IoT Sensors Integration and Data Management

FIWARE Context Broker and the storage systems

Antonio Jara
HOP Ubiquitous S.L. (www.hopu.eu)
jara@hopu.eu

OPEN APIs FOR OPEN MINDS

www.fiware.org
@Fiware

www.smartsdk.eu

Introduction

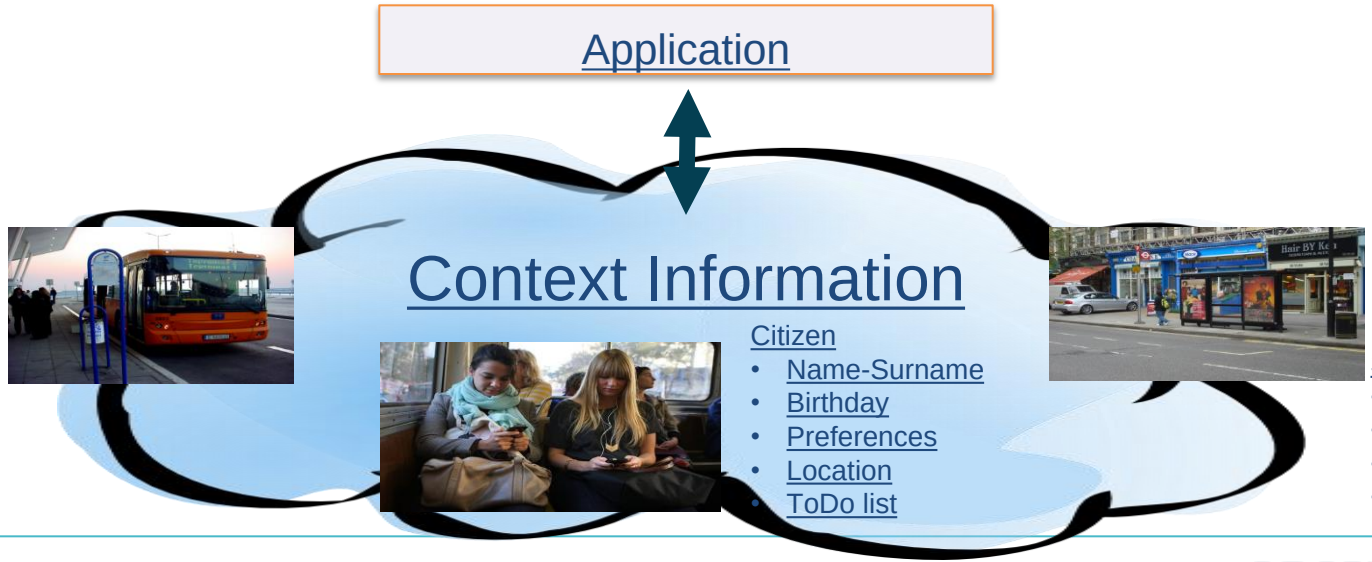
Orion Context Broker

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

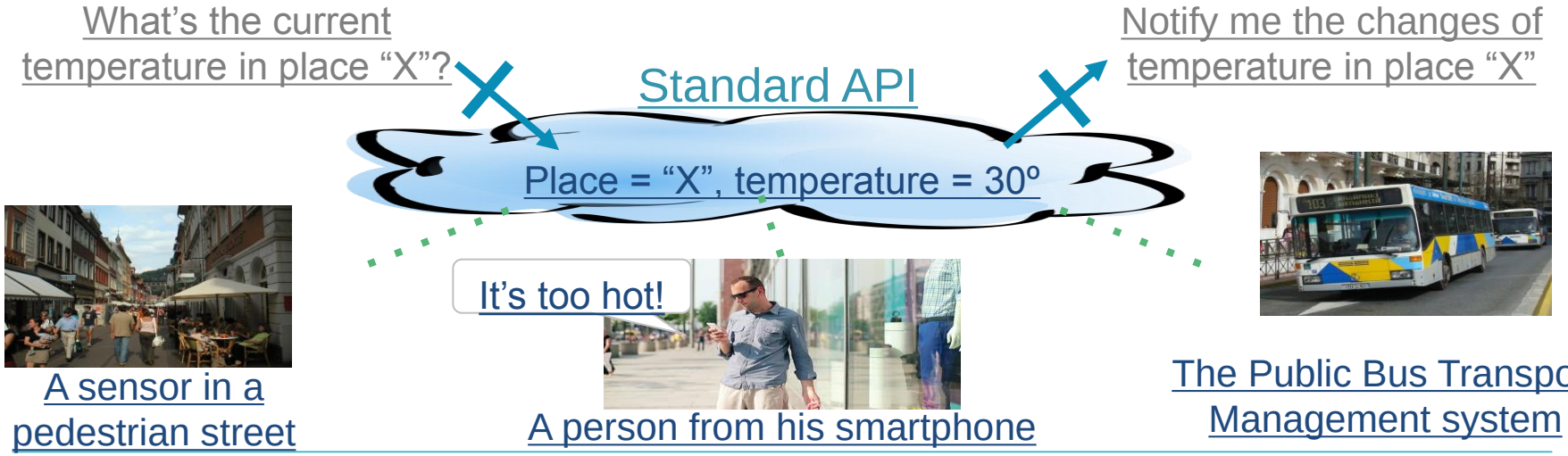
Context information refers to the values of attributes characterizing entities relevant to the application

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

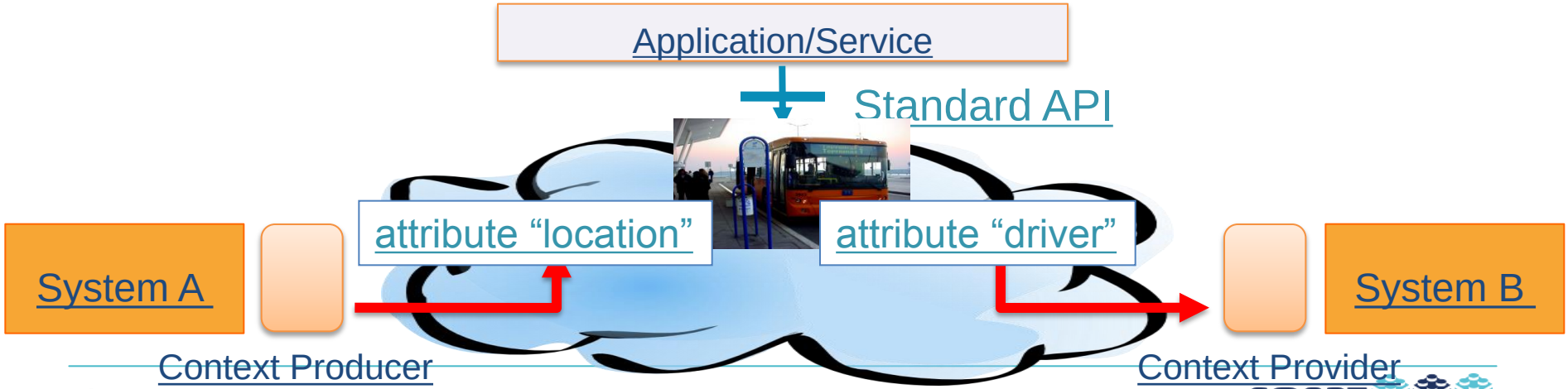


A non-intrusive approach is required

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

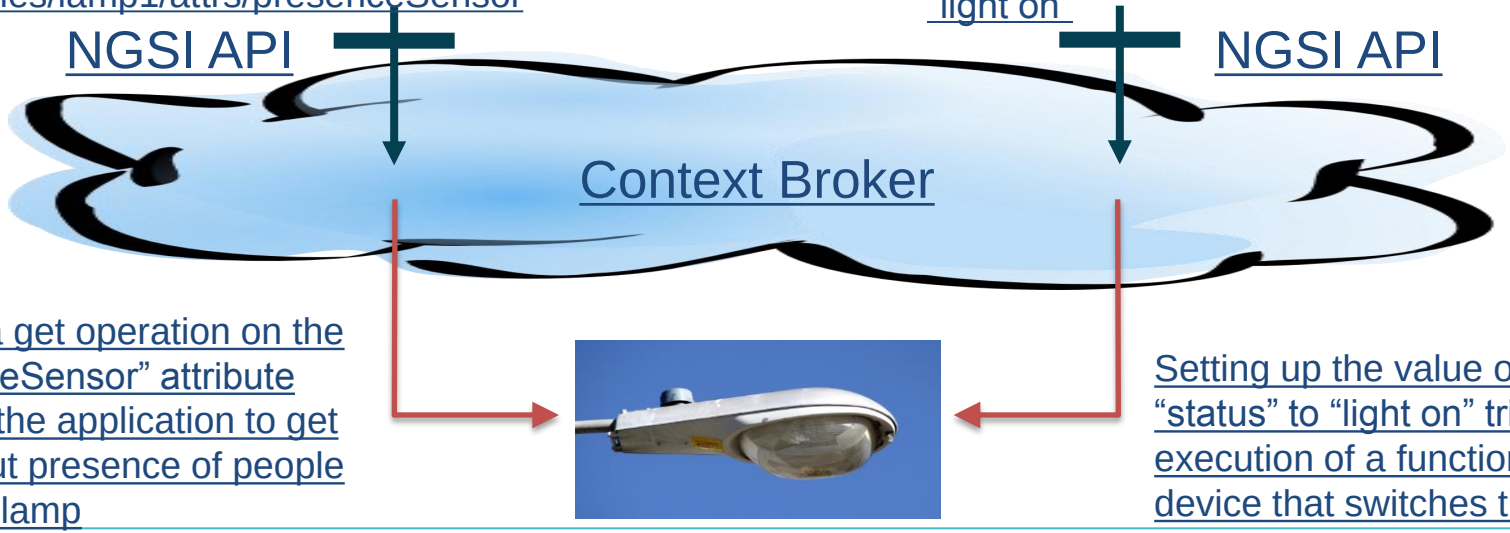


FIWARE NGSI: “The SNMP for IoT”

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

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

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 can manage it

Orion Context Broker

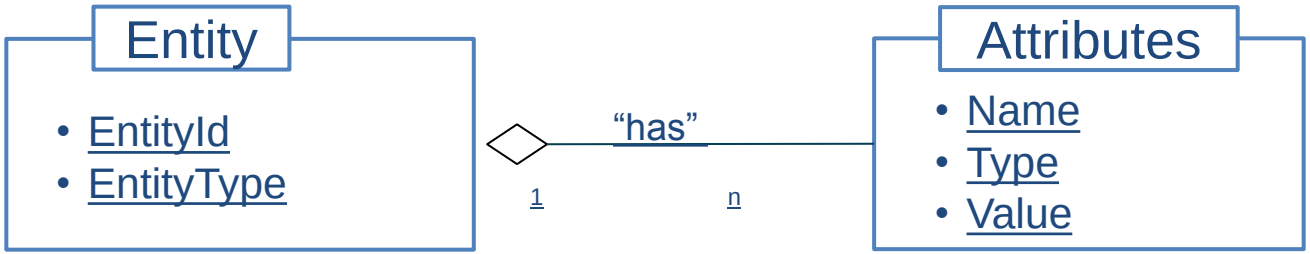
Main functions:

- Context management

HTTP and REST-based

- JSON payload support

Context in NGSI is based in an **entity-attribute** model:



Two “flavors” of NGSI API

NGSIv1

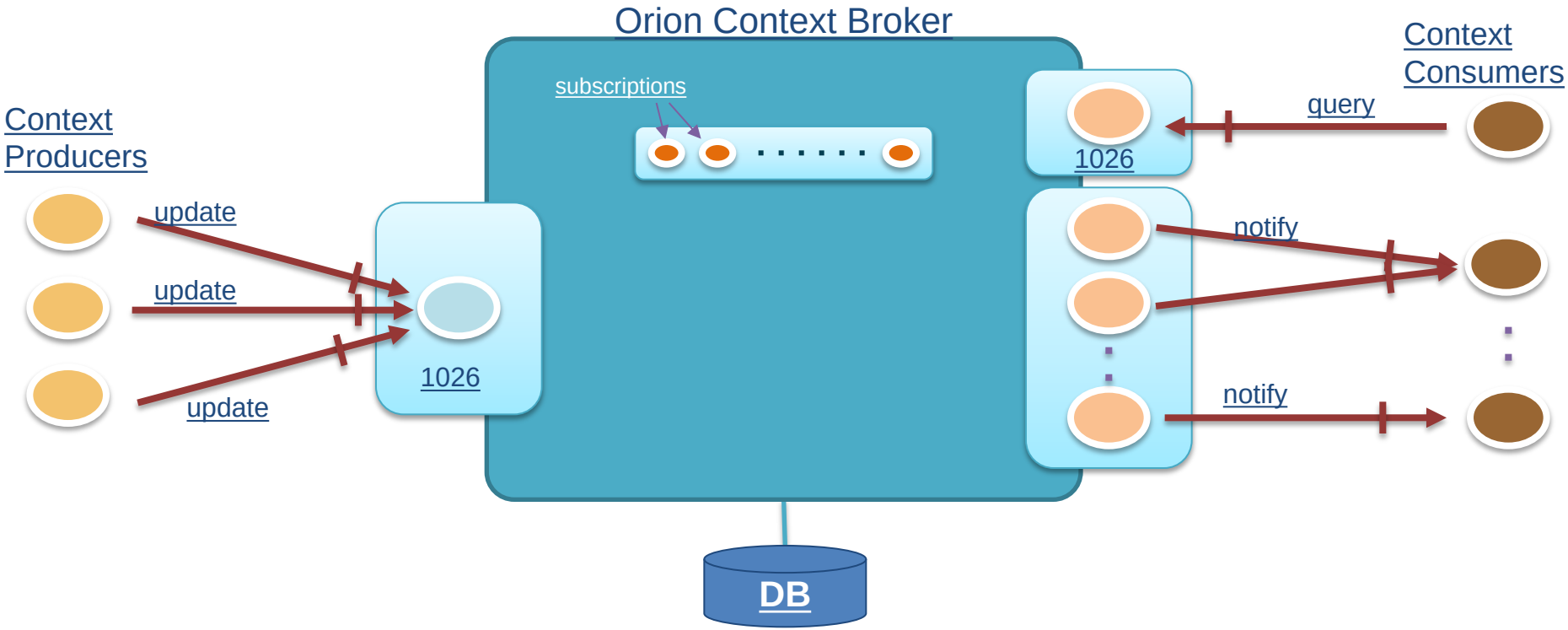
- Original NGSI RESTful binding of OMA-NGSI
- Implemented in 2013
- Uses the ***/v1*** prefix in resource URL

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)
- Uses the ***/v2*** prefix in resource URL

Introduction to NGSIv2: https://docs.google.com/presentation/d/1_fv9dB5joCsOCHlb4Ld6A-QmeIYhDzHgFHUWreGmvKU/edit#slide=id.g53c31d7074fd7bc7_0

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

update

query

Context Producer

Context Broker

Context Consumer

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
```

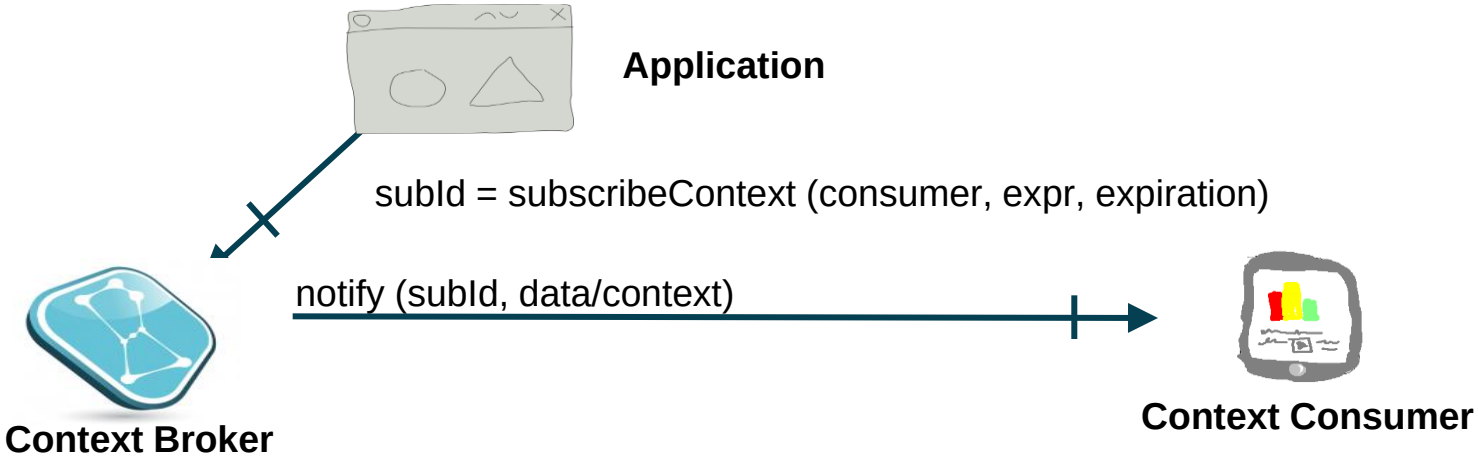
The full description of the Simple Query Language for filtering can be found in the NGSIv2 Specification document

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

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

Context Broker operations: push data

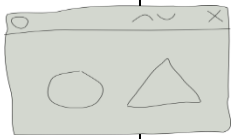
Context Consumers can subscribe to receive context information that satisfy certain conditions using the **subscribe** operation. Such subscriptions may have an expiration time.

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

```
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"
        }
    }
}]
```

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

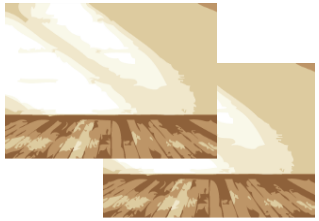
They implement extra functionality that cannot be achieved with RESTful operations, e.g. to create several entities with the same operation

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)

HOP Ubiquitous image

Image:
https://storage.googleapis.com/fiware/CentOS6%20-%20Orion%20-%20Master%201.rar

VirtualBox image (it's big!)

User/pass:
root/hopumaster1

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
```

Advanced Features

Orion Context Broker

Metadata
Compound attribute/metadata values
Type browsing
Geo-location
Query filters
Custom notifications
Attribute/metadata filtering and special attribute/metadata
Registrations & context providers

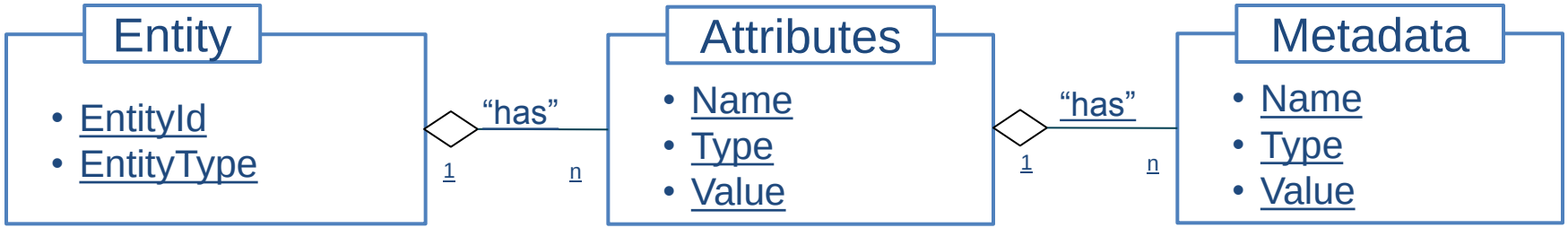
Metadata

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

Geo-location

Entities can have an attribute that specifies its location

Several attribute types can be used

- geo:point (for points)
- geo:line (for lines)
- geo:box (for boxes)
- geo:polygon (for polygons)
- geo:json (for arbitrary geometries, in GeoJson standard)

Example: create an entity called Madrid

…and create a couple more towns:

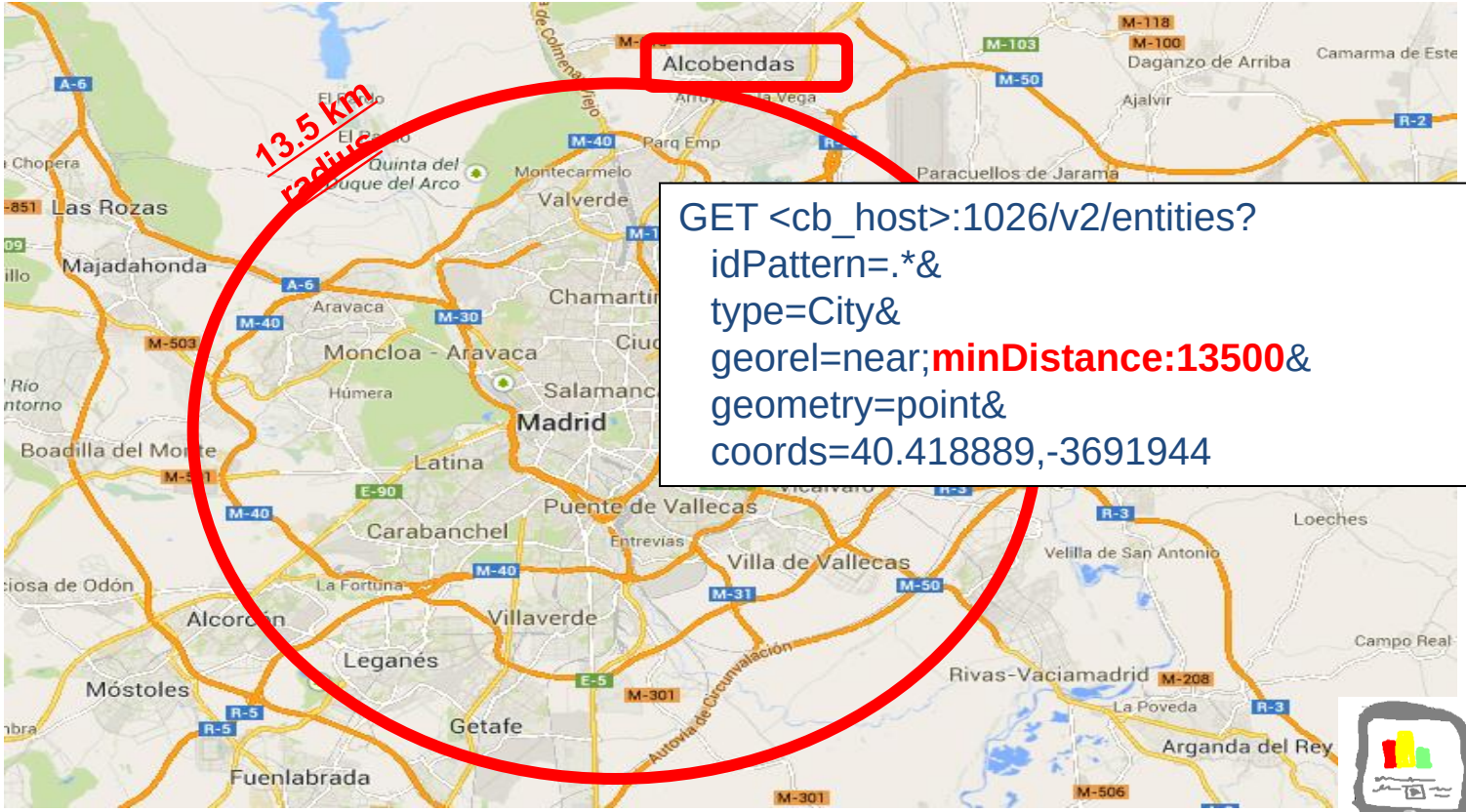
- Leganés
- Alcobendas

```
POST <cb_host>:1026/v2/entities
{
    "type": "City",
    "id": "Madrid",
    "position": {
        "type": "geo:point",
        "value": "40.418889, -3.691944"
    }
}
```

Geo-location – Circle

Geo-location – Max distance

Geo-location – Min distance




```
GET <cb_host>:1026/v2/entities?
  idPattern=.*&
  type=City&
  georel=near;minDistance:13500&
  geometry=point&
  coords=40.418889,-3691944
```

More geo-relationships

Apart from **near**, the following georel can be used

- georel=coveredBy
- georel=intersects
- georel=equals
- georel=disjoint

See NGSIv2 Specification for a detailed description

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)

By **entity type pattern** (regex)

```
GET
<cb_host>:1026/v2/entities?idPattern=^Room[2-5]
```

By **geographical location**

Described in detail in previous slides

```
GET
<cb_host>:1026/v2/entities?typePattern=T[ABC]
```

Filters can be used simultaneously (i.e. like **AND** condition)

Query filters

By **attribute value** (q)

GET <cb_host>:1026/v2/entities?**q=temperature>25**

attribute name

attribute sub-key (for compound attribute values only)

GET
<cb_host>:1026/v2/entities?**q=tirePressure.frontRight**
>130

attribute name

metadata name

metadata sub-key (for compound metadata values only)

By **m**

GET
<cb_host>:1026/v2/entities?**mq=temperature.avg>25**

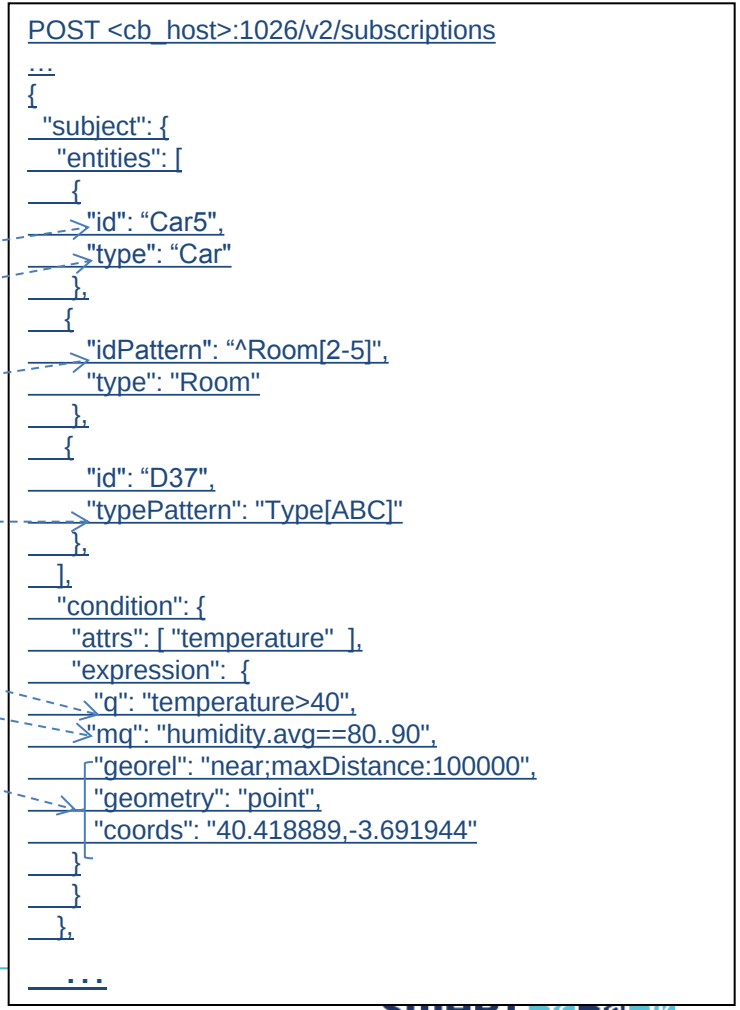
GET <cb_host>:1026/v2/entities?**mq=tirePressure.accuracy.frontRight**
>90

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

Query filters

Filters can be also used in subscriptions

- id
- type
- id pattern
- type pattern
- attribute values
- metadata value
- geographical location



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

notification

```
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

Registration & Context Providers

POST <**cb_host**>:1026/v1/registry/registerContext

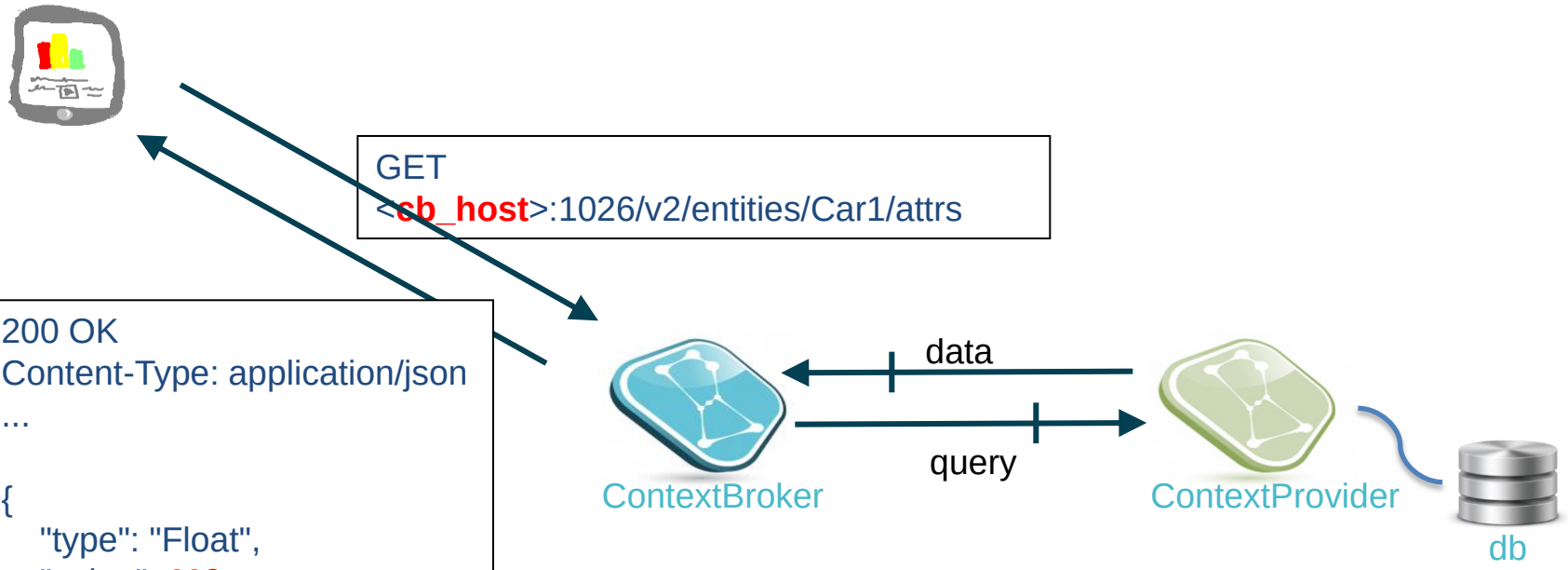
```
…
{
   "contextRegistrations": [
      {
         "entities": [
            {
               "type": "Car",
               "isPattern": "false",
               "id": "Car1"
            },
            "attributes": [
            {
               "name": "speed",
               "type": "float",
               "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

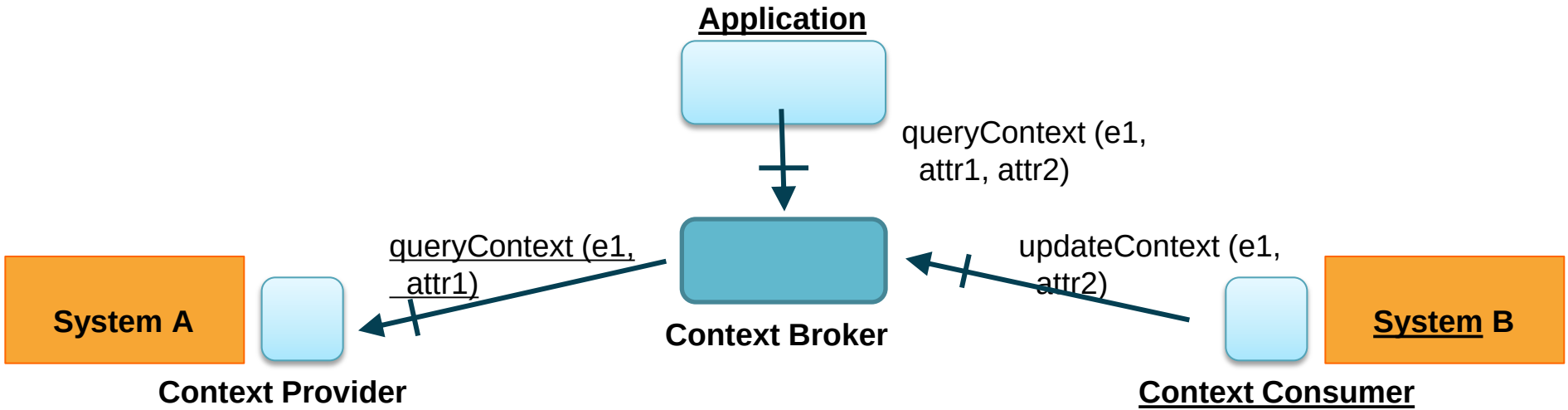


INTEGRATION WITH SENSOR NETWORKS - IOT

Integration with existing systems

Context adapters will be developed to interface with existing systems (e.g., municipal services management systems in a smart city) acting as Context Providers, Context Producers, or both

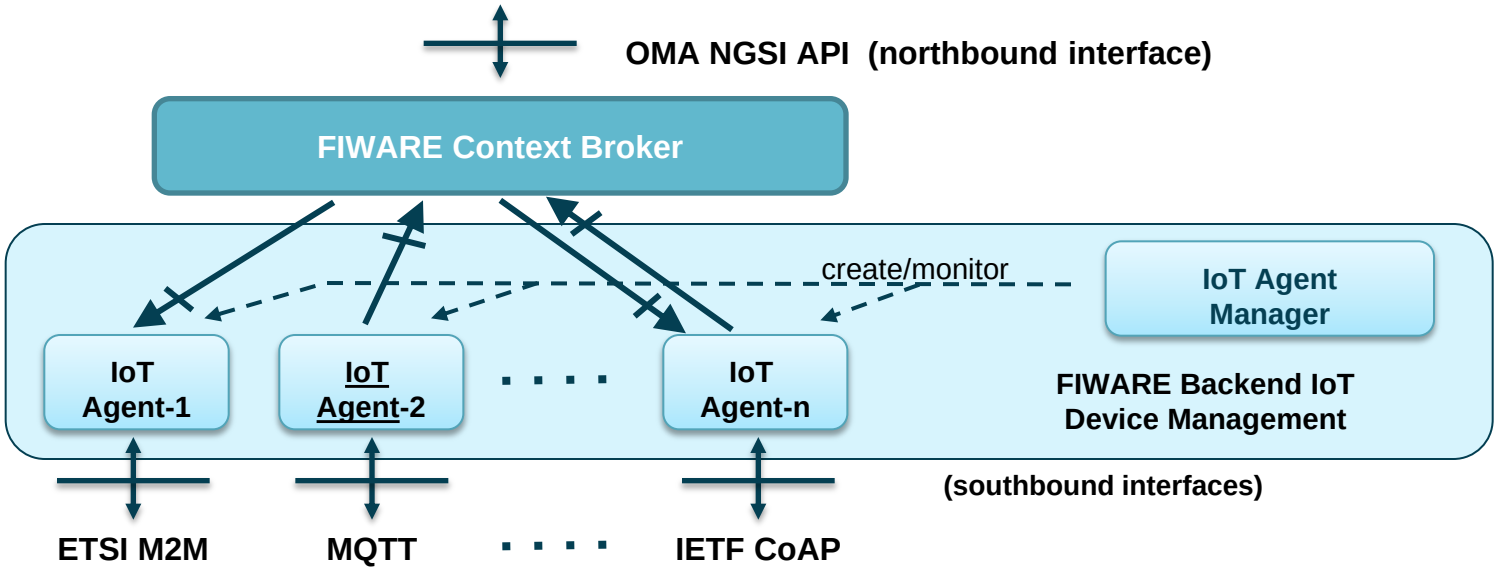
Some attributes from a given entity may be linked to a Context Provider while other attributes may be linked to Context Producers



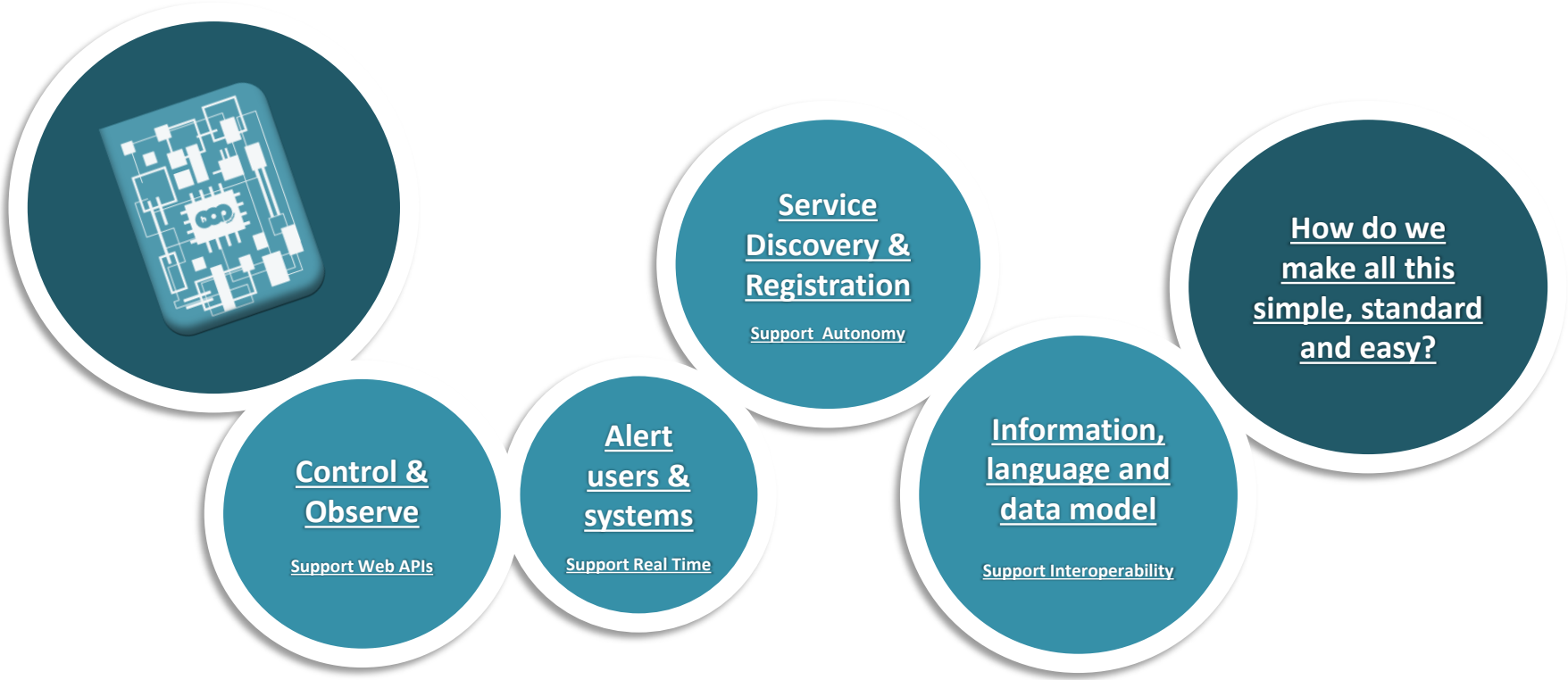
Integration with sensor networks

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



OMA LwM2M, what does it do?



IoT Protocols: OMA LwM2M

Open Mobile Alliance (OMA) was established in 2002

OMA LWM2M is the evolution of the expertise from OMA in Device Management (DM) for addressing the new requirements from constrained devices and fill the gap between 3GPP, IETF CoAP, ETSI and OMA-DM

Background

Define interfaces, protocols (SMS, CoAP) and security support between machines and the server/cloud

Define a Object and Resources Data Model (Semantic)

Leverage the expertise in Device Management with support for firmware update, connectivity, discovery, access control, bootstrapping and remote management

Goals

Bootstrapping: Pre-provisioned of security credentials (keys, tokens) , configuration of LWM2M Servers IP, etc.

Registration: Register the Client and its OMA Web Objects in the Resource Directory (Local or Cloud)

Management and Service: Operational mode to read, update, and manage objects/resources

Information Reporting: Observation of Resources for events notification

Functions/Interfaces

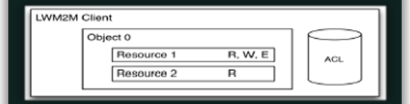
A device can have multiple Objects. An Object is a collection of resources. A resource is the atomic piece of data (e.g., temperature value, sampling frequency)

OMA and other SDOs [1] can define and register Objects

Object Data Model presents a highly efficient payload

Objects and Resources can have multiple instances. */{Obj. ID}/{Obj. Instance}/{Res. ID}/*

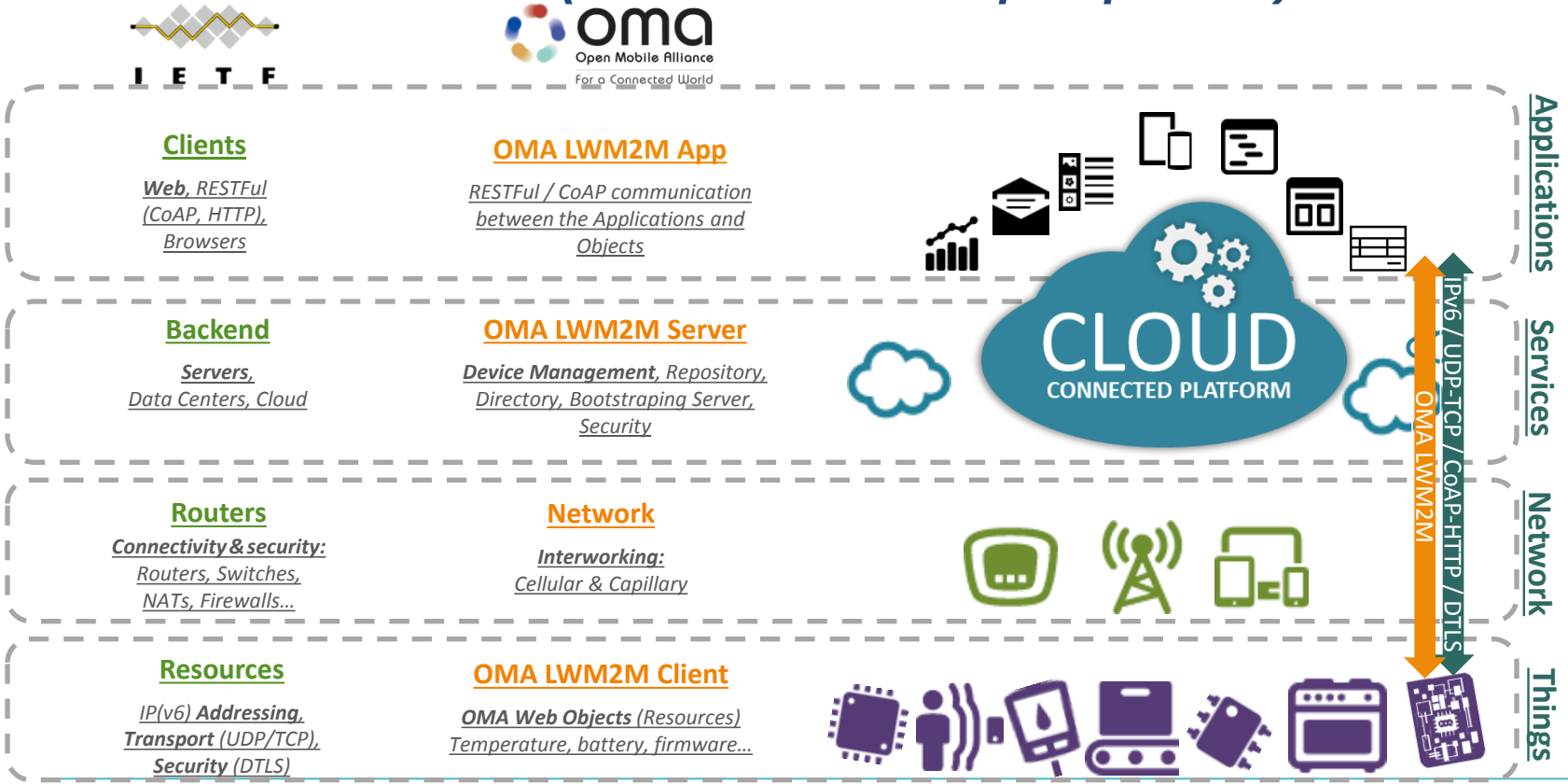
Supported meta-data for security control such as Access Control List (ACL)



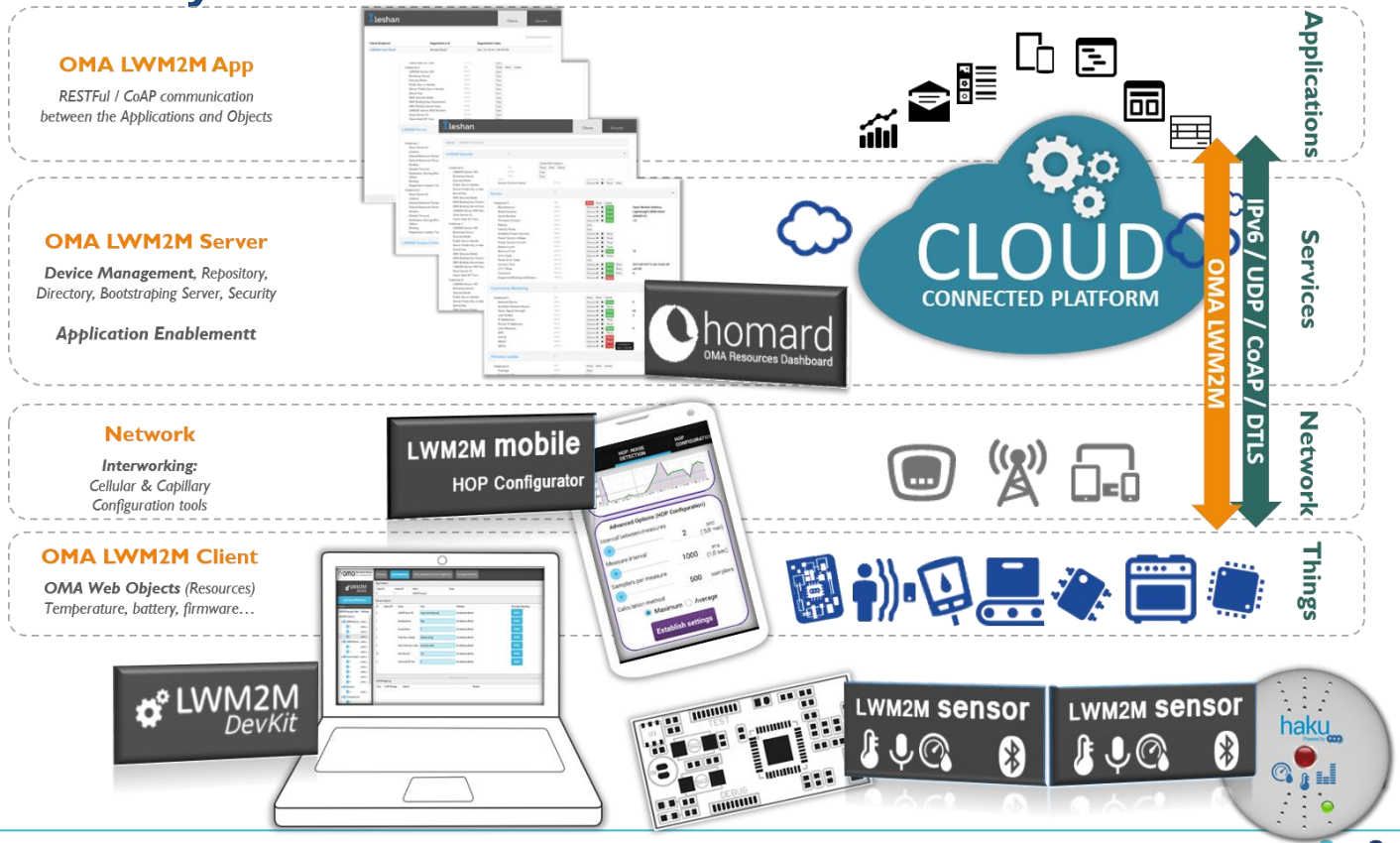
1- HOP Ubiquitous is member of IPSO Alliance & OMA Web Objects Builder.

Object Data Model

Architecture Overview *(Standards-driven perspective)*



OMA LwM2M Ecosystem



OMA LwM2M Sensor

GPIO

General Purpose Inputs and Outputs

Temperature & Humidty

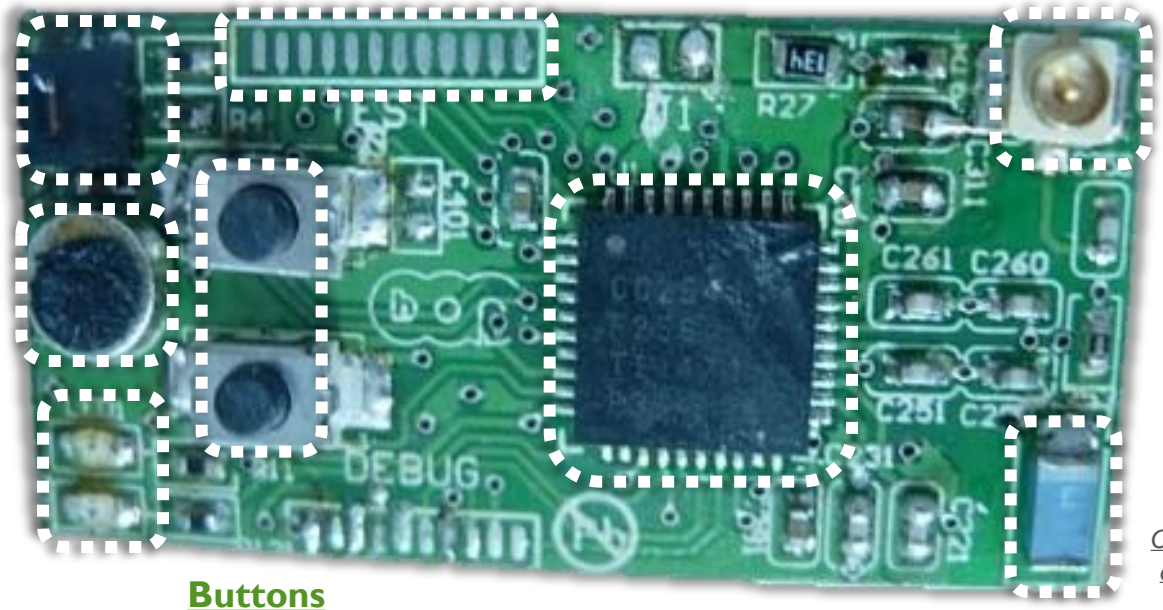
Environmental monitor

Microphone

Activity analysis based on noise

LEDs

Notifications and status indication



External Antenna

IPEX Connector for external Antenna for infrastructure deployments (10 – 90 meters coverage)

Antenna

Ceramic Antenna for personal area and proximity solutions (1 – 10 meters coverage)

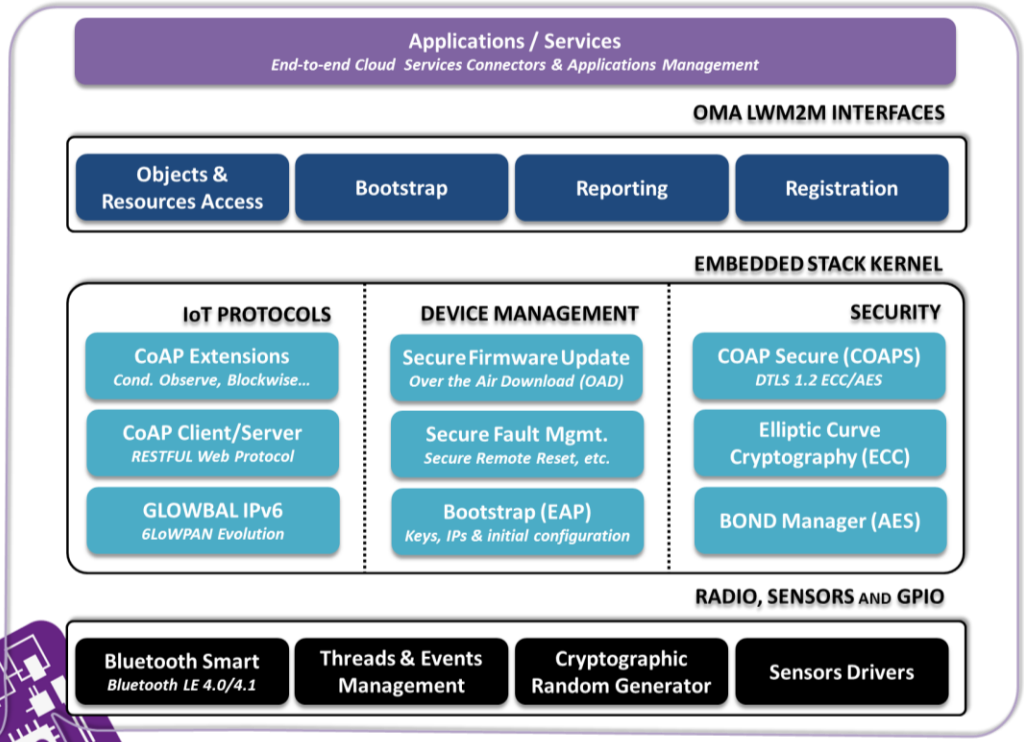
Buttons

Interaction with the User

Bluetooth Smart

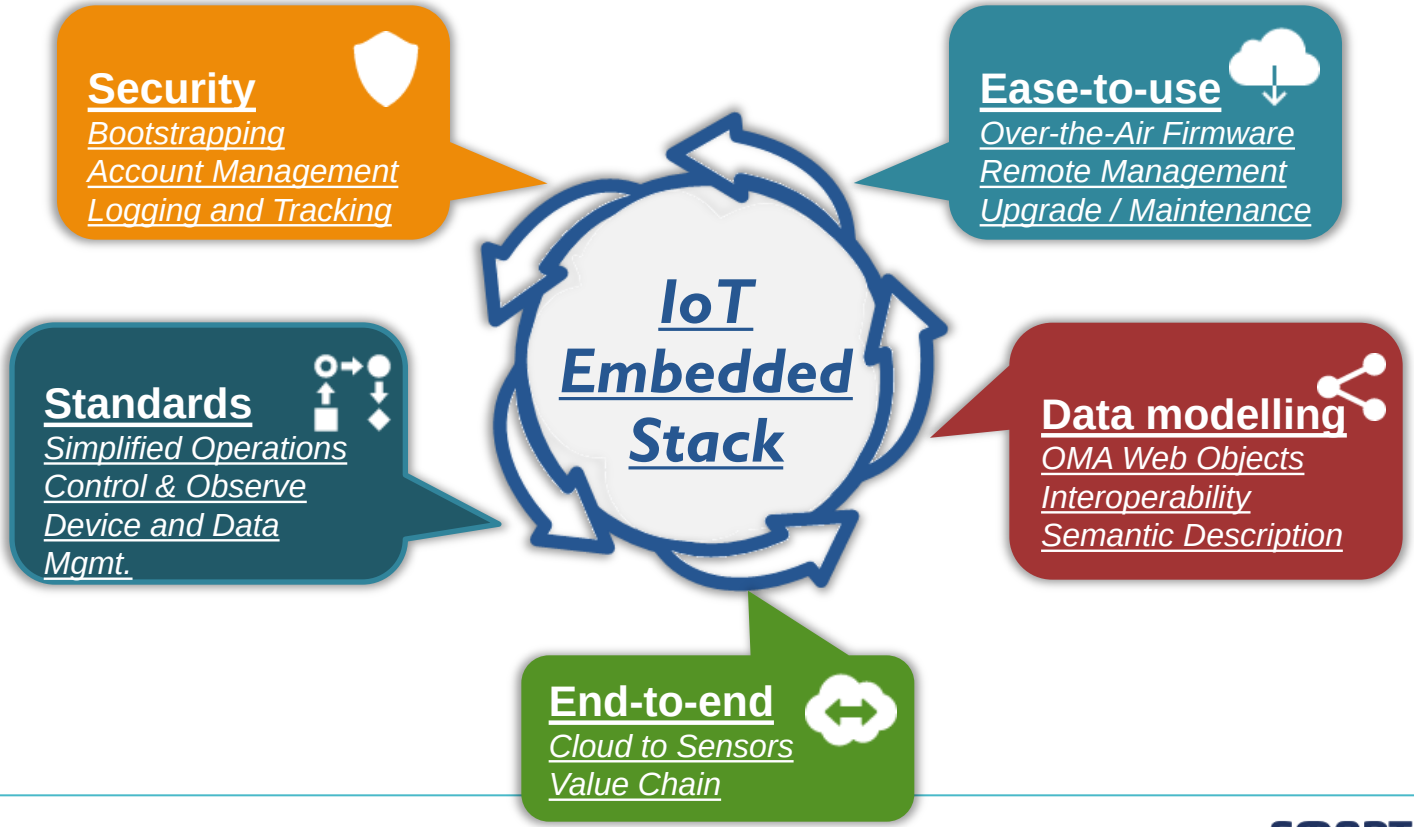
System on Chip Bluetooth Smart and OMA LWM2M / CoAP / IPv6 transceiver

IoT Embedded Stack

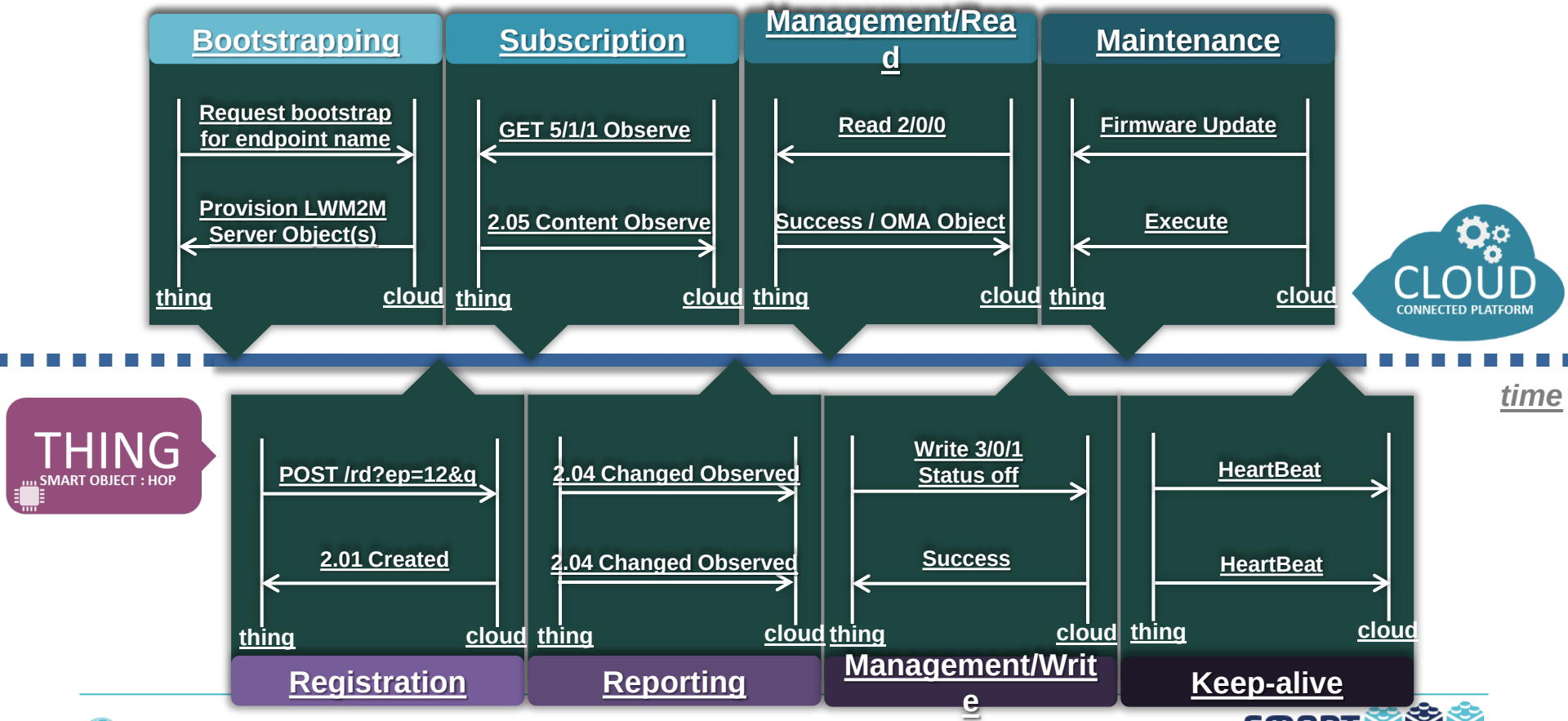


1. IPv6-oriented *(CoAP/UDP/DTLS)*
2. OMA LWM2M by design
3. Security by design
4. Beyond Bluetooth Smart (IEEE 802.15.4g, WiFi Low Power)
5. HOP Configurator *(Mobile Tool)*
6. Industry support *(Real Time, Pub-Sub, Management)*

OMA LwM2M Functions



OMA LwM2M Examples of Communication



EXAMPLE / DEMO

Tests online

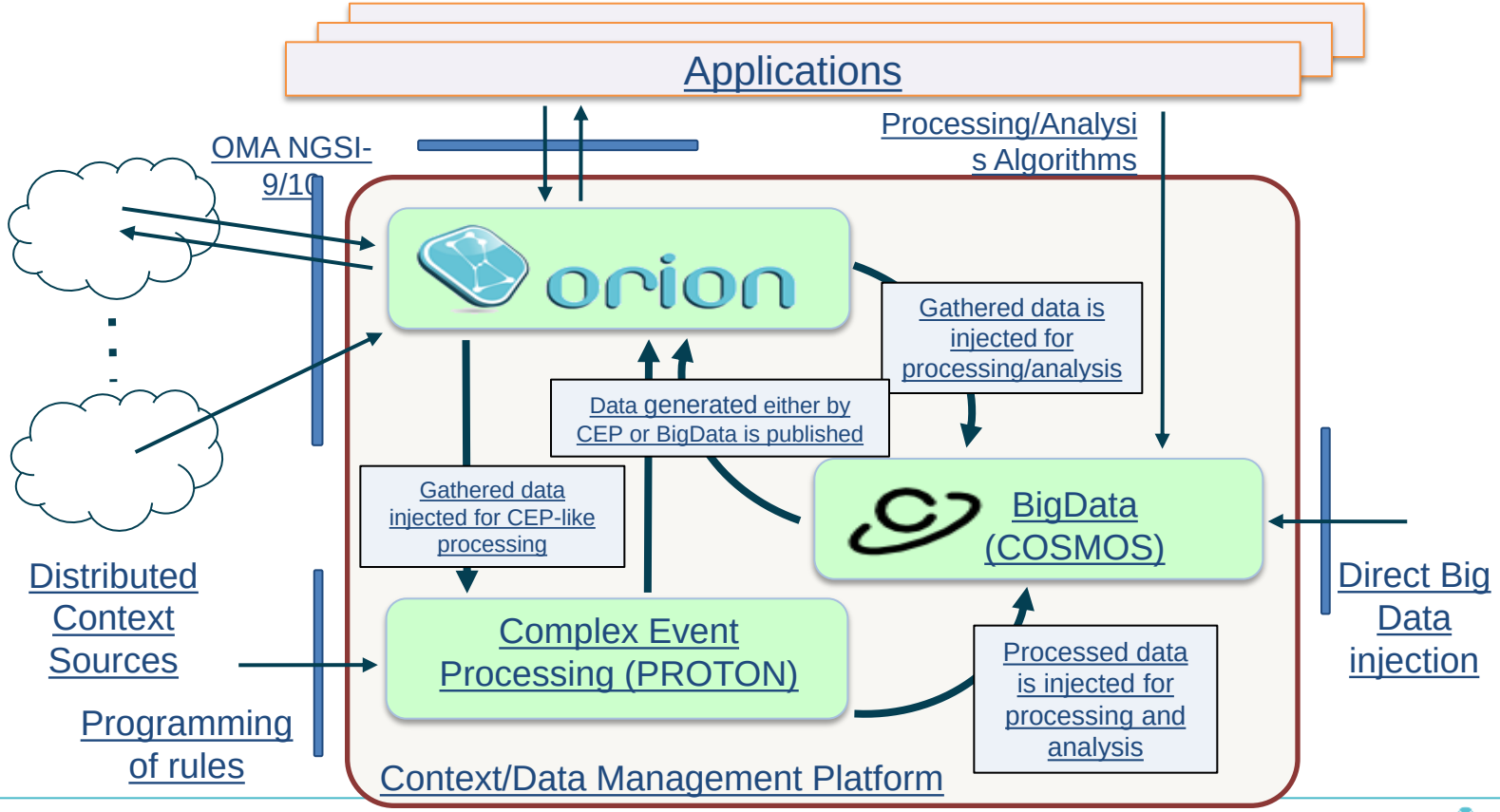
1- Download postman https://www.getpostman.com/

2- Import a collection of scripts to our server:
http://orion.hopu.eu:1026/version

3- Get Collection:
https://www.getpostman.com/collections/1edf18d66df5d547f655
- ORION v2 Get Entity Data Model HOP240ac403f392
- ORION v1 Write Entity Attr (availability) HOP240ac403f392

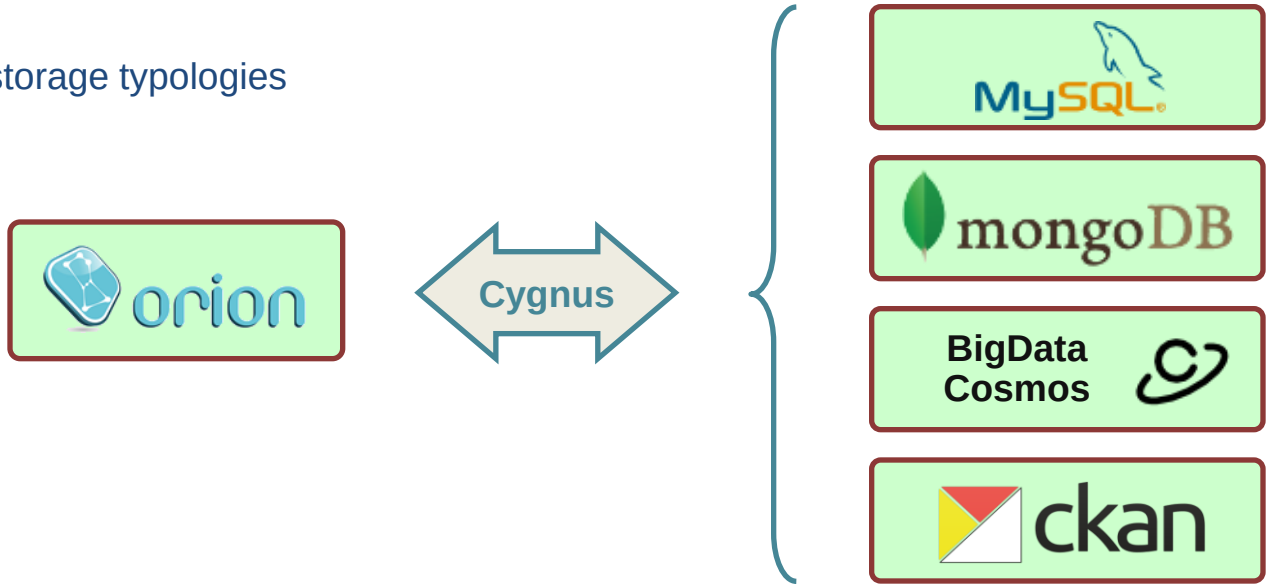
INTEGRATION WITH OTHER SYSTEMS

FI-WARE Context/Data Management Platform



How to store the context information

- Context Broker don’t persist historic data
- Cygnus: adapter between Context Broker and Storage
- There are 4 storage typologies



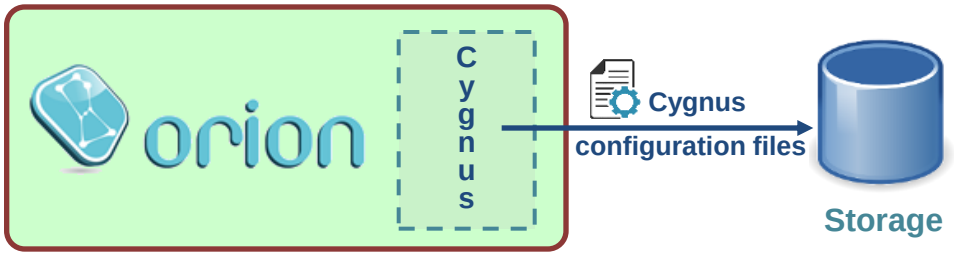
The 4 Storage Systems

- MySQL is **related** database
- mongoDB is **not related** database (NoSQL)
- Cosmos is **Hadoop** ecosystem-based to manage huge amounts of previously stored data
  - **HDFS** as its distributed file system
  - **Hadoop core** as its **MapReduce** engine
  - **HiveQL** and **Pig** for querying the data
  - **Oozie** as remote MapReduce jobs and Hive launcher

- CKAN (*Comprehensive Knowledge Archive Network*) is (Open) Data Platform
  - create private/public **datasets** (JSON, CSV, TXT, PDF, …)
  - REST/JSON APIs to access **data** and **metadata**

Install Cygnus on Context Broker VM

- Deploy Context Broker
- Install Cygnus (on Context Broker VM):
  - `$ sudo -i`
  - `# yum install cygnus`
- Cygnus configuration files (next sections)
- Status/Start/Restart/Stop command
  - `# service cygnus status`
  - `# service cygnus start`
  - `# service cygnus restart`
  - `# service cygnus stop`



Cygnus configuration files

- All files are available (as template) in /usr/cygnus/conf/ folder (in detail see **README.md** file):
  - **log4j.properties** `cp /usr/cygnus/conf/log4j.properties.template /usr/cygnus/conf/log4j.properties`
  - **flume-env.sh** `cp /usr/cygnus/conf/flume-env.sh.template /usr/cygnus/conf/flume-env.sh`
  - **krb5.conf** `cp /usr/cygnus/conf/krb5.conf.template /usr/cygnus/conf/krb5.conf`
  - **matching_table.conf** `cp /usr/cygnus/conf/matching_table.conf.template /usr/cygnus/conf/matching_table.conf`
- Files per instance:
  - **agent.conf**
  - **cygnus_instance.conf**
- Replace "**<id>**" tag with your instance name:
  - **agent_<id>.conf**
  - **cygnus_instance_<id>.conf**

1. Context Broker and MySQL

- You need an instance of MySQL Database

- Tips (to install database on Context Broker instance)

  - `# yum install mysql mysql-server mysql-libs mysql-server`
  - `# service mysqld start`
  - `# mysqladmin -u root password NEWPASSWORD`

- Edit instance files and rename them with **<id>** = **mysql**

  - **cygnus_instance_mysql.conf**
  - **agent_mysql.conf**

1. Context Broker and MySQL

- Edit cygnus_instance_mysql.conf

```
# Who to run cygnus as. Note that you may need to use root if you want
# to run cygnus in a privileged port (<1024)
CYGNUS_USER=cygnus

# Where is the config folder
CONFIG_FOLDER=/usr/cygnus/conf

# Which is the config file
CONFIG_FILE=/usr/cygnus/conf/agent_mysql.conf

# Name of the agent. The name of the agent is not trivial, since it is the base for the Flume parameters
# naming conventions, e.g. it appears in .sources.http-source.channels=...
AGENT_NAME=cygnusagent

# Name of the logfile located at /var/log/cygnus. It is important to put the extension '.log' in order to the log rotation works properly
LOGFILE_NAME=cygnus.log

# Administration port. Must be unique per instance
ADMIN_PORT=8081

# Polling interval (seconds) for the configuration reloading
POLLING_INTERVAL=30
```

1. Context Broker and MySQL

- Edit agent_mysql.conf

```
# multi-threading).
cygnusagent.sources = http-source
cygnusagent.sinks = mysql-sink
cygnusagent.channels = mysql-channel

#=============================================
# source configuration
# channel name where to write the notification events
cygnusagent.sources.http-source.channels = mysql-channel
# source class, must not be changed
cygnusagent.sources.http-source.type = org.apache.flume.source.http.HTTPSource
# listening port the Flume source will use for receiving incoming notifications
cygnusagent.sources.http-source.port = 5050
# Flume handler that will parse the notifications, must not be changed
cygnusagent.sources.http-source.handler = com.telefonica.iot.cygnus.handlers.OrionRestHandler
# URL target
cygnusagent.sources.http-source.handler.notification_target = /notify
# Default service (service semantic depends on the persistence sink)
cygnusagent.sources.http-source.handler.default_service = def_serv
# Default service path (service path semantic depends on the persistence sink)
cygnusagent.sources.http-source.handler.default_service_path = def_servpath
# Number of channel re-injection retries before a Flume event is definitely discarded (-1 means infinite retries)
```

1. Context Broker and MySQL

- Edit agent_mysql.conf

```
# ================================================
# OrionMySQLSink configuration
# channel name from where to read notification events
cygnusagent.sinks.mysql-sink.channel = mysql-channel
# sink class, must not be changed
cygnusagent.sinks.mysql-sink.type = com.telefonica.iot.cygnus.sinks.OrionMySQLSink
#Â true if the grouping feature is enabled for this sink, false otherwise
cygnusagent.sinks.mysql-sink.enable_grouping = false
# the FQDN/IP address where the MySQL server runs
cygnusagent.sinks.mysql-sink.mysql_host = localhost
# the port where the MySQL server listes for incomming connections
cygnusagent.sinks.mysql-sink.mysql_port = 3306
# a valid user in the MySQL server
cygnusagent.sinks.mysql-sink.mysql_username = root
# password for the user above
cygnusagent.sinks.mysql-sink.mysql_password = root
# how the attributes are stored, either per row either per column (row, column)
cygnusagent.sinks.mysql-sink.attr_persistence = row
# select the table type from table-by-destination and table-by-service-path
cygnusagent.sinks.mysql-sink.table_type = table-by-destination
#Â number of notifications to be included within a processing batch
cygnusagent.sinks.mysql-sink.batch_size = 100
```

Example of publish/subscribe

updateContext :

```
{
    "contextElements": [
        {
            "type": "Car",
            "isPattern": "false",
            "id": "Car1",
            "attributes": [
            {
                "name": "speed",
                "type": "float",
                "value": "98"
            }
            ]
        }
    ],
    "updateAction": "APPEND"
}
```

queryContext :

```
{
    "entities": [
        {
            "type": "Car",
            "isPattern": "false",
            "id": "Car1"
        }
     ]
}
```

subscribeContext:

```
{
    "entities": [
        {
            "type": "Car",
            "isPattern": "false",
            "id": "Car1"
        }
    ],
    "attributes": [
        "speed"
    ],
    "reference": "http://localhost:5050/notify",
    "duration": "P1M",
    "notifyConditions": [
        {
            "type": "ONCHANGE",
            "condValues": [
                "speed"
            ]
        }
    ],
    "throttling": "PT1S"
}
```

Headers:

Content-Type: **application/json**
Accept: application/json
Fiware-Service: **vehicles**
Fiware-ServicePath: **/4wheels**

1. Context Broker and MySQL - Example: check the context data

```
$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql> show databases;
+----------------------+
| Database             |
+----------------------+
| information_schema   |
| mysql                |
| test                 |
| vehicles             |
+----------------------+
4 rows in set (0.01 sec)
mysql> use vehicles;
...
Database changed
mysql> show tables;
+--------------------+
| Tables_in_vehicles |
+--------------------+
| 4wheels_car1_car   |
+--------------------+
1 row in set (0.00 sec)
mysql> select * from 4wheels_car1_car;
...
```

2. Context Broker and mongoDB

- You need an instance of mongoDB Database
- Tips (Context Broker instance has already got it)
- Edit instance files and rename them with **<id>** = **mongo**
  - **cygnus_instance_mongo.conf**
  - **agent_mongo.conf**

2. Context Broker and mongoDB

- Edit cygnus_instance_mongo.conf

```
# Who to run cygnus as. Note that you may need to use root if you want
# to run cygnus in a privileged port (<1024)
CYGNUS_USER=cygnus

# Where is the config folder
CONFIG_FOLDER=/usr/cygnus/conf

# Which is the config file
CONFIG_FILE=/usr/cygnus/conf/agent_mongo.conf

# Name of the agent. The name of the agent is not trivial, since it is the base for the Flume parameters
# naming conventions, e.g. it appears in .sources.http-source.channels=...
AGENT_NAME=cygnusagent

# Name of the logfile located at /var/log/cygnus. It is important to put the extension '.log' in order to the log rotation works properly
LOGFILE_NAME=cygnus.log

# Administration port. Must be unique per instance
ADMIN_PORT=8081

# Polling interval (seconds) for the configuration reloading
POLLING_INTERVAL=30
```

2. Context Broker and mongoDB

- Edit agent_mongo.conf

```
cygnusagent.sources = http-source
cygnusagent.sinks = mongo-sink
cygnusagent.channels = mongo-channel

#=============================================
# source configuration
# channel name where to write the notification events
cygnusagent.sources.http-source.channels = mongo-channel
# source class, must not be changed
cygnusagent.sources.http-source.type = org.apache.flume.source.http.HTTPSource
# listening port the Flume source will use for receiving incoming notifications
cygnusagent.sources.http-source.port = 5050
# Flume handler that will parse the notifications, must not be changed
cygnusagent.sources.http-source.handler = com.telefonica.iot.cygnus.handlers.OrionRestHandler
# URL target
cygnusagent.sources.http-source.handler.notification_target = /notify
# Default service (service semantic depends on the persistence sink)
cygnusagent.sources.http-source.handler.default_service = def_serv
# Default service path (service path semantic depends on the persistence sink)
cygnusagent.sources.http-source.handler.default_service_path = def_servpath
# Number of channel re-injection retries before a Flume event is definitely discarded (-1 means infinite retries)
cygnusagent.sources.http-source.handler.events_ttl = 10
```

2. Context Broker and mongoDB

- Edit agent_mongo.conf

```
# sink class, must not be changed
cygnusagent.sinks.mongo-sink.type = com.telefonica.iot.cygnus.sinks.OrionMongoSink
# channel name from where to read notification events
cygnusagent.sinks.mongo-sink.channel = mongo-channel
#Â true if the grouping feature is enabled for this sink, false otherwise
cygnusagent.sinks.mongo-sink.enable_grouping = false
# FQDN/IP:port where the MongoDB server runs (standalone case) or comma-separated list of FQDN/IP:
cygnusagent.sinks.mongo-sink.mongo_hosts = localhost:27017
# a valid user in the MongoDB server (or empty if authentication is not enabled in MongoDB)
cygnusagent.sinks.mongo-sink.mongo_username =
# password for the user above (or empty if authentication is not enabled in MongoDB)
cygnusagent.sinks.mongo-sink.mongo_password =
# prefix for the MongoDB databases
cygnusagent.sinks.mongo-sink.db_prefix = sth_
# prefix pro the MongoDB collections
cygnusagent.sinks.mongo-sink.collection_prefix = sth_
# true is collection names are based on a hash, false for human redable collections
cygnusagent.sinks.mongo-sink.should_hash = false
# specify if the sink will use a single collection for each service path, for each entity or for each attribute
cygnusagent.sinks.mongo-sink.data_model = dm-by-entity
# how the attributes are stored, either per row either per column (row, column)
cygnusagent.sinks.mongo-sink.attr_persistence = row
```

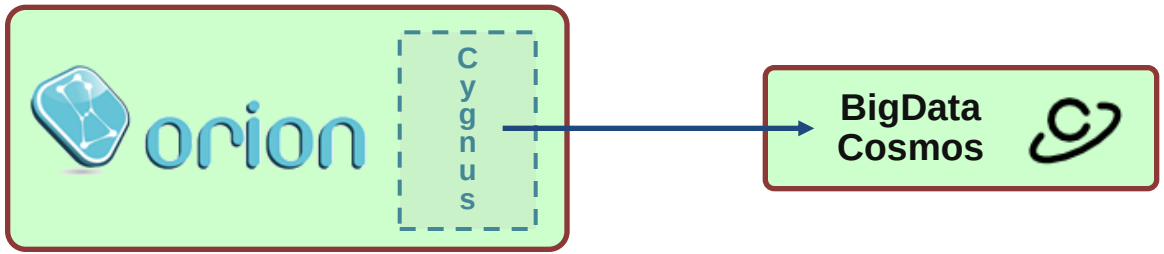
empty fields

2. Context Broker and mongoDB - Example: check the context data

```
$ mongo
MongoDB shell version: 2.6.9
connecting to: test
...
> show databases
admin           (empty)
local           0.031GB
orion           0.031GB
orion-vehicles  0.031GB
sth_vehicles    0.031GB
test            (empty)
> use sth_vehicles
switched to db sth_vehicles
> show collections
sth_/4wheels_car1_car
system.indexes
> db['sth_/4wheels_car1_car'].find()
...
```

3. Context Broker and Cosmos

- You need an instance of Big Data Analysis Cosmos GE
- Tips (FIWARE Lab provide you this feature at this link **cosmos.lab.fiware.org**)
- Edit instance files and rename them with **<id>** = **cosmos**
  - **cygnus_instance_cosmos.conf**
  - **agent_cosmos.conf**



3. Context Broker and Cosmos

- Edit cygnus_instance_cosmos.conf

```
# Where is the config folder
CONFIG_FOLDER=/usr/cygnus/conf

# Which is the config file
CONFIG_FILE=/usr/cygnus/conf/agent_cosmos.conf

# Name of the agent. The name of the agent is not trivial, since it is the base for the Flume parameters
# naming conventions, e.g. it appears in .sources.http-source.channels=...
AGENT_NAME=cygnusagent

# Name of the logfile located at /var/log/cygnus. It is important to put the extension '.log' in order to the log rotation works properly
LOGFILE_NAME=cygnus.log

# Administration port. Must be unique per instance
ADMIN_PORT=8081

# Polling interval (seconds) for the configuration reloading
POLLING_INTERVAL=30
```

3. Context Broker and Cosmos

- Edit agent_cosmos.conf

```
cygnusagent.sources = http-source
cygnusagent.sinks = hdfs-sink
cygnusagent.channels = hdfs-channel

#=============================================
# source configuration
# channel name where to write the notification events
cygnusagent.sources.http-source.channels = hdfs-channel
# source class, must not be changed
cygnusagent.sources.http-source.type = org.apache.flume.source.http.HTTPSource
# listening port the Flume source will use for receiving incoming notifications
cygnusagent.sources.http-source.port = 5050
# Flume handler that will parse the notifications, must not be changed
cygnusagent.sources.http-source.handler = com.telefonica.iot.cygnus.handlers.OrionRestHandler
# URL target
cygnusagent.sources.http-source.handler.notification_target = /notify
# Default service (service semantic depends on the persistence sink)
cygnusagent.sources.http-source.handler.default_service = def_serv
# Default service path (service path semantic depends on the persistence sink)
cygnusagent.sources.http-source.handler.default_service_path = def_servpath
# Number of channel re-injection retries before a Flume event is definitely discarded (-1 means infinite r
```

3. Context Broker and Cosmos

- Edit agent_cosmos.conf

```
# If you are using Kerberos authentication, then the usage of FQDNs instead of IP addresses is mandatory
cygnusagent.sinks.hdfs-sink.hdfs_host = cosmos.lab.fiware.org
# port of the HDFS service listening for persistence operations; 14000 for httpfs, 50070 for webhdfs
cygnusagent.sinks.hdfs-sink.hdfs_port = 14000
# username allowed to write in HDFS
cygnusagent.sinks.hdfs-sink.hdfs_username = pasquale.vitale
#Â password for the above username; this is only required for Hive authentication
cygnusagent.sinks.hdfs-sink.hdfs_password = xxxxxxxx
# OAuth2 token for HDFS authentication
cygnusagent.sinks.hdfs-sink.oauth2_token = P8vBzFdJB2ZBNHwfxUHFrfc1buTx7n
# how the attributes are stored, available formats are json-row, json-column, csv-row and csv-column
cygnusagent.sinks.hdfs-sink.file_format = json-column
#Â number of notifications to be included within a processing batch
cygnusagent.sinks.hdfs-sink.batch_size = 100
# timeout for batch accumulation
cygunsagent.sinks.hdfs-sink.batch_timeout = 30
# Hive enabling
cygnusagent.sinks.hdfs-sink.hive = true
# Hive server version, 1 or 2 (ignored if hive is false)
cygnusagent.sinks.hdfs-sink.hive.server_version = 2
# Hive FQDN/IP address of the Hive server (ignored if hive is false)
cygnusagent.sinks.hdfs-sink.hive.host = cosmos.lab.fiware.org
# Hive port for Hive external table provisioning (ignored if hive is false)
```

3. Context Broker and Cosmos - How to get the auth-token

- Cosmos WebHDFS access is protected with OAuth2
- Request:

```
curl -X POST "https://cosmos.lab.fiware.org:13000/cosmos-auth/v1/token" -H "Content-Type: application/x-www-form-urlencoded" -d "grant_type=password&username=pasquale.vitale@eng.it&password=mypassword" -k
```

- Response:

```
{"access_token": "P8vBzFdJB2ZBNHwfxUHFrfc1buTx7n", "token_type": "Bearer", "expires_in": 3600, "refresh_token": "He8aJdQiEkpbnQB4KFAS1DFra9RhNq"}
```

- Example to get file status:

```
curl -X GET "http://cosmos.lab.fiware.org:14000/webhdfs/v1/user/pasquale.vitale?op=liststatus&user.name=pasquale.vitale" -H "X-Auth-Token: P8vBzFdJB2ZBNHwfxUHFrfc1buTx7n"
```

```
{"FileStatuses":{"FileStatus":[...]}}
```

3. Context Broker and Cosmos - Example: check the context data

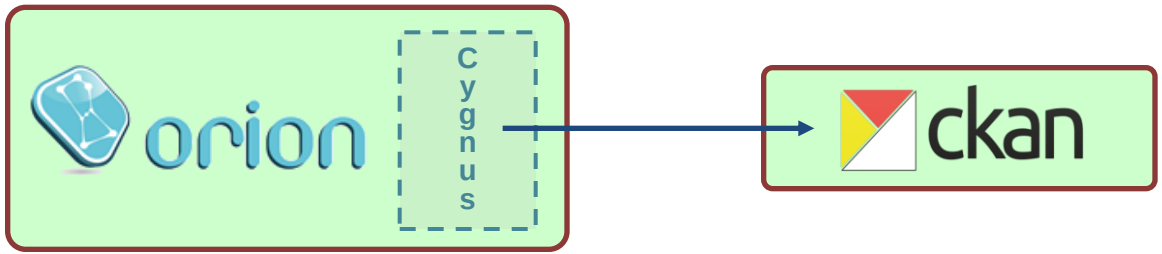
- Cosmos GUI - http://cosmos.lab.fi-ware.org/cosmos-gui

- Connection and check data in Cosmos:

```
$ ssh pasquale.vitale@cosmos.lab.fi-ware.org
pasquale.vitale@cosmos.lab.fiware.org's password:

$ hadoop fs -lsr /user/pasquale.vitale/
drwxr-----   - pasquale.vitale pasquale.vitale    0 2015-07-21 17:11 /user/pasquale.vitale/vehicles
drwxr-----   - pasquale.vitale pasquale.vitale    0 2015-07-21 17:11 /user/pasquale.vitale/vehicles/4wheels
drwxr-----   - pasquale.vitale pasquale.vitale    0 2015-09-24 15:54 /user/pasquale.vitale/vehicles/4wheels/car1_car
-rw-r-----   3 pasquale.vitale pasquale.vitale  820 2015-09-24 15:54 /user/pasquale.vitale/vehicles/4wheels/car1_car/car1_car

$ hadoop fs -cat /user/pasquale.vitale/vehicles/4wheels/car1_car/car1_car.txt
{"recvTime":"2015-07-21T15:11:11.344Z","speed":"25", "speed_md":[]}
{"recvTime":"2015-07-21T15:41:30.233Z","speed":"125", "speed_md":[]}
{"recvTime":"2015-07-21T15:45:52.305Z","speed":"99", "speed_md":[]}
```

4. Context Broker and CKAN

- You need an instance of CKAN
- Tips (FIWARE Lab provide you this feature at this link **data.lab.fiware.org**)
- Edit instance files and rename them with **<id>** = **ckan**
  - **cygnus_instance_ckan.conf**
  - **agent_ckan.conf**



4. Context Broker and CKAN

- Edit cygnus_instance_ckan.conf

```
# Who to run cygnus as. Note that you may need to use root if you want
# to run cygnus in a privileged port (<1024)
CYGNUS_USER=cygnus

# Where is the config folder
CONFIG_FOLDER=/usr/cygnus/conf

# Which is the config file
CONFIG_FILE=/usr/cygnus/conf/agent_ckan.conf

# Name of the agent. The name of the agent is not trivial, since it is the base for the Flume parameters
# naming conventions, e.g. it appears in .sources.http-source.channels=...
AGENT_NAME=cygnusagent

# Name of the logfile located at /var/log/cygnus. It is important to put the extension '.log' in order to the log rotation works properly
LOGFILE_NAME=cygnus.log

# Administration port. Must be unique per instance
ADMIN_PORT=8081

# Polling interval (seconds) for the configuration reloading
POLLING_INTERVAL=30
```

4. Context Broker and CKAN

- Edit agent_ckan.conf

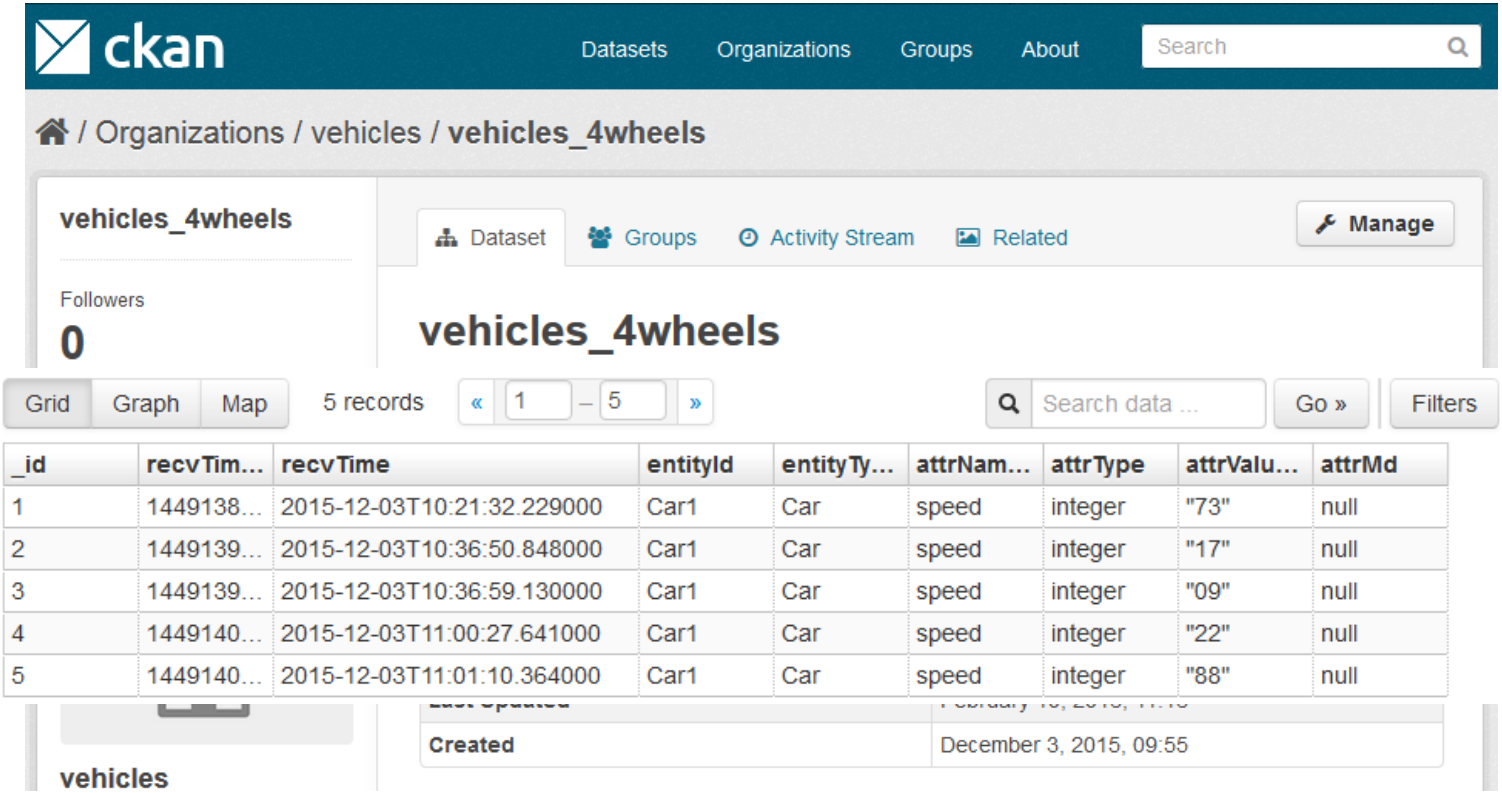
```
cygnusagent.sources = http-source
cygnusagent.sinks = ckan-sink
cygnusagent.channels = ckan-channel

#=============================================
# source configuration
# channel name where to write the notification events
cygnusagent.sources.http-source.channels = ckan-channel
# source class, must not be changed
cygnusagent.sources.http-source.type = org.apache.flume.source.http.HTTPSource
# listening port the Flume source will use for receiving incoming notifications
cygnusagent.sources.http-source.port = 5050
# Flume handler that will parse the notifications, must not be changed
cygnusagent.sources.http-source.handler = com.telefonica.iot.cygnus.handlers.OrionRestHandler
# URL target
cygnusagent.sources.http-source.handler.notification_target = /notify
# Default service (service semantic depends on the persistence sink)
cygnusagent.sources.http-source.handler.default_service = def_serv
# Default service path (service path semantic depends on the persistence sink)
cygnusagent.sources.http-source.handler.default_service_path = def_servpath
# Number of channel re-injection retries before a Flume event is definitely discarded (-1 means infinite retries)
cygnusagent.sources.http-source.handler.events_ttl = 10
```

4. Context Broker and CKAN

- Edit agent_ckan.conf

```
# ================================================
# OrionCKANSink configuration
# channel name from where to read notification events
cygnusagent.sinks.ckan-sink.channel = ckan-channel
# sink class, must not be changed
cygnusagent.sinks.ckan-sink.type = com.telefonica.iot.cygnus.sinks.OrionCKANSink
#Â true if the grouping feature is enabled for this sink, false otherwise
cygnusagent.sinks.ckan-sink.enable_grouping = false
# the CKAN API key to use
cygnusagent.sinks.ckan-sink.api_key = 316d8266-a743-495b-b369-cde8d65630ee
# the FQDN/IP address for the CKAN API endpoint
cygnusagent.sinks.ckan-sink.ckan_host =data.lab.fiware.org
# the port for the CKAN API endpoint
cygnusagent.sinks.ckan-sink.ckan_port = 443
# Orion URL used to compose the resource URL with the convenience operation URL to query
cygnusagent.sinks.ckan-sink.orion_url = http://localhost:1026
# how the attributes are stored, either per row either per column (row, column)
cygnusagent.sinks.ckan-sink.attr_persistence = row
# enable SSL for secure Http transportation; 'true' or 'false'
cygnusagent.sinks.ckan-sink.ssl = true
# number of notifications to be included within a processing batch
cygnusagent.sinks.ckan-sink.batch_size = 100
```

4. Context Broker and CKAN - Example: check the context data



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

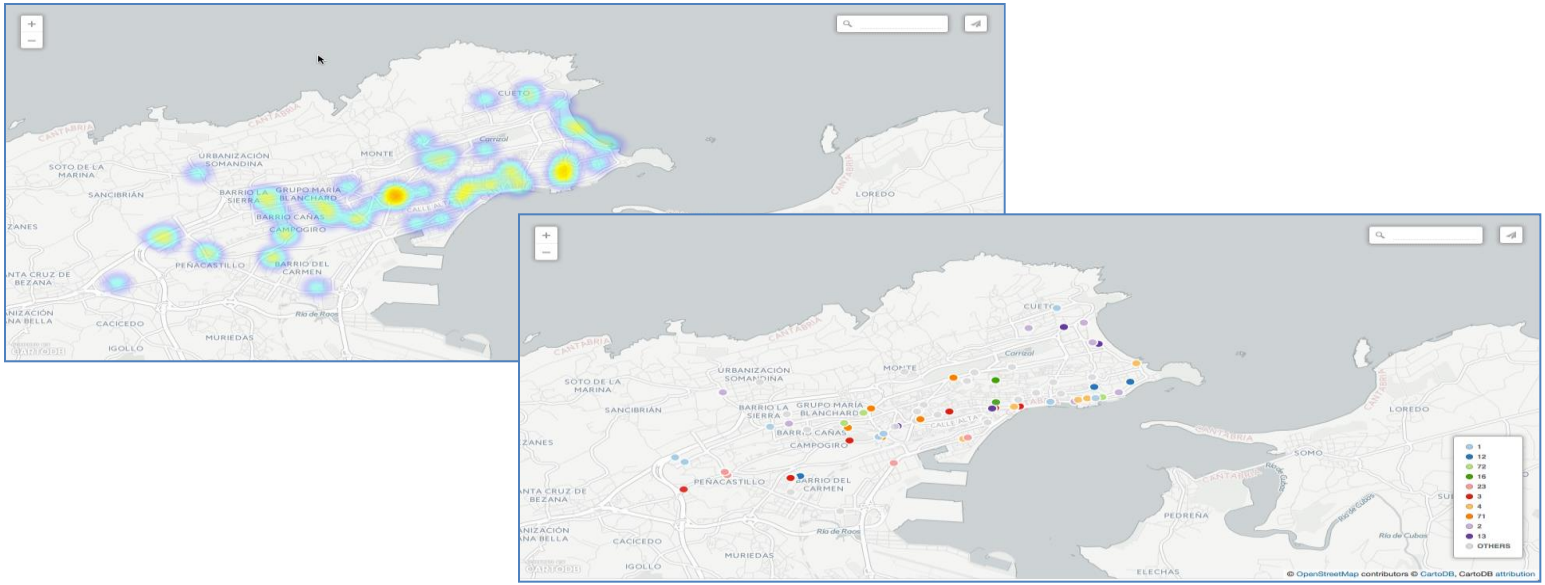
Open a Postman session and rock and roll

Postman collection: https://github.com/Fiware/tutorials.TourGuide-App/blob/develop/contrib/CampusParty2016.postman_collection

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

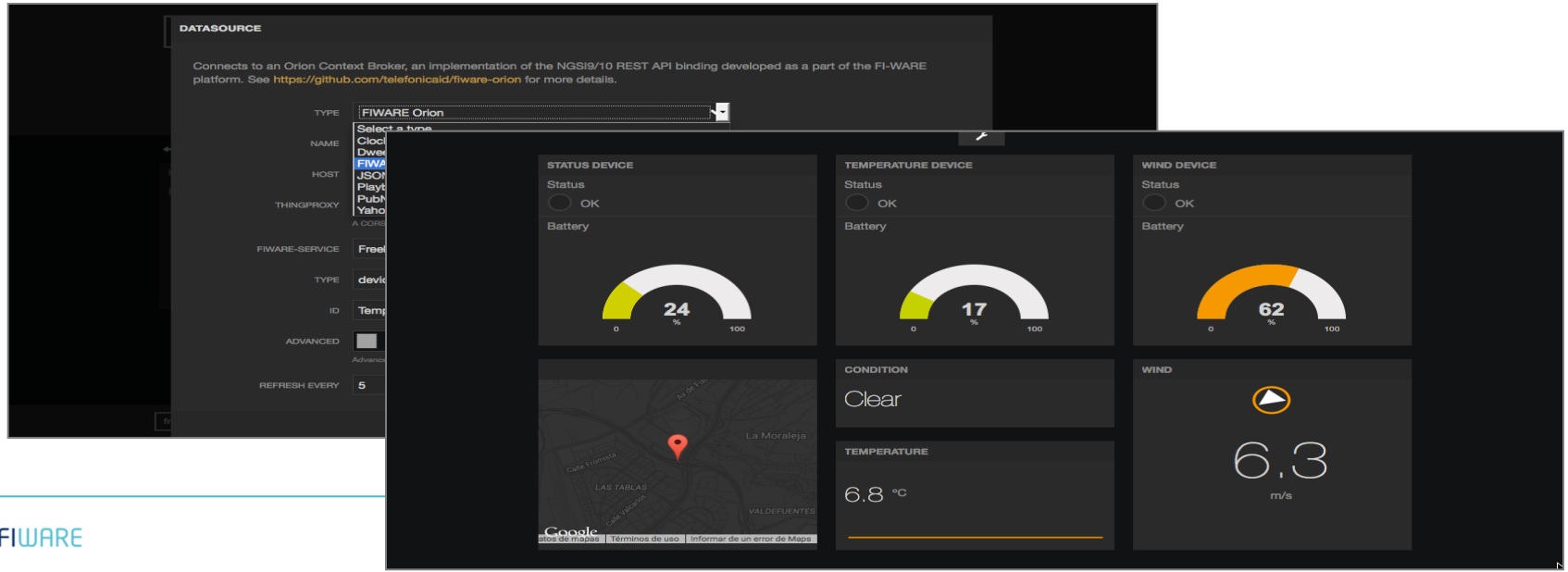


NGSI Plugin for Freeboard

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



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

TEST IT AT HOME

Setting up the environment

1- Download the Virtual Machine with Orion Context Broker + UbiBox Driver (Gateway for IoT sensors) + Examples
https://storage.googleapis.com/fiware/CentOS6%20-%20Orion%20-%20Master%201.rar

2- Install VirtualBox - https://www.virtualbox.org/

Run the environment

1- Open MongoDB instance in a terminal

./mongoDB/bin/mongod

2- Open Orion Context Broker instance

contextBroker -fg -logLevel DEBUG -httpTimeout 30000

3- Open Lightweight IOTAgent in a terminal

./iotagent/lightweightm2m-iotagent/bin/lwm2mAgent.js

Test and play

1- Read

iotagent/scripts_SMR/1.QueryAttributes.Orion

2- Write

iotagent/scripts_SMR/2.1.WriteTwoResources.Orion
iotagent/scripts_SMR/2.2.WriteOpaque.Orion

3- Execute

iotagent/scripts_SMR/3.ExecuteCommand.Orion

4- Test all the queries that you have seen in this tutorial

Thanks!

www.fiware.org
@Fiware

www.smartsdk.eu