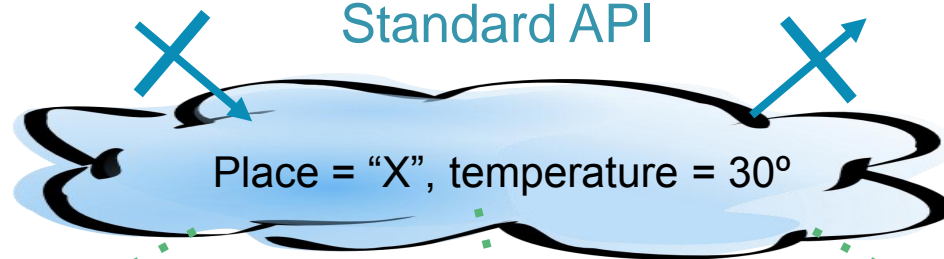
Different sources of context need to be handle

- 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

What's the current temperature in place "X"?

Notify me the changes of temperature in place "X"

Standard API



Place = "X", temperature = 30°



A sensor in a pedestrian street

It's too hot!



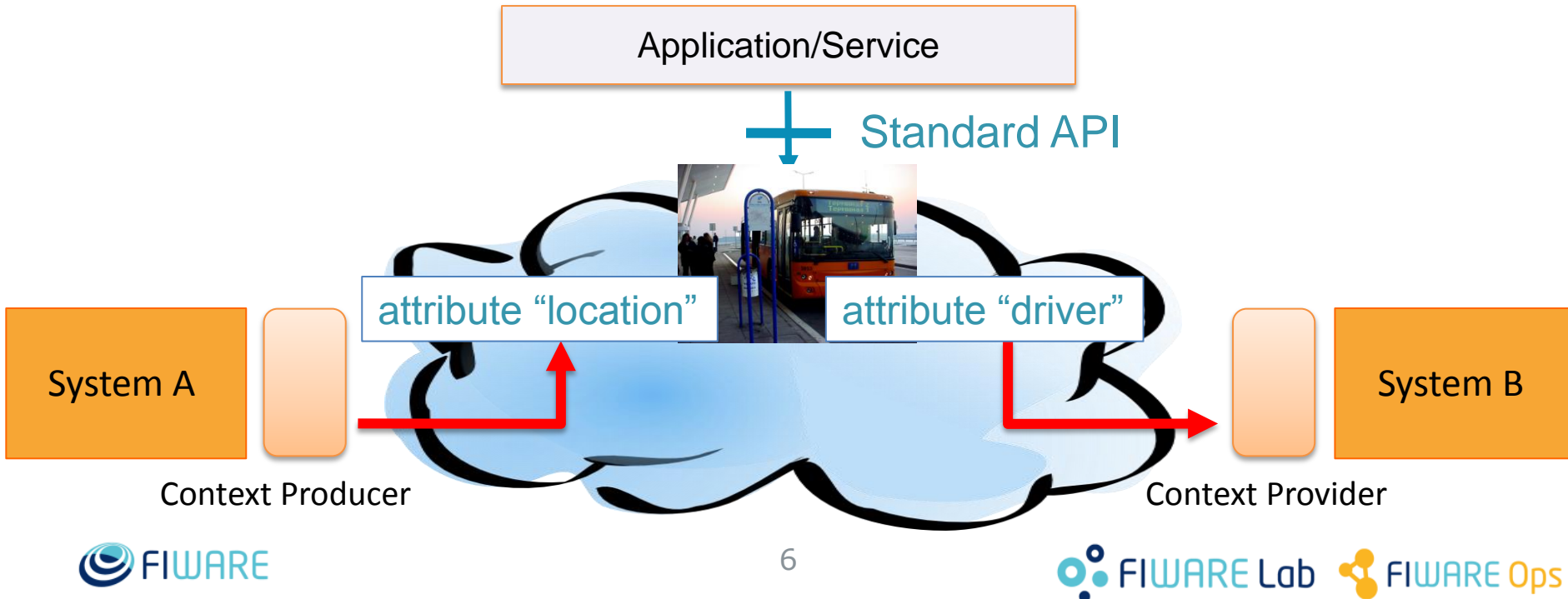
A person from his smartphone



The Public Bus Transport Management system

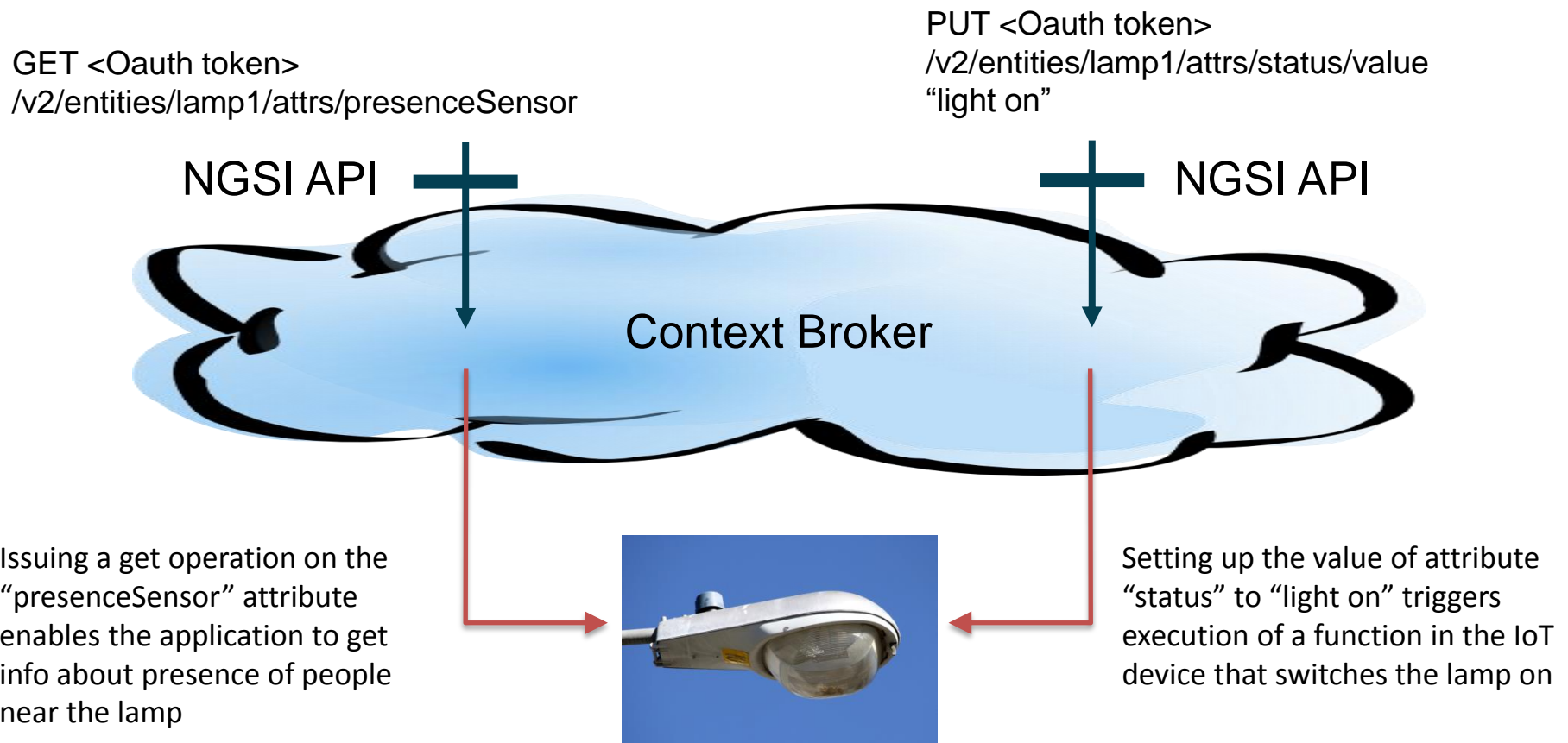
A non-intrusive approach is required

- 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



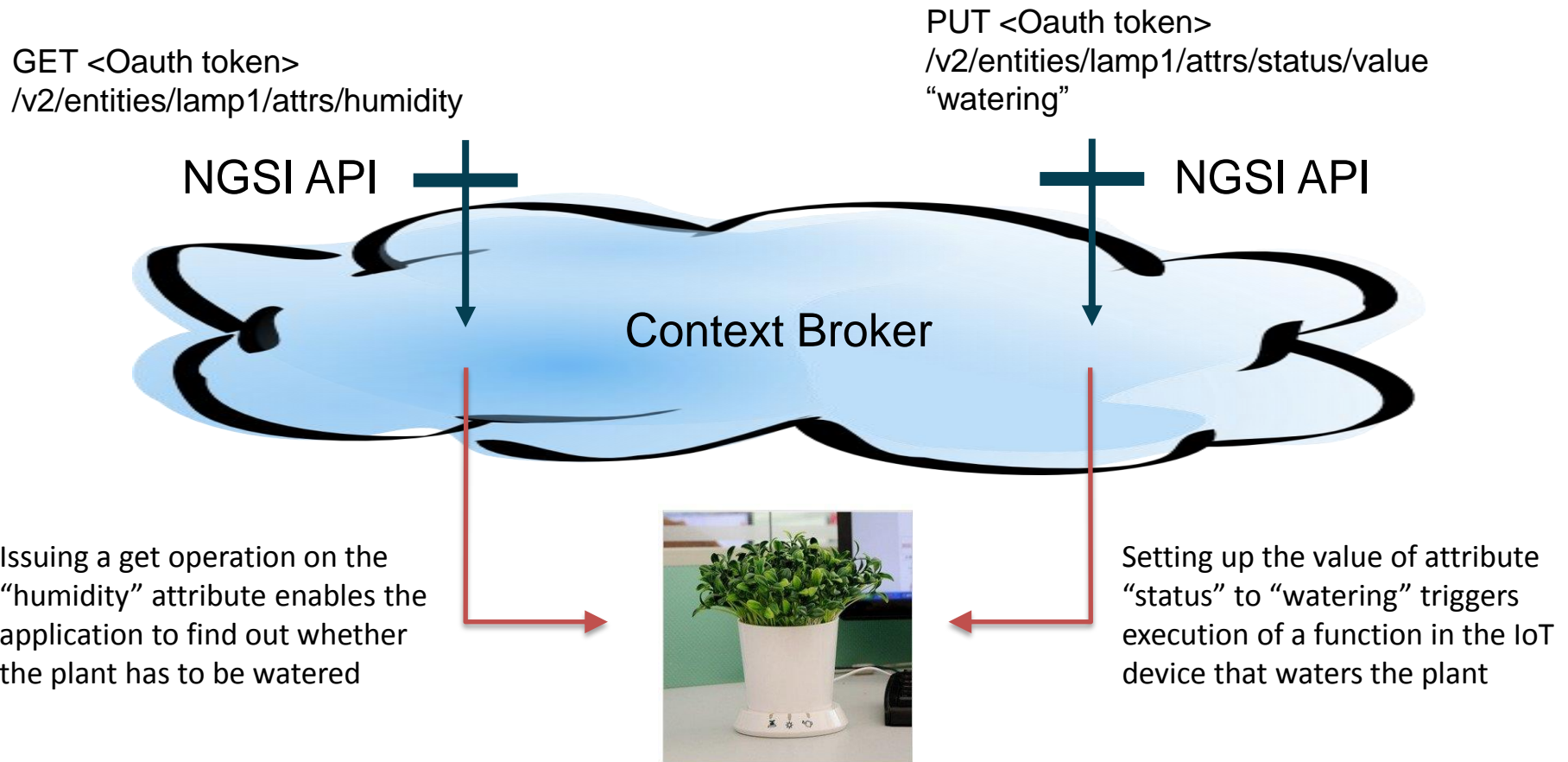
FIWARE NGSI: “The SNMP for IoT”

- 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



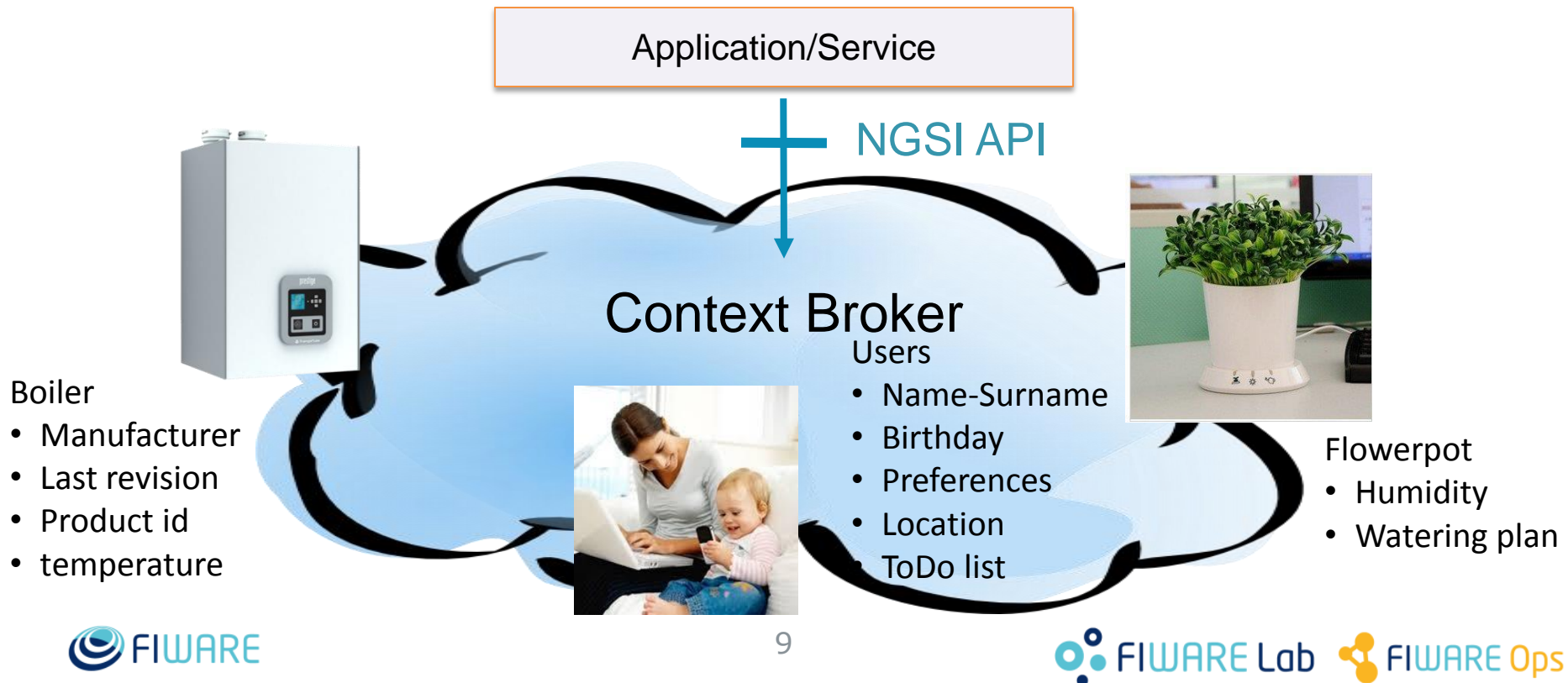
Connecting to the Internet of Things

- 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



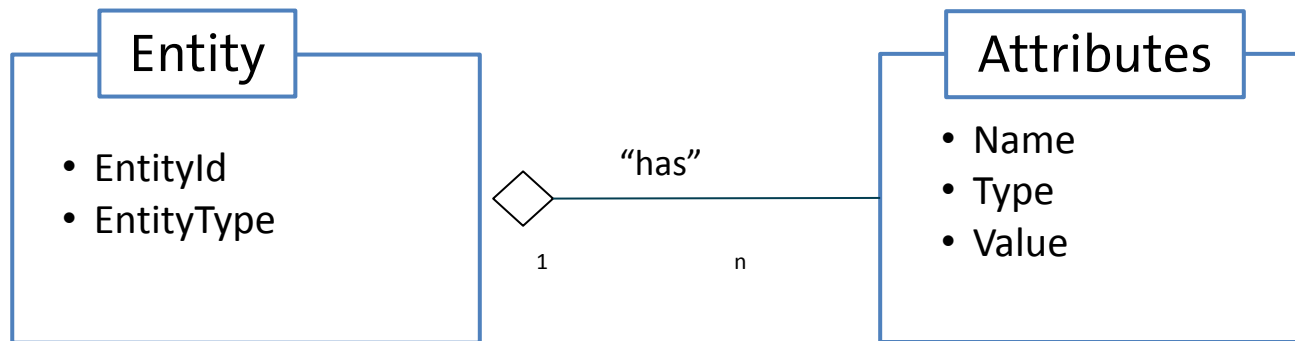
Context Management in FIWARE

- 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) (not yet in NGSIv2)*
- 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)
 - Not yet finished at the present moment (but usable)
 - Current NGSIv2 version is **Release Candidate 2016.05**
 - Uses the **/v2** prefix in resource URL
- Introduction to NGSIv2
 - https://docs.google.com/presentation/d/1_fv9dB5joCsOCHlb4Ld6A-QmeIYhDzHgFHUWreGmvKU/edit#slide=id.g53c31d7074fd7bc7_0

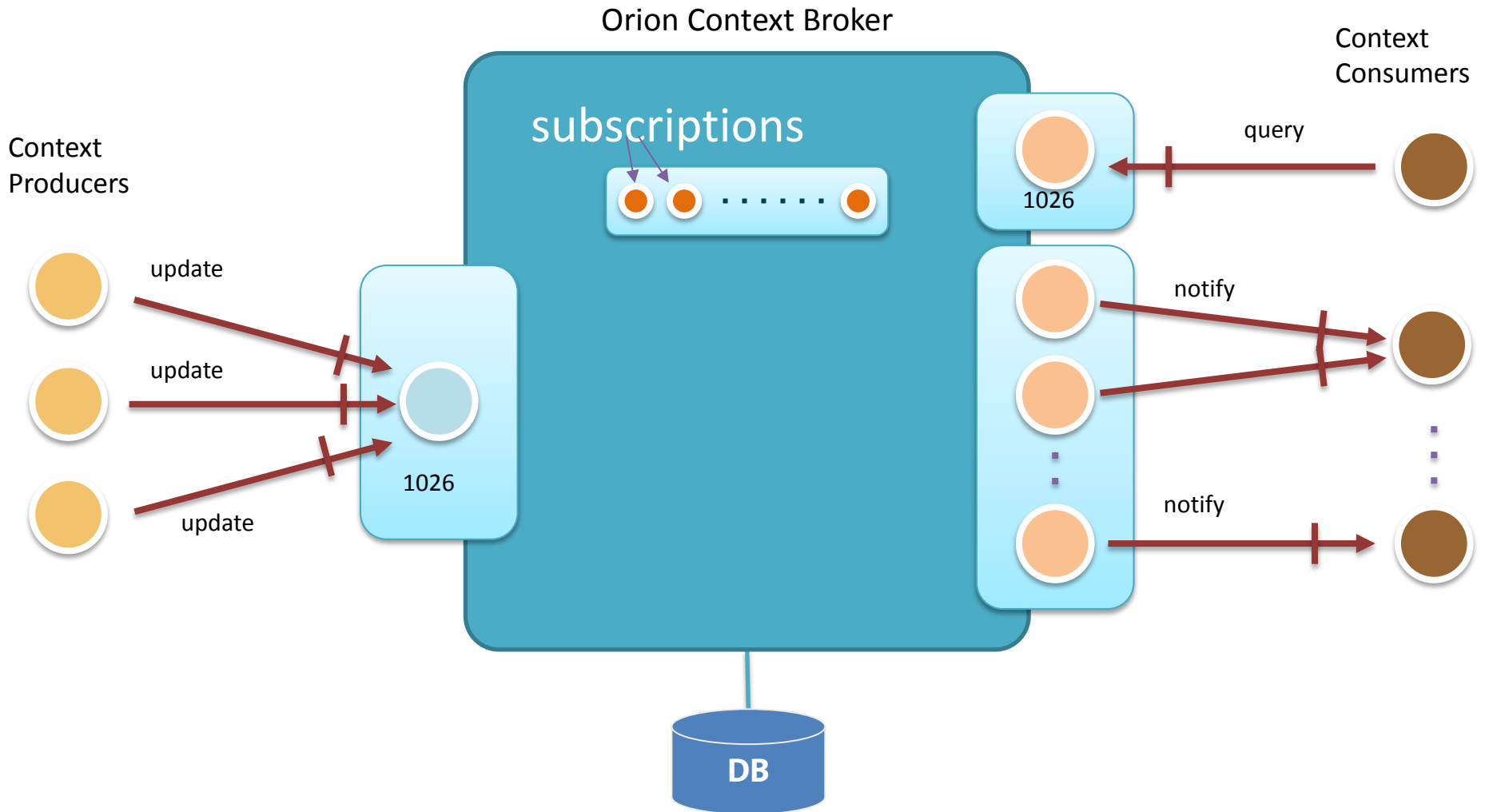
NGSIv2 status (AKA the “NGSIv2 disclaimer”)

- 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 outgoing 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 context management availability functionality (not yet implemented in NGSIv2)
 - 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" : "1.3.0",  
    "uptime" : "7 d, 21 h, 33 m, 39 s",  
    "git_hash" : "af44fd1fbdbbfd28d79ef4f929e871e515b5452e",  
    "compile_time" : "Tue Jun 15 11:52:53 CET 2016",  
    "compiled_by" : "fermin",  
    "compiled_in" : "centollo"  
  }  
}
```



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

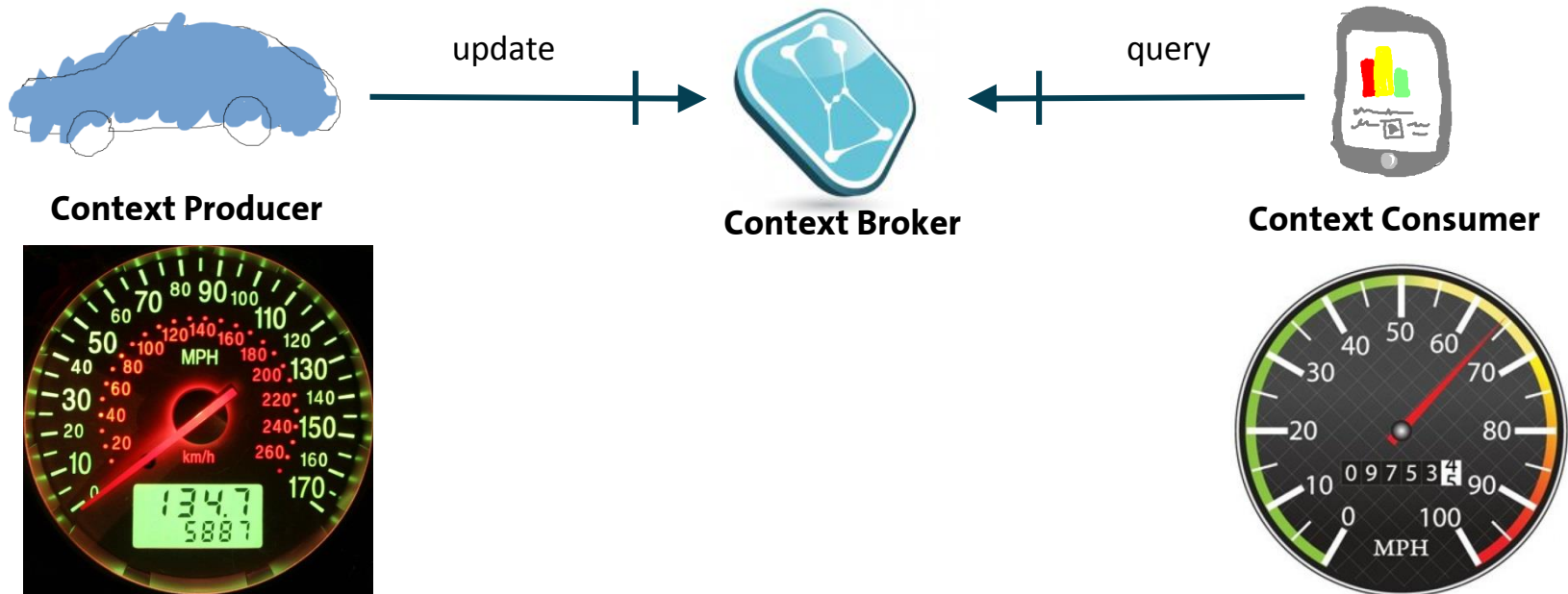
Orion Context Broker Basic Operations

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
```



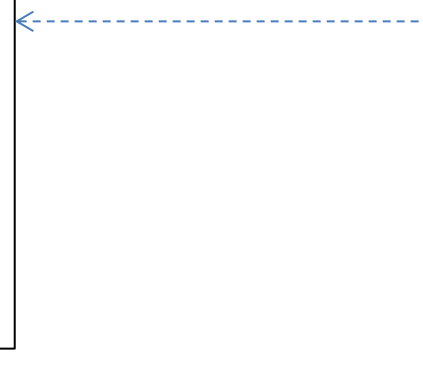
Quick Usage Example: Car Speed Update (1)

```
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
```

```
...
```



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
```

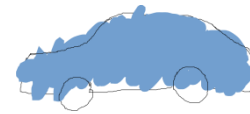
Quick Usage Example: Car Speed Update (2)

```
PUT <cb_host>:1026/v2/entities/Car1/attrs/speed/value
```

```
Content-Type: text/plain
```

```
...
```

```
115
```



```
204 No Content
```

```
...
```



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  
  }  
}
```



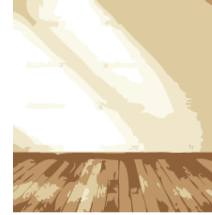
```
201 Created
```

```
...
```



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

...



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
```



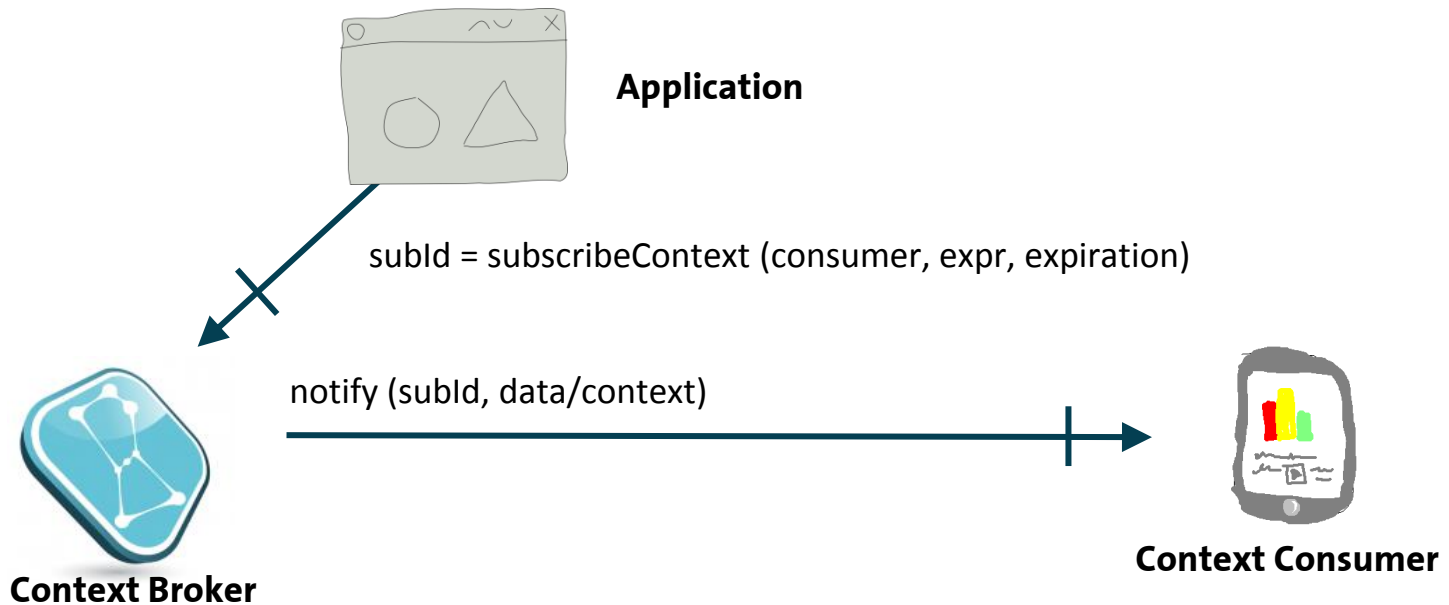
The full description of the Simple Query Language for filtering can be found in the NGSIv2 Specification document

```
200 OK
Content-Type: application/json
...
[
  {
    "id": "Room1",
    "pressure": 720,
    "temperature": 25,
    "type": "Room"
  }
]
```



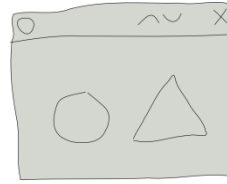
Context Broker operations: push data

- **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

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",
      "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**

Content-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)

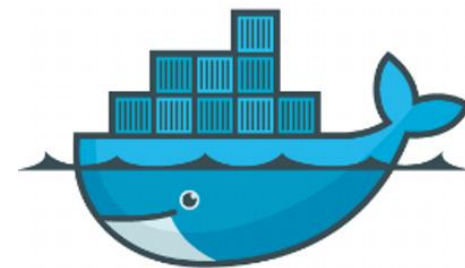
- FIWARE Lab image
 - Image: **orion-psb-image-R<x>.<y>**
- VirtualBox image
 - <http://bit.ly/fiware-orion024-vbox> (it's big!)
 - User/pass:
 - fiware/fiware
 - root/fiware
- **Hint:** update Orion package once the VM is deployed



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 Context Broker to Backbone Sync

- <https://github.com/digitalilusion/o2bb>

Introduction

- **FIWARE Orion Context Broker** (aka Orion) is a **General Enabler** (GE) that provides object allocation, query and Pub/Sub operations. Formally, Orion is implemented following the NGS10/NGSI9 specification.
- **BackboneJS** is a JavaScript framework which enables us to build rich web apps.
- **Orion 2 Backbone** (aka O2BB) is "the glue" between these entities. With O2BB we can integrate our project quickly in an easy way with FIWARE's Orion.

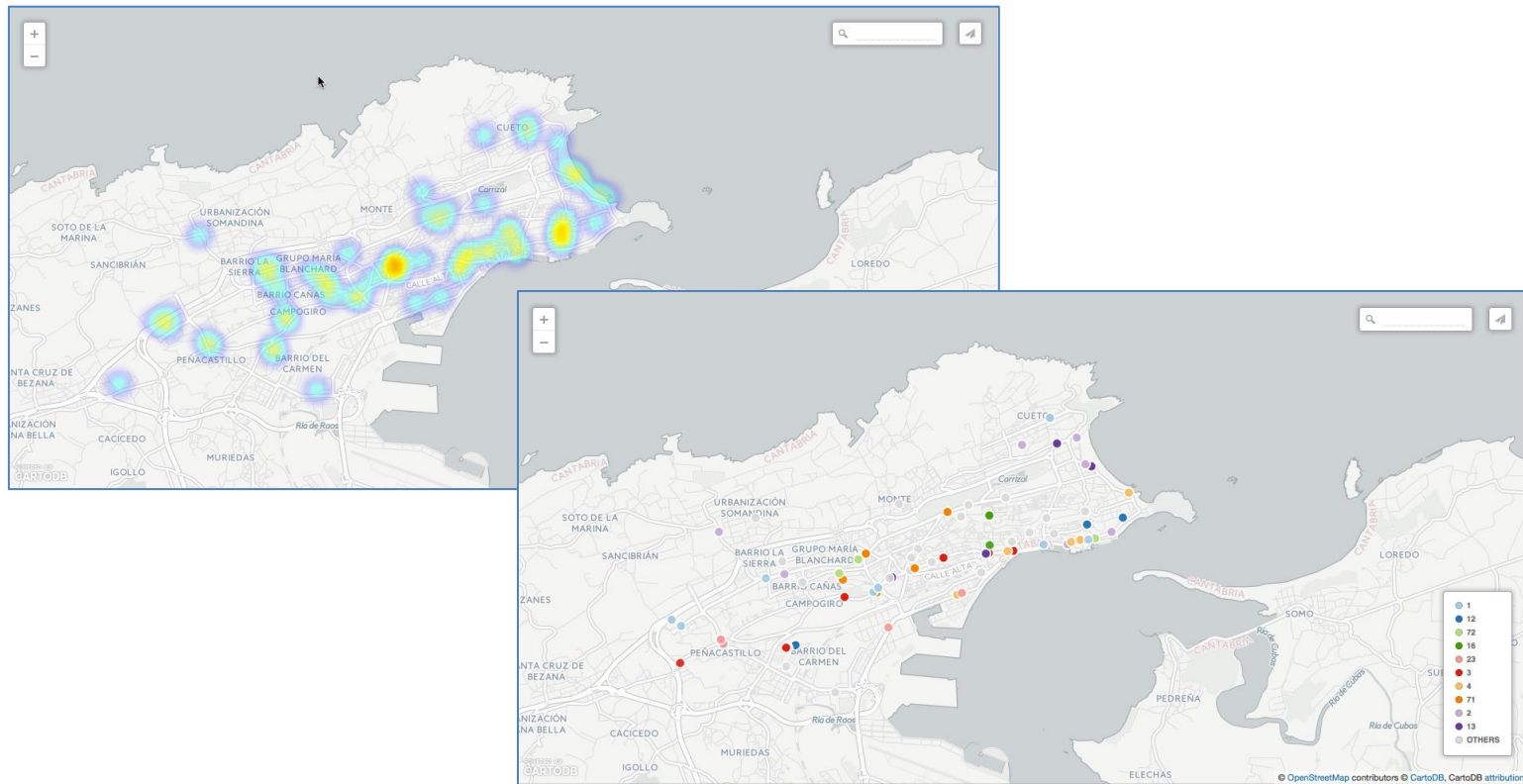
O2BB enables to companies accelerated by FIWARE a rapid, easy and cross-platform integration with Orion GE

We've developed O2BB for **Outbarriers project**

NGSI Context Adaptor for CartoDB

Show your entities in a map with no effort, create history animations, heat maps and clusters representations

- <https://github.com/telefonicaid/fiware-dataviz>



NGSI Plugin for Freeboard

Create a real time dashboard for your entities, representing gauges, spark lines and maps. No coding required!

- <https://github.com/telefonicaid/fiware-dataviz>

In addition, Freeboard freemium version integrates Orion off-the-shelf

The image shows a two-part screenshot. The top part is a configuration window for a 'DATASOURCE' in Freeboard. It is set to connect to an Orion Context Broker. A dropdown menu for 'TYPE' is open, showing 'FIWARE Orion' selected. Other fields include 'NAME' (Clock), 'HOST' (Dweet.io), 'THINGPROXY' (Public), and 'FIWARE-SERVICE' (Freeboard). The bottom part of the image shows a live dashboard with several widgets: three battery status gauges (24%, 17%, and 62%), a map of a location in Las Tablas, a 'CONDITION' widget showing 'Clear', and a 'WIND' widget showing 6.3 m/s. The FIWARE logo is in the bottom left, and 'ps' is in the bottom right.

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://telefonicaid.github.io/fiware-orion/api/v2/stable/>
 - 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 through StackOverflow
 - Ask your questions using the “fiware-orion” tag
 - Look for existing questions at <http://stackoverflow.com/questions/tagged/fiware-orion>

Thanks!



www.fiware.org
@Fiware 

(References to Orion manual sections and links in this presentation are valid at time of writing this –September 16th, 2016- but they may change along time)