Managing Context Information at large scale (Introduction)

www.fiware.org
@Fiware

Contact twitter
@fermingalan
@LeandroJGuillen

Contact email
fermin.galanmarquez@telefonica.com
leandro.guillen@imdea.org
kengunnar.zangelin@telefonica.com

(Reference Orion Context Broker version: 0.20.0)

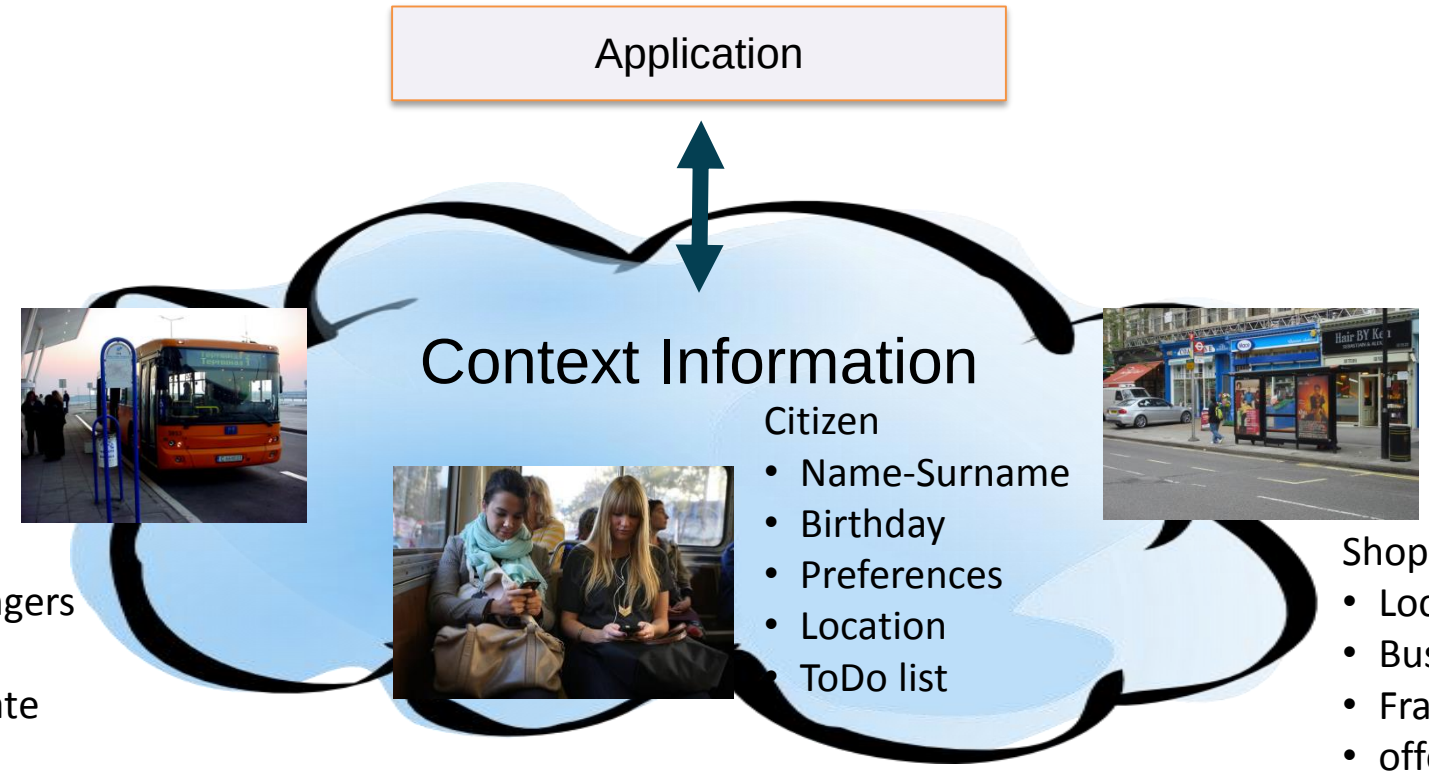
Introduction

Orion Context Broker

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

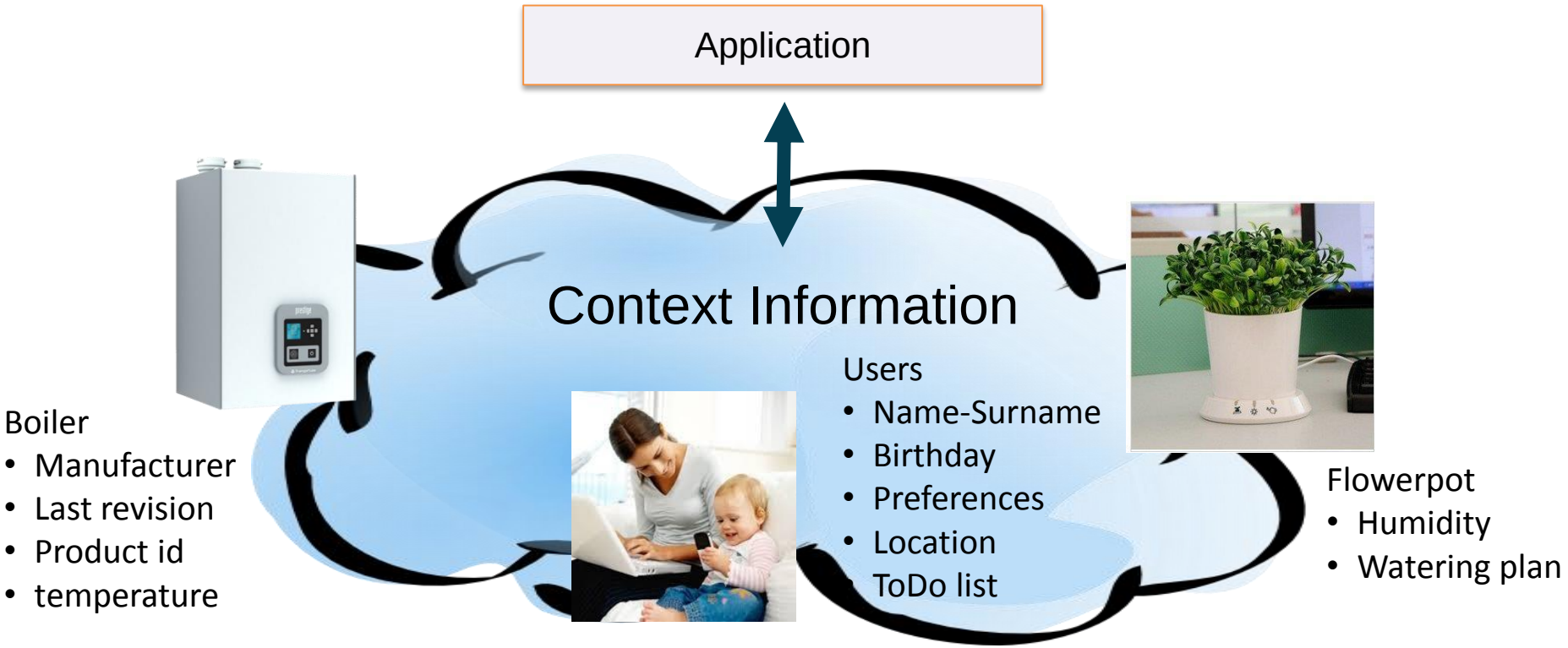
Being "Smart" requires first being "Aware"

- 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

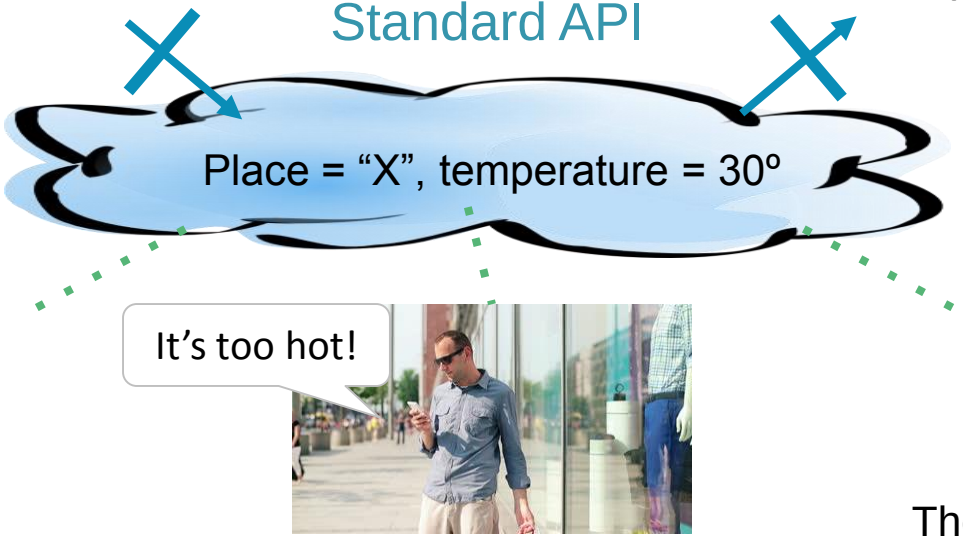


Different sources of context need to be handle

- Context information may come from many sources:
  - Existing systems
  - Users, through mobile apps
  - Sensor networks (Internet of Things)
- 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"



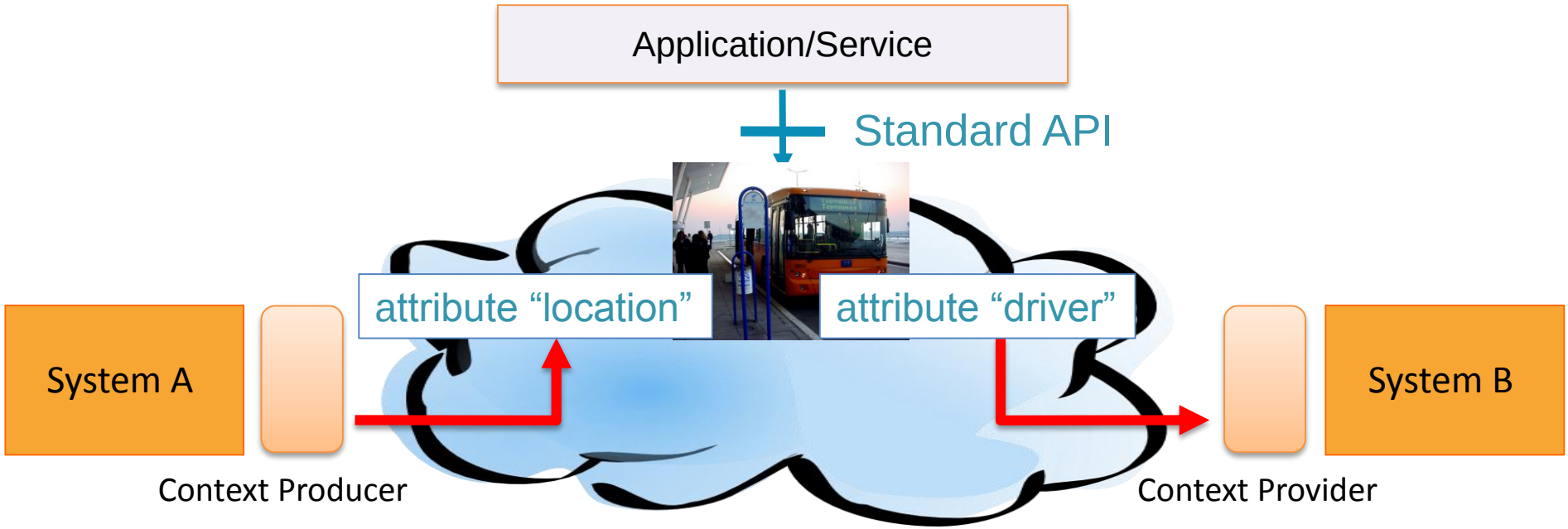
A sensor in a pedestrian street

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

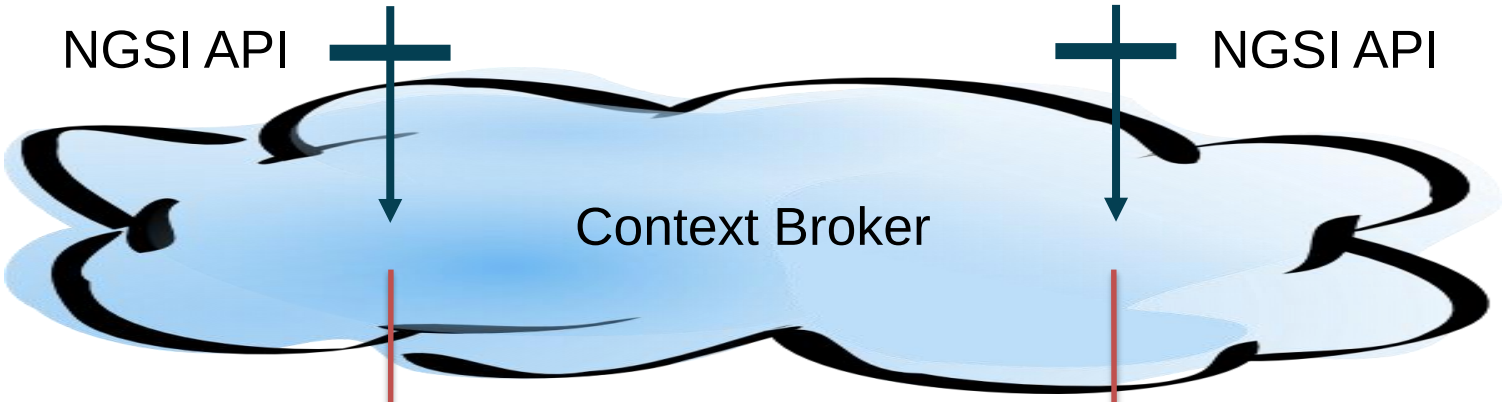


FIWARE NGSI: "The SNMP for IoT"

- 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>
/V1/contextEntities/lamp1/attributes/presenceSensor

PUT <Oauth token>
/V1/contextEntities/lamp1/attributes/status
"light on"



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

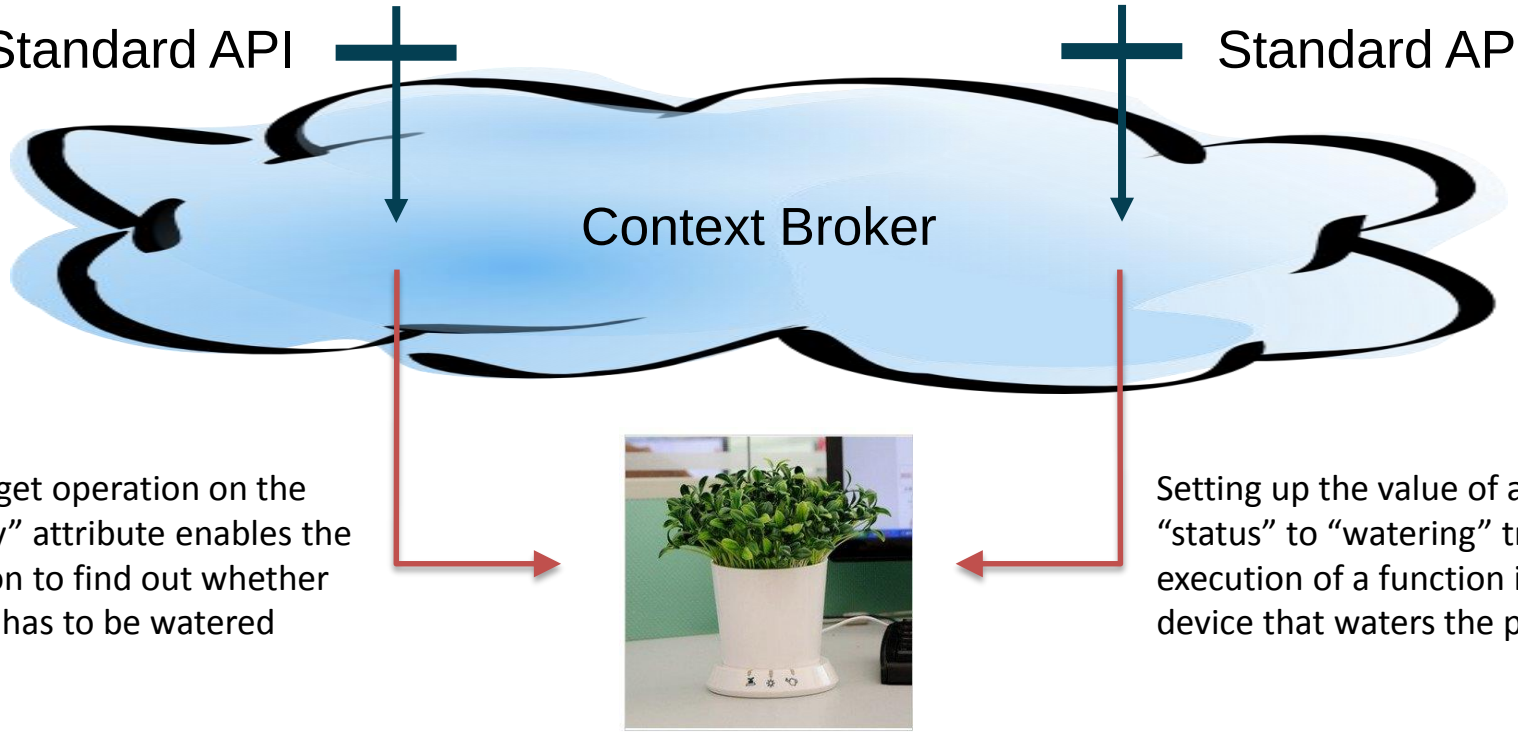
Setting up the value of attribute "status" to "light on" triggers execution of a function in the IoT device that switches the lamp on

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>
/V1/contextEntities/lamp1/attributes/presenceSensor

PUT <Oauth token>
/V1/contextEntities/lamp1/attributes/status
“light on”

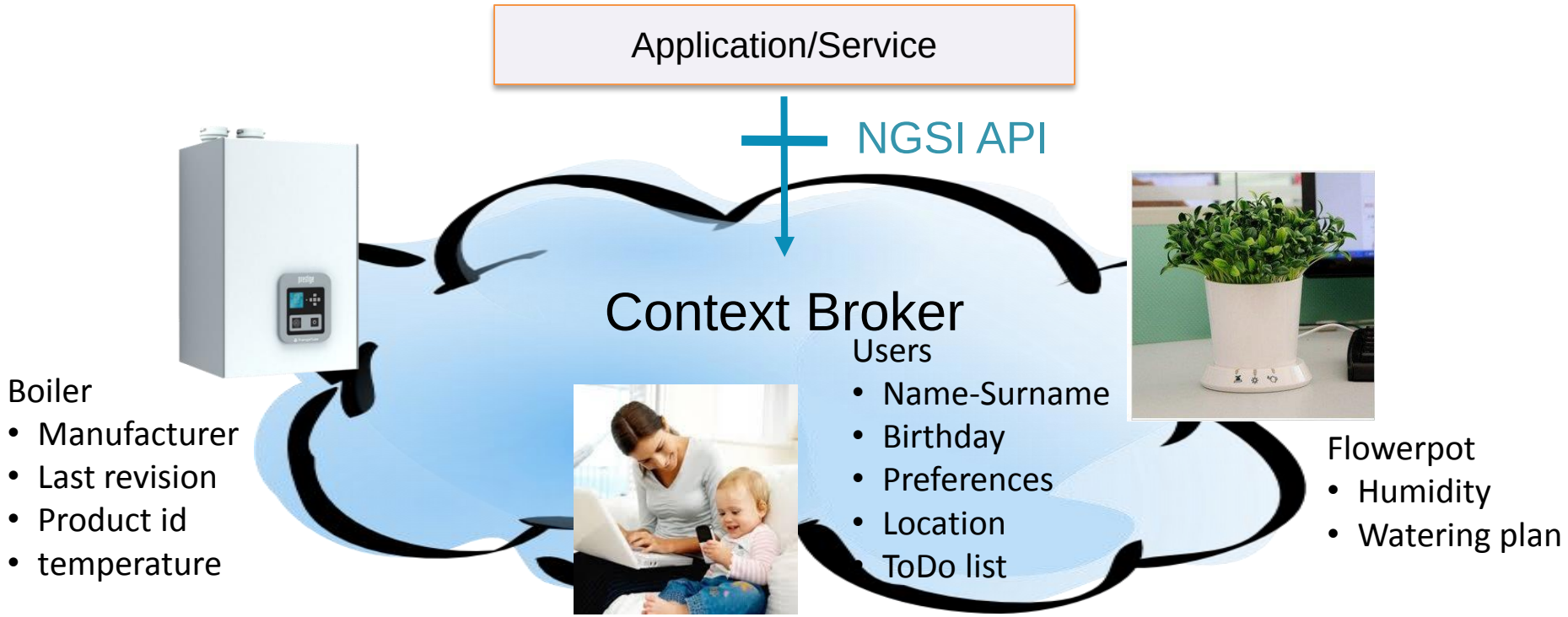


Issuing a get operation on the “humidity” attribute enables the application to find out whether the plant has to be watered

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

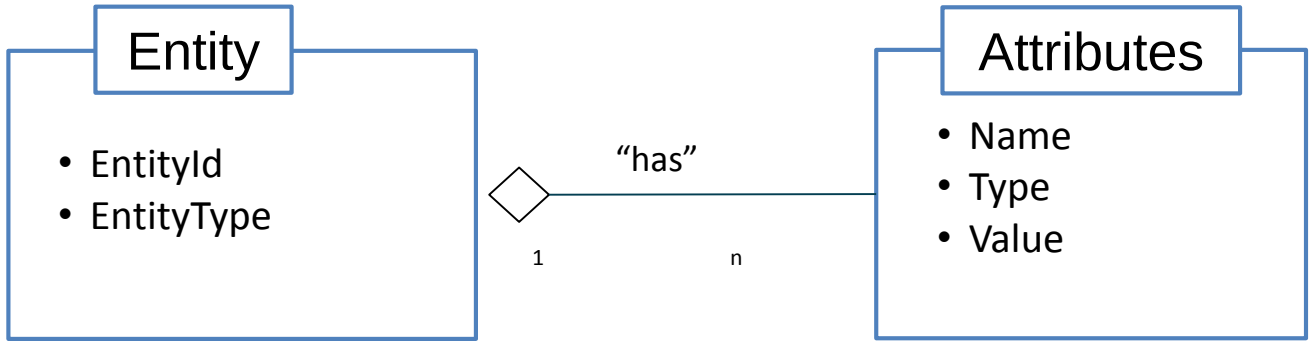
Context Management in FIWARE

- 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 gets quickly used to it

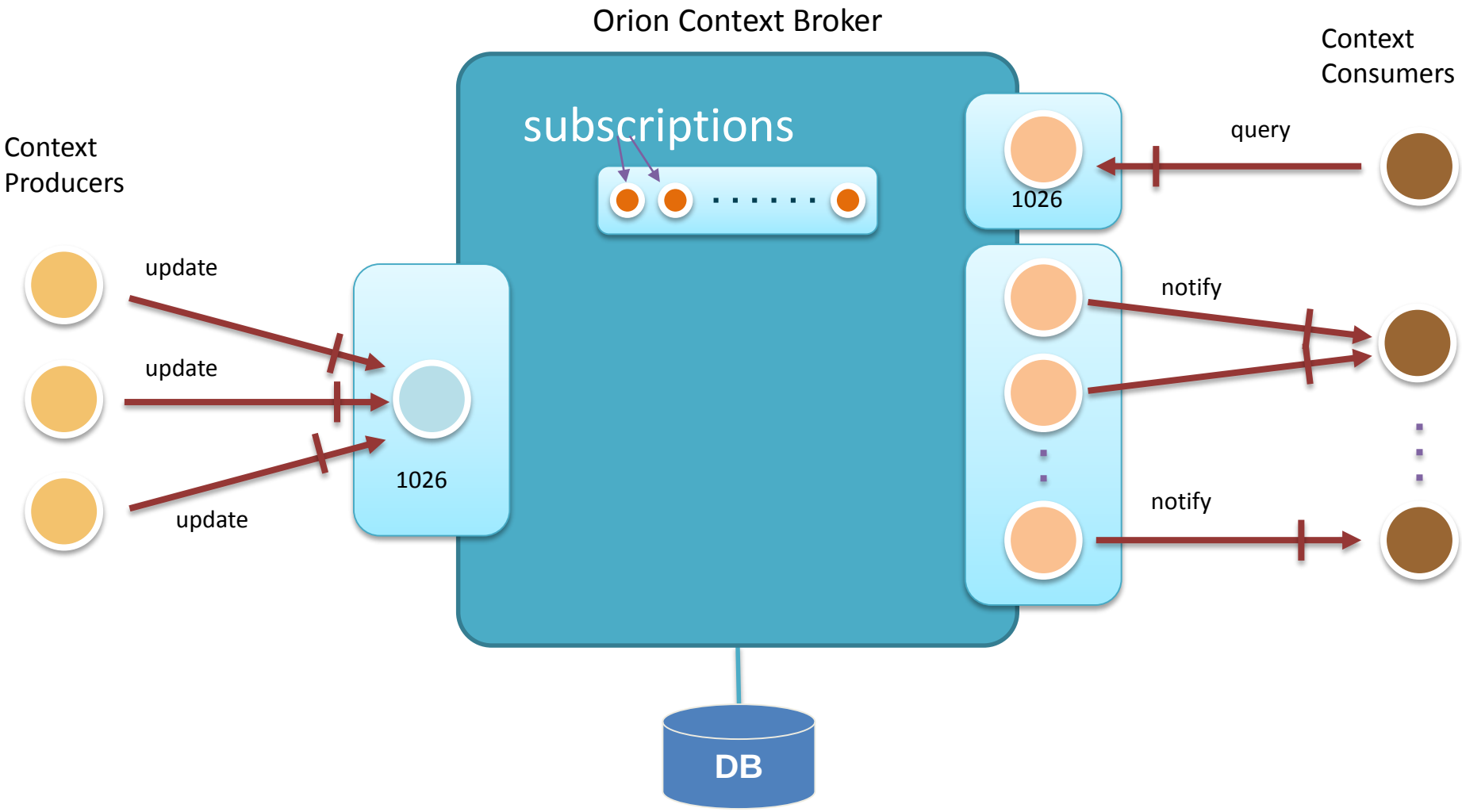


Orion Context Broker

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



Orion Context Broker in a nutshell



Orion Context Broker – check health

GET <cb_host>:1026/version


```
{
  "orion" : {
    "version" : "0.17.0",
    "uptime" : "7 d, 21 h, 33 m, 39 s",
    "git_hash" : "238c3642ad67899da7c1ff08aba4b5c846b4901a",
    "compile_time" : "Mon Dec 1 11:27:18 CET 2014",
    "compiled_by" : "fermin",
    "compiled_in" : "centollo"
  }
}
```

Orion Context Broker Basic Operations

Entities

- GET /v1/contextEntities/{entityID}
  - Retrieves an entity
- POST /v1/contextEntities/{entityID}
  - Creates an entity
- PUT /v1/contextEntities/{entityID}
  - Updates an entity
- DELETE /v1/contextEntities/{entityID}
  - Deletes an entity

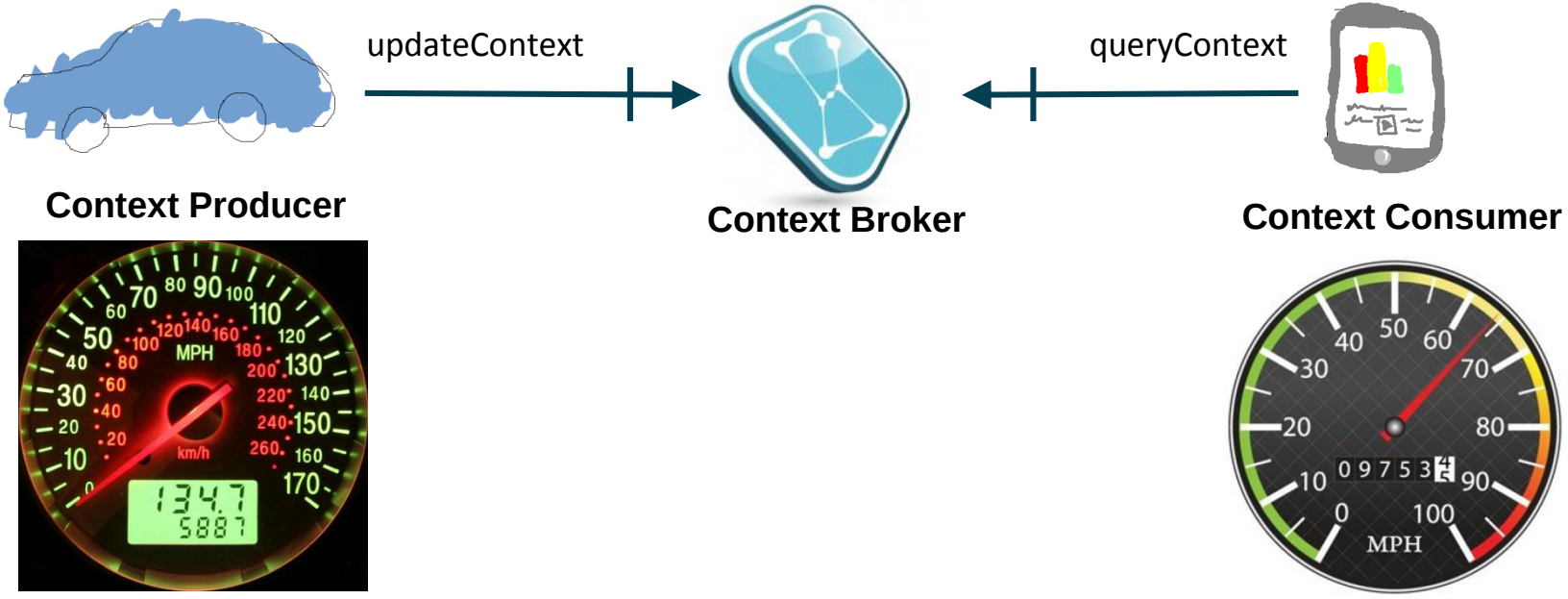
Orion Context Broker Basic Operations

Attributes

- GET /v1/contextEntities/{entityID}/attributes/{attrName}
  - Retrieves an attribute's value
- POST /v1/contextEntities/{entityID}/attributes/{attrName}
  - Creates a new attribute for an entity
- PUT /v1/contextEntities/{entityID}/attributes/{attrName}
  - Updates an attribute's value
- DELETE /v1/contextEntities/{entityID}/attributes/{attrName}
  - Deletes an attribute

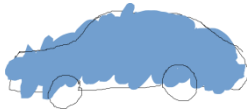
Context Broker operations: create & pull data

- **Context Producers** publish data/context elements by invoking the **updateContext** operation on a Context Broker.
- **Context Consumers** can retrieve data/context elements by invoking the **queryContext** operation on a Context Broker



Quick Usage Example: Car Create

POST <cb_host>:1026/v1/contextEntities



```
...
{
  "id": "Car1",
  "type": "Car",
  "attributes": [
  {
    "name": "speed",
    "type": "float",
    "value": "98"
  }
  ]
}
```

Entity id and type can be included in the URL, e.g.
POST /v1/contextEntities/type/Car/id/Car1

200 OK

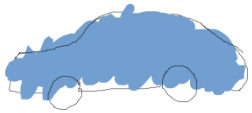
```
...
{
   "contextResponses": [
      {
         "attributes": [
            {
               "name": "speed",
               "type": "float",
               "value": ""
            }
         ],
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ],
   "id": “Car1",
   "isPattern": "false",
   "type": “Car"
}
```

Quick Usage Example: Car UpdateContext (1)

PUT <cb_host>:1026/v1/contextEntities/type/Car/id/Car1/attributes/**speed**

...

```
{
   "value": "110"
}
```



200 OK

...

```
{
   "code": "200",
   "reasonPhrase": "OK"
}
```

Quick Usage Example: Car QueryContext (1)

GET <cb_host>:1026/v1/contextEntities/type/Car/id/Car1/attributes/**speed**

200 OK

...


```
{
   "attributes": [
      {
         "name": "speed",
         "type": "float",
         "value": "110"
      }
   ],
   "statusCode": {
      "code": "200",
      "reasonPhrase": "OK"
   }
}
```


You can get all the attributes of the entity using the entity URL:

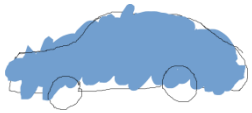
GET/v1/contextEntities/type/Car/id/Car1

Quick Usage Example: Car UpdateContext (2)

PUT <cb_host>:1026/v1/contextEntities/type/Car/id/Car1/attributes/**speed**

...

```
{
   "value": "115"
}
```



200 OK

...

```
{
   "code": "200",
   "reasonPhrase": "OK"
}
```

Quick Usage Example: Car QueryContext (2)

GET <cb_host>:1026/v1/contextEntities/type/Car/id/Car1/attributes/**speed**

200 OK

...


```
{
   "attributes": [
      {
         "name": "speed",
         "type": "float",
         "value": "115"
      }
   ],
   "statusCode": {
      "code": "200",
      "reasonPhrase": "OK"
   }
}
```

Quick Usage Example: Room Create (1)

POST <cb_host>:1026/v1/contextEntities

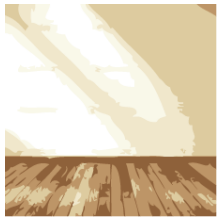
```
...
{
  "id": "Room1",
  "type": "Room",
  "attributes": [
  {
    "name": "temperature",
    "type": "float",
    "value": "24"
  },
  {
    "name": "pressure",
    "type": “integer",
    "value": "718"
  }
  ]
}
```

```
200 OK
...
{
   "contextResponses": [
      {
         "attributes": [
            {
               "name": "temperature",
               "type": "float",
               "value": ""
            },
            {
               "name": "pressure",
               "type": "float",
               "value": ""
            }
         ],
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ],
   "id": "Room1",
   "isPattern": "false",
   "type": "Room"
}
```

Quick Usage Example: Room UpdateContext (1)

PUT <cb_host>:1026/v1/contextEntities/type/**Room**/id/**Room1**

```
...
{
  "attributes": [
  {
    "name": "temperature",
    "type": "float",
    "value": "25"
  },
  {
    "name": "pressure",
    "type": "integer",
    "value": "720"
  }
  ]
}
```



200 OK

```
...
{
  "contextResponses": [
    {
      "attributes": [
        {
          "name": "temperature",
          "type": "float",
          "value": ""
        },
        {
          "name": "pressure",
          "type": "integer",
          "value": ""
        }
      ],
      "statusCode": {
        "code": "200",
        "reasonPhrase": "OK"
      }
    }
  ],
  "id": "Room1",
  "isPattern": "false",
  "type": "Room"
}
```

Quick Usage Example: Room QueryContext (1)

GET <cb_host>:1026/v1/contextEntities/type/Room/id/Room1/attributes/**temperature**

```
200 OK
...
{
   "attributes": [
      {
         "name": "temperature",
         "type": "float",
         "value": "25"
      }
   ],
   "statusCode": {
      "code": "200",
      "reasonPhrase": "OK"
   }
}
```

Quick Usage Example: Room QueryContext (2)

GET <cb_host>:1026/v1/contextEntities/type/**Room**/id/**Room1**

```
200 OK
...
{
   "contextElement": {
      "attributes": [
         {
            "name": "temperature",
            "type": "float",
            "value": "25"
         },
         {
            "name": "pressure",
            "type": "float",
            "value": "720"
         }
      ],
      "id": "Room1",
      "isPattern": "false",
      "type": "Room"
   },
   "statusCode": {
      "code": "200",
      "reasonPhrase": "OK"
   }
}
```

Quick Usage Example: Room Create (2)

POST <cb_host>:1026/v1/contextEntities

```
...
{
  "id": "Room2",
  "type": "Room",
  "attributes": [
  {
    "name": "temperature",
    "type": "float",
    "value": "29"
  },
  {
    "name": "pressure",
    "type": “integer",
    "value": "730"
  }
  ]
}
```

```
200 OK
...
{
   "contextResponses": [
      {
         "attributes": [
            {
               "name": "temperature",
               "type": "float",
               "value": ""
            },
            {
               "name": "pressure",
               "type": "float",
               "value": ""
            }
         ],
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ],
   "id": "Room1",
   "isPattern": "false",
   "type": "Room"
}
```

Quick Usage Example: Room QueryContext (3)

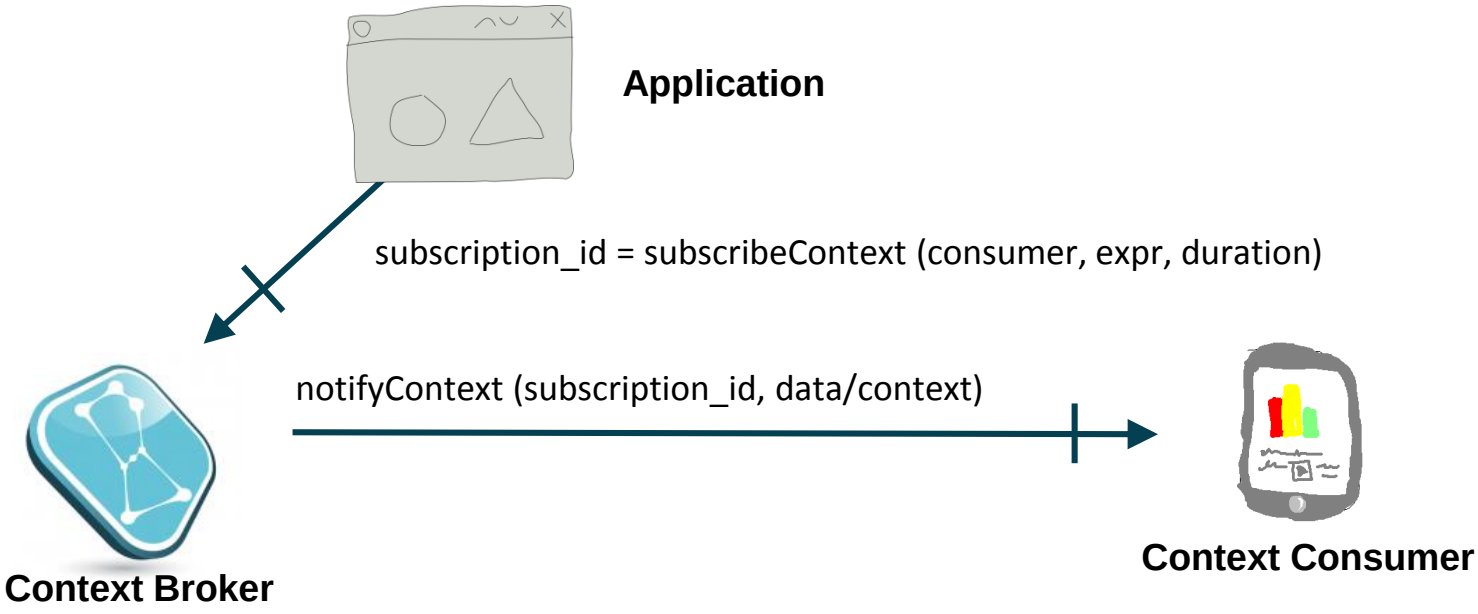
```
POST <cb_host>:1026/v1/queryContext
...
{
   "entities": [
      {
         "type": "Room",
         "isPattern": "true",
         "id": "Room.*"
      },
      "attributes": [
         "temperature"
      ]
   ]
}
```

```
200 OK
...
{
   "contextResponses": [
      {
         "contextElement": {
            "attributes": [
               {
                  "name": "temperature",
                  "type": "float",
                  "value": "25"
               }
            ],
            "id": "Room1",
            "isPattern": "false",
            "type": "Room"
         },
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      },
      {
         "contextElement": {
            "attributes": [
               {
                  "name": "temperature",
                  "type": "float",
                  "value": "29"
               }
            ],
            "id": "Room2",
            "isPattern": "false",
            "type": "Room"
         },
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ]
}
```

Context Broker operations: push data

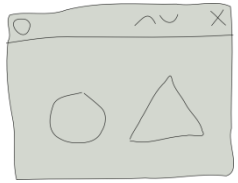
- **Context Consumers** can subscribe to receive context information that satisfy certain conditions using the **subscribeContext**. Such subscriptions may have a duration.

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



Quick Usage Example: Subscription

POST <cb_host>:1026/v1/subscribeContext



```
...
{
   "entities": [
      {
         "type": "Room",
         "isPattern": "false",
         "id": "Room1"
      }
   ],
   "attributes": [
      "temperature"
   ],
   "reference": "http://<host>:<port>/publish",
   "duration": "P1M",
   "notifyConditions": [
      {
         "type": "ONCHANGE",
         "condValues": [
            "temperature"
         ]
      }
   ],
   "throttling": "PT5S"
}
```

200 OK

```
...
{
   "subscribeResponse": {
      "duration": "P1M",
      "subscriptionId": "51c0ac9ed714fb3b37d7d5a8",
      "throttling": "PT5S"
   }
}
```

Quick Usage Example: Notification

25

19

Quick Usage Example: Notification

POST http://<host>:<port>/publish

```
...
{
 "subscriptionId" : "51c0ac9ed714fb3b37d7d5a8",
 "originator" : "localhost",
 "contextResponses" : [
  {
   "contextElement" : {
    "attributes" : [
     {
      "name" : "temperature",
      "type" : "float",
      "value" : "19"
     }
    ],
    "type" : "Room",
    "isPattern" : "false",
    "id" : "Room1"
   },
   "statusCode" : {
    "code" : "200",
    "reasonPhrase" : "OK"
   }
  }
 ]
}
```

Orion Context Broker Standard Operations

	Functions	Operations
NGSI10	• Query, • Update and • Subscribe to context elements	• updateContext • queryContext • subscribeContext • updateContextSubscription • unsubscribeContextSubscription

- They are equivalent to previous convenience operations in functionality
- All them use POST as verb, including the parameters in the payload
- More expressive than convenience operations. Some advance functionality is only supported with standar operations (e.g. geo-aware query)
- They are not a substitute but a complement to convenience operations

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
  - 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

Managing Context Information at large scale (Advanced Topics)

www.fiware.org
@Fiware

Contact twitter
@fermingalan
@LeandroJGuillen

Contact email
fermin.galanmarquez@telefonica.com
leandro.guillen@imdea.org
kengunnar.zangelin@telefonica.com

(Reference Orion Context Broker version: 0.20.0)

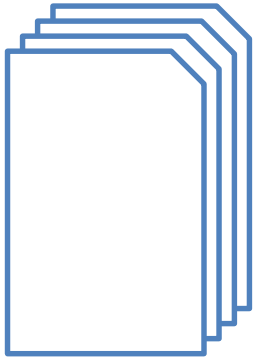
Advanced Features

Orion Context Broker

- Pagination
- Compound attribute values
- Geo-location
- Metadata
- Registrations & context providers
- Multitenancy
- Entity service paths

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)
  - details
    - Returns total elements (default: "off")



Pagination

- Example, querying the first 100 entries:

  POST <orion_host>:1026/v1/queryContext?limit=100&details=on

- The first 100 elements are returned, along with the following errorCode in the response:

```
"errorCode": {
      "code": "200",
      "details": "Count: 322",
      "reasonPhrase": "OK"
   }
```

- Now we now there are 322 entities, we can keep querying the broker for them:
  - POST <orion_host>:1026/v1/queryContext?offset=100&limit=100
  - POST <orion_host>:1026/v1/queryContext?offset=200&limit=100
  - POST <orion_host>:1026/v1/queryContext?offset=300&limit=100

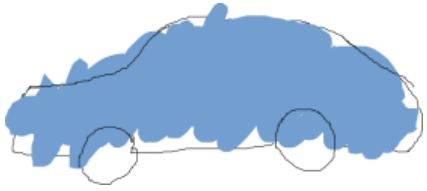
Compound Attribute Values

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

Compound Attribute 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:

```
{
  "contextElements": [
    {
      "type": "Car",
      "isPattern": "false",
      "id": "Car1",
      "attributes": [
      {
        "name": "tirePressure",
        "type": "kPa",
        "value": {
          "frontRight": "120",
          "frontLeft": "110",
          "backRight": "115",
          "backLeft": "130"
        }
      }
      ]
    }
  ],
  "updateAction": "APPEND"
}
```



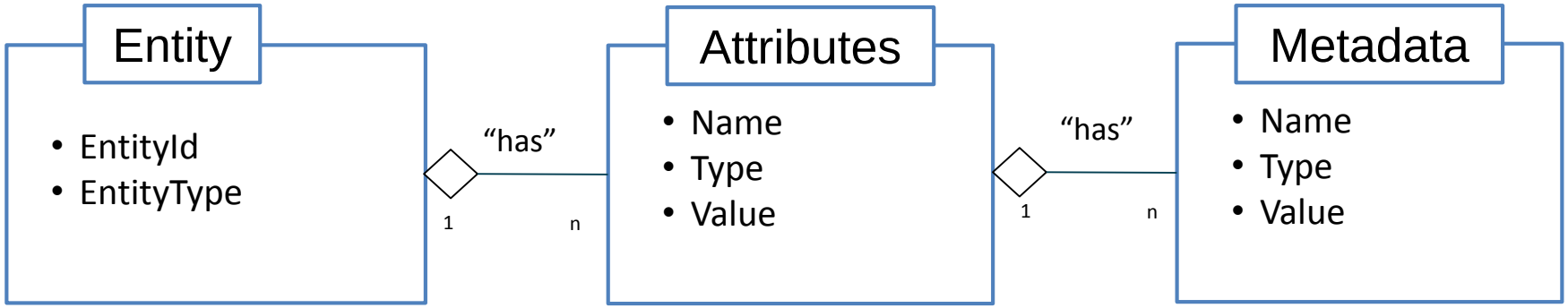
Metadata

- Users may attach metadata to attributes
- Reserved metadatas: ID, Location, creDate and modDate
- Examples:

```
…
"attributes": [
  {
   "name": "temperature",
   "type": "float",
   "value": "26.5",
   "metadatas": [
   {
    "name": "accuracy",
    "type": "float",
    "value": "0.9"
   }
   ]
  }
  ]
…
```

```
…
"attributes": [
  {
   "name": "temperature",
   "type": "float",
   "value": "26.5",
   "metadatas": [
   {
    "name": "average",
    "type": "float",
    "value": "22.4"
   }
   ]
  }
  ]
…
```

Complete NGSI Model



Geo-location

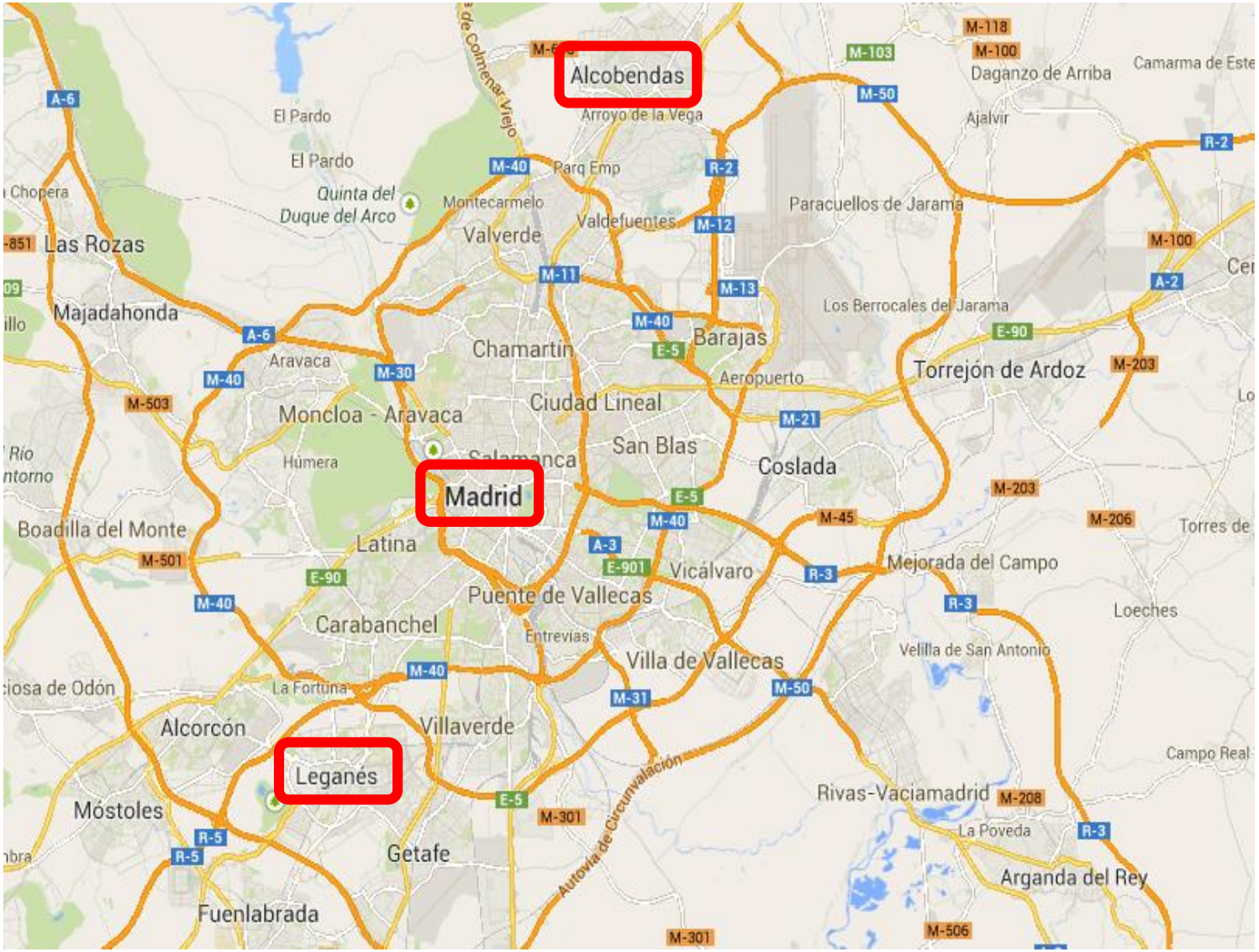
- Entities can have an attribute that specifies its location
  - Using a "location" metadata
- Example: create an entity called Madrid

…and create a couple more towns:

- Leganés
- Alcobendas

```
POST <cb_host>:1026/v1/updateContext
{
 "contextElements": [
 {
  "type": "City",
  "isPattern": "false",
  "id": "Madrid",
  "attributes": [
  {
   "name": "position",
   "type": "coords",
   "value": "40.418889, -3.691944",
   "metadatas": [
    {
     "name": "location",
     "type": "string",
     "value": "WSG84"
    }
   ]
  }
  ]
 }
 ],
 "updateAction": "APPEND"
}
```

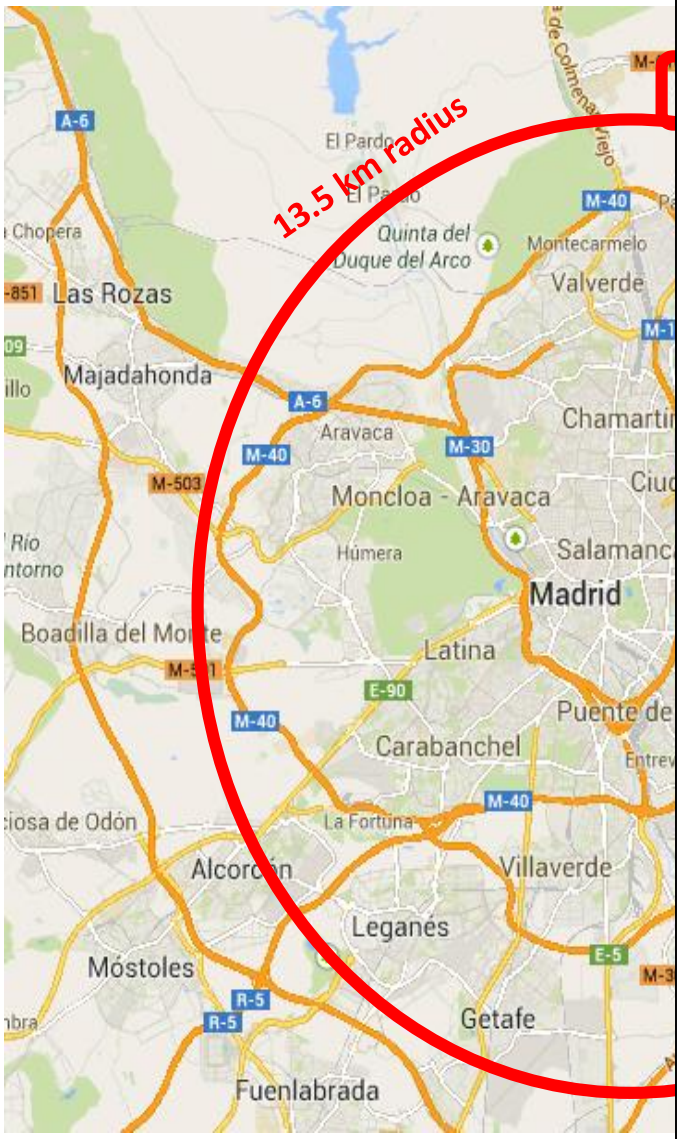
Geo-location – Circle



Geo-location – Circle


```
POST <cb_host>:1026/v1/queryContext
...
{
 "entities": [
 {
  "type": "City",
  "isPattern": "true",
  "id": ".*"
 }
 ],
 "restriction": {
  "scopes": [
  {
   "type" : "FIWARE::Location",
   "value" : {
    "circle": {
     "centerLatitude": "40.418889",
     "centerLongitude": "-3.691944",
     "radius": "13500"
    }
   }
  }
  ]
 }
}
```

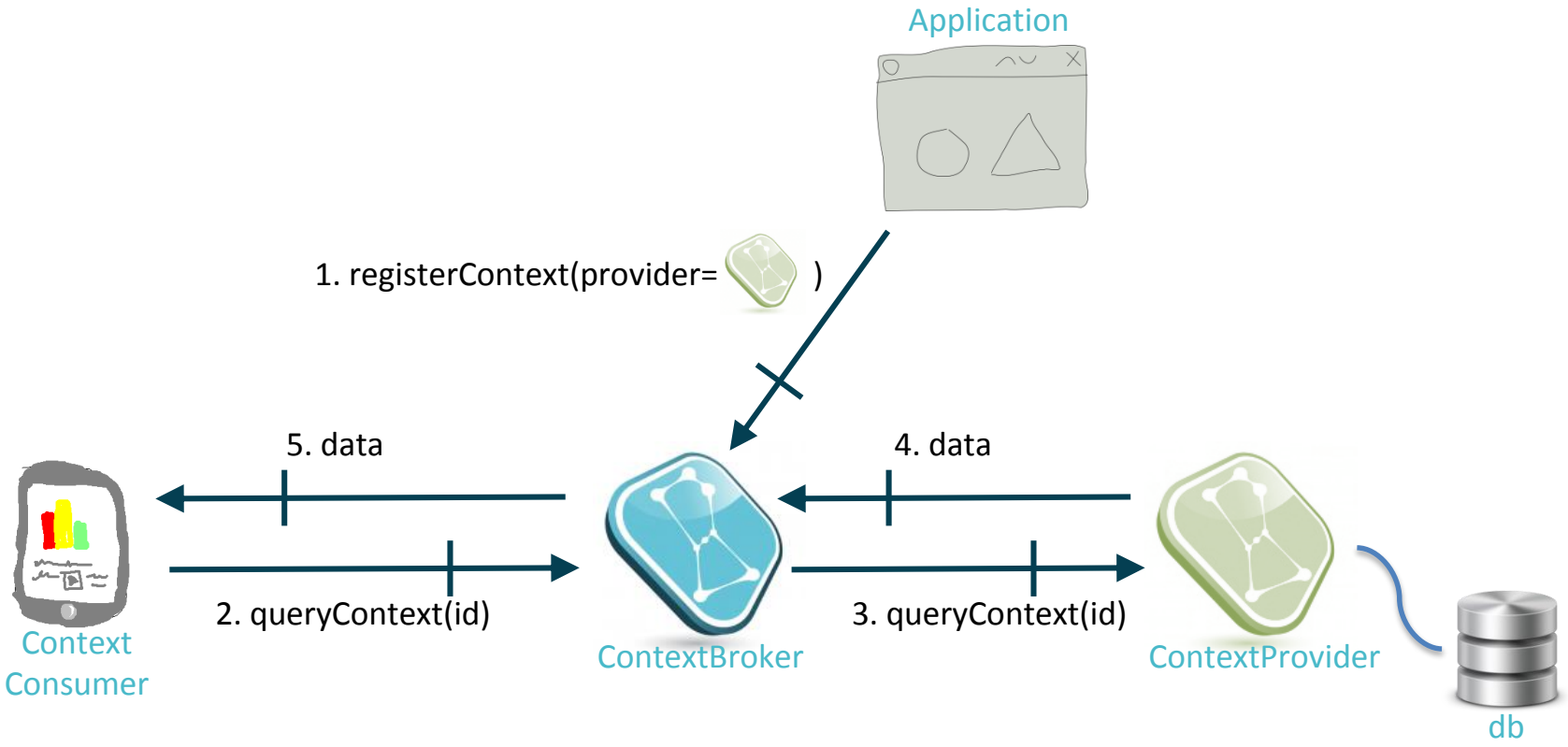
Geo-location – Inverse Circle




```
POST <cb_host>:1026/v1/queryContext
...
{
 "entities": [
 {
  "type": "City",
  "isPattern": "true",
  "id": ".*"
 }
 ],
 "restriction": {
  "scopes": [
  {
   "type" : "FIWARE::Location",
   "value" : {
    "circle": {
     "centerLatitude": "40.418889",
     "centerLongitude": "-3.691944",
     "radius": "13500",
     "inverted": "true"
    }
   }
  }
  ]
 }
}
```

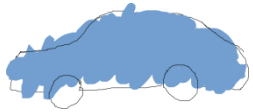
Registration & Context Providers

- Uncached queries and updates



Registration & Context Providers

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



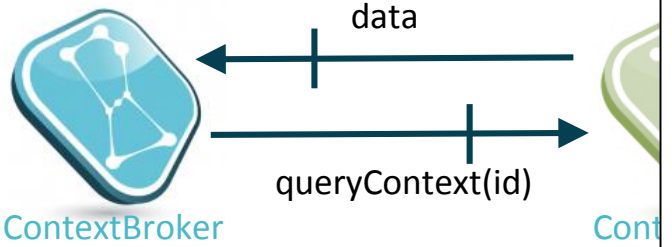
```
...
{
   "contextRegistrations": [
      {
         "entities": [
            {
               "type": "Car",
               "isPattern": "false",
               "id": "Car1"
            },
            "attributes": [
            {
               "name": "speed",
               "type": "float",
               "isDomain": "false"
            }
         ],
         "providingApplication": "http://contextprovider.com/Cars"
      }
   ],
   "duration": "P1M"
}
```

200 OK

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

Registration & Context Providers

```
POST <cb_host>:1026/v1/queryContext
...
{
   "entities": [
      {
         "type": "Car",
         "isPattern": "false",
         "id": "Car1"
      }
   ]
}
```




```
200 OK
...
{
   "contextResponses": [
      {
         "contextElement": {
            "attributes": [
               {
                  "name": "speed",
                  "type": "float",
                  "value": "100"
               }
            ],
            "id": "Car1",
            "isPattern": "false",
            "type": "Car"
         },
         "statusCode": {
            "code": "200",
            "details": "Redirected to context provider
http://contextprovider.com/Cars",
            "reasonPhrase": "OK"
         }
      }
   ]
}
```

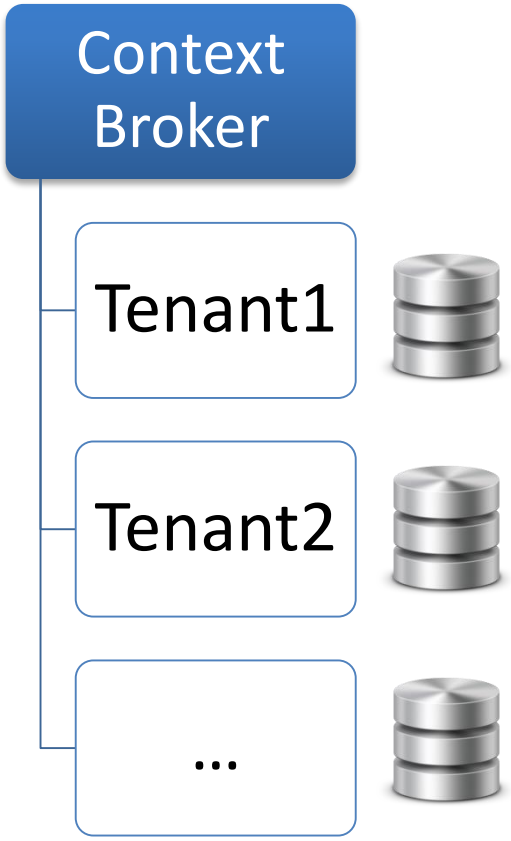
Orion Context Broker - Operations

	Functions	Operations
NGSI9	• Register, • Search and • Subscribe for context sources	• registerContext • discoverContextAvailability • subscribeContextAvailability • updateContextAvailabilitySubscription • unsubscribeContextAvailability
NGSI10	• Query, • Update and • Subscribe to context elements	• updateContext • queryContext • subscribeContext • updateContextSubscription • unsubscribeContextSubscription

Multitenancy

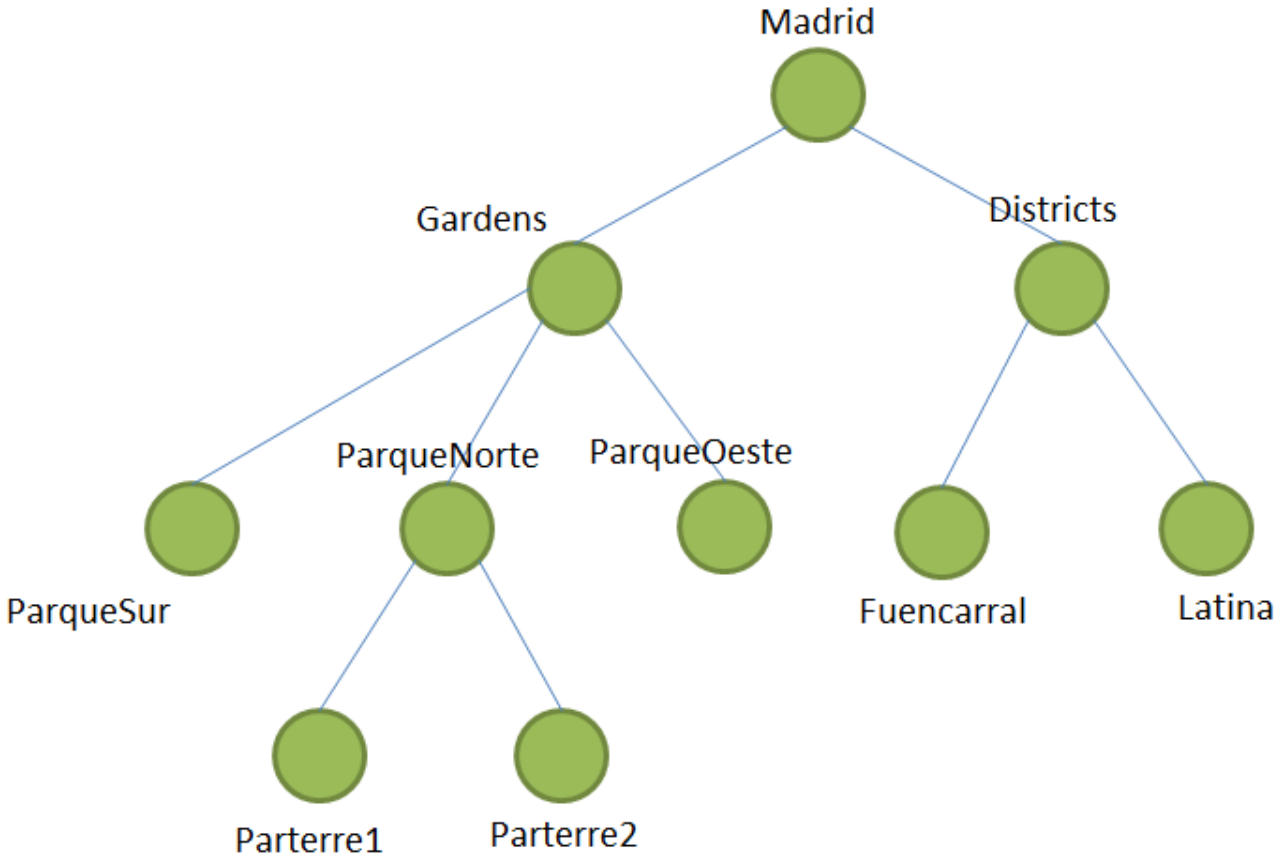
- Simple multitenant model based on logical database separation.
- It eases tenant-based authorization provided by other components.
- Just use an additional HTTP header called "Fiware-Service", whose value is the tenant name. Example:

  Fiware-Service: Tenant1



Entity Service Paths

- A service path is a hierarchical scope assigned to an entity at creation time (with updateContext).

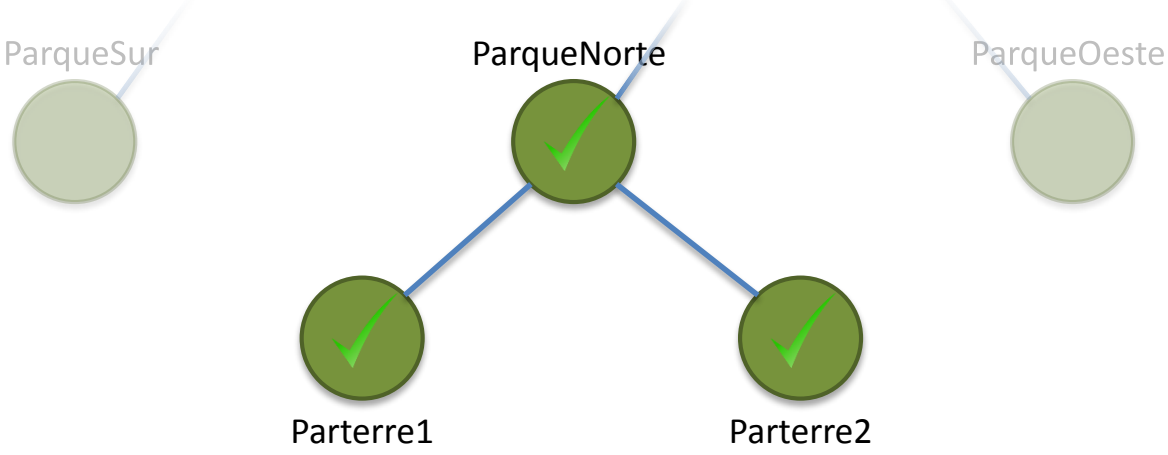


Entity Service Paths

- 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

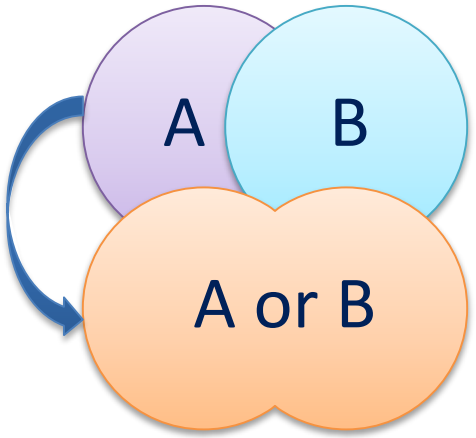
- Properties:
  - A query on a service path will look only into the specified node
  - Use "ParentNode/#" to include all child nodes
  - Queries without Fiware-ServicePath resolve to "/#"
  - Entities will fall in the "/" node by default

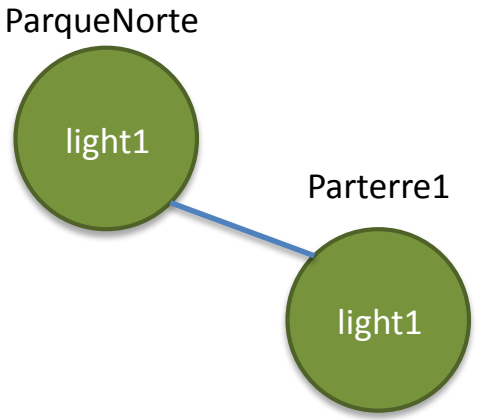


Entity Service Paths

- 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)





Bonus Track: Orion Context Explorer

- Publicly available browser-based front-end for Orion Context Broker
  - Open source development by VM9
- Authentication integrated with FIWARE Lab account

- **Have a look!**
  - http://orionexplorer.com/

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
  - 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

Thanks!

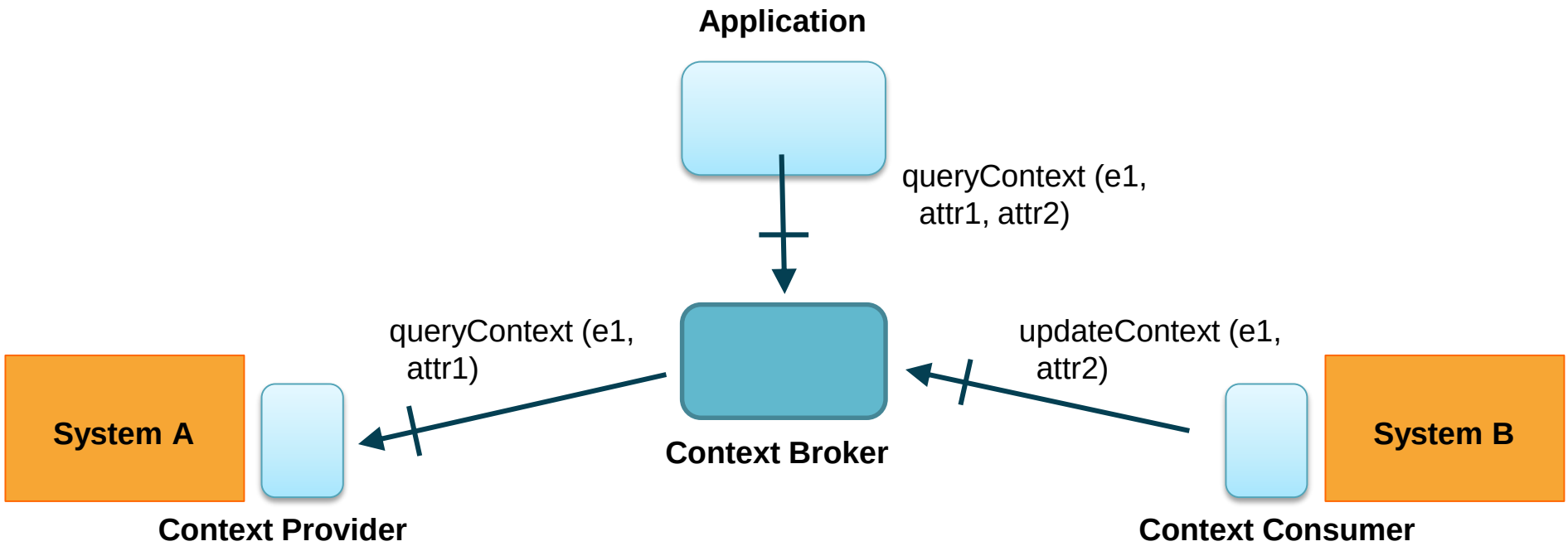
www.fiware.org
@Fiware

Backup slides

BACKUP SLIDES

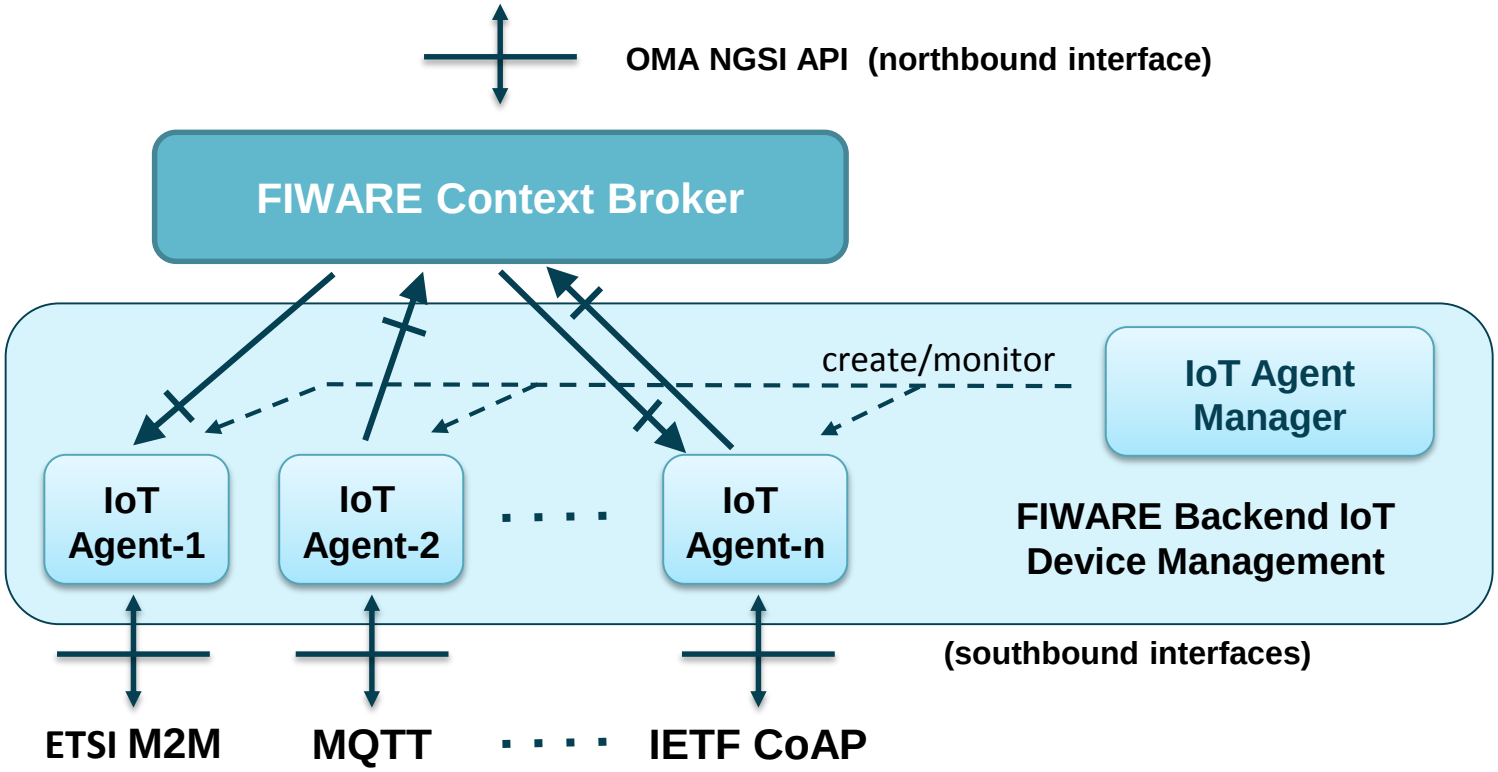
Integration with existing systems

- 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

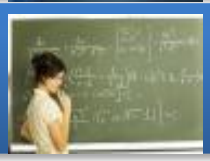


Integration with sensor networks

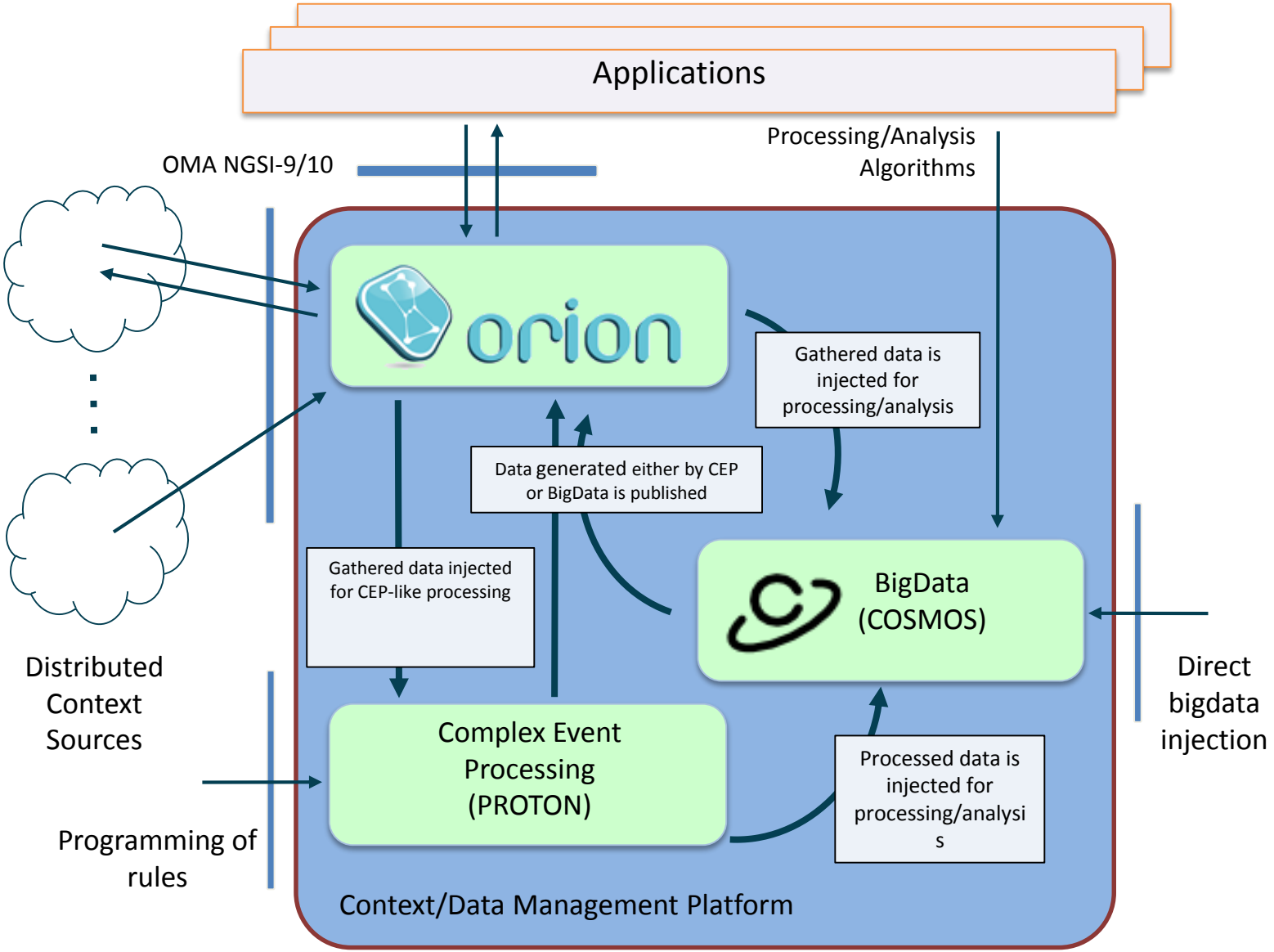
- The backend IoT Device Management GE enables creation and configuration of NGSI IoT Agents that connect to sensor networks
- Each NGSI IoT Agent can behave as Context Consumers or Context Providers, or both



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

FI-WARE Context/Data Management Platform



More on Convenience Operations

Subscriptions

- POST /v1/contextSubscriptions
  - Creates a subscription
- PUT /v1/contextSubscriptions/{subID}
  - Updates a subscription
- DELETE /v1/contextSubscriptions/{subID}
  - Cancel a subscription

More on Convenience Operations

Entity Types

- GET /v1/contextTypes
  - Retrieve a list of all entity types currently in Orion, including their corresponding attributes
- GET /v1/contextTypes/{typeID}
  - Retrieve attributes associated to an entity type

PRO TIP

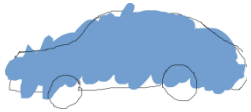
GET /v1/contextTypes?collapse=true

Retrieves a list of all entity types without attribute info

Std Op Example: Car Create

POST localhost:1026/v1/updateContext

...



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

200 OK

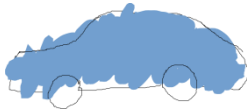
...

```
{
   "contextResponses": [
      {
         "contextElement": {
            "attributes": [
               {
                  "name": "speed",
                  "type": "float",
                  "value": ""
               }
            ],
            "id": "Car1",
            "isPattern": "false",
            "type": "Car"
         },
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ]
}
```

Std Op Example: Car UpdateContext (1)

POST localhost:1026/v1/updateContext

...



```
{
   "contextElements": [
      {
         "type": "Car",
         "isPattern": "false",
         "id": "Car1",
         "attributes": [
         {
            "name": "speed",
            "type": "float",
            "value": "110"
         }
         ]
      }
   ],
   "updateAction": "UPDATE"
}
```

200 OK

...

```
{
   "contextResponses": [
      {
         "contextElement": {
            "attributes": [
               {
                  "name": "speed",
                  "type": "float",
                  "value": ""
               }
            ],
            "id": "Car1",
            "isPattern": "false",
            "type": "Car"
         },
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ]
}
```

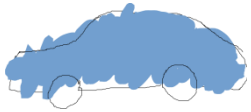
Std Op Example: Car QueryContext (1)

```
POST <cb_host>:1026/v1/queryContext
...
{
   "entities": [
      {
         "type": "Car",
         "isPattern": "false",
         "id": "Car1"
      }
   ]
}
```

```
200 OK
...
{
   "contextResponses": [
      {
         "contextElement": {
            "attributes": [
               {
                  "name": "speed",
                  "type": "float",
                  "value": "110"
               }
               ],
            "id": "Car1",
            "isPattern": "false",
            "type": "Car"
         },
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ]
}
```

Std Op Example: Car UpdateContext (2)

POST localhost:1026/v1/updateContext



```
...
{
   "contextElements": [
      {
         "type": "Car",
         "isPattern": "false",
         "id": "Car1",
         "attributes": [
         {
            "name": "speed",
            "type": "float",
            "value": "115"
         }
         ]
      }
   ],
   "updateAction": "UPDATE"
}
```

200 OK

```
...
{
   "contextResponses": [
      {
         "contextElement": {
            "attributes": [
               {
                  "name": "speed",
                  "type": "float",
                  "value": ""
               }
            ],
            "id": "Car1",
            "isPattern": "false",
            "type": "Car"
         },
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ]
}
```

Std Op Example: Car QueryContext (2)

```
POST <cb_host>:1026/v1/queryContext
...
{
   "entities": [
      {
         "type": "Car",
         "isPattern": "false",
         "id": "Car1"
      }
   ]
}
```

```
200 OK
...
{
   "contextResponses": [
      {
         "contextElement": {
            "attributes": [
               {
                  "name": "speed",
                  "type": "float",
                  "value": "115"
               }
               ],
            "id": "Car1",
            "isPattern": "false",
            "type": "Car"
         },
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ]
}
```

Std Op Example: Room Create (1)

POST localhost:1026/v1/updateContext

```
...
{
   "contextElements": [
      {
         "type": "Room",
         "isPattern": "false",
         "id": "Room1",
         "attributes": [
         {
            "name": "temperature",
            "type": "float",
            "value": "24"
         },
         {
            "name": "pressure",
            "type": "integer",
            "value": "718"
         }

         ]
      }
   ],
   "updateAction": "APPEND"
}
```

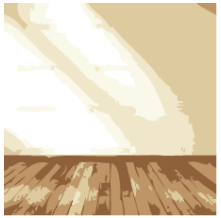
200 OK

```
...
{
   "contextResponses": [
      {
         "contextElement": {
            "attributes": [
               {
                  "name": "temperature",
                  "type": "float",
                  "value": ""
               },
               {
                  "name": "pressure",
                  "type": "integer",
                  "value": ""
               }
            ],
            "id": "Room1",
            "isPattern": "false",
            "type": "Room"
         },
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ]
}
```

Std Op Example: Room UpdateContext (1)

POST localhost:1026/v1/updateContext

...



```
{
   "contextElements": [
      {
         "type": "Room",
         "isPattern": "false",
         "id": "Room1",
         "attributes": [
         {
            "name": "temperature",
            "type": "float",
            "value": "25"
         },
         {
            "name": "pressure",
            "type": "integer",
            "value": "720"
         }

         ]
      }
   ],
   "updateAction": "UPDATE"
}
```

200 OK

...

```
{
   "contextResponses": [
      {
         "contextElement": {
            "attributes": [
               {
                  "name": "temperature",
                  "type": "float",
                  "value": ""
               },
               {
                  "name": "pressure",
                  "type": "integer",
                  "value": ""
               }
            ],
            "id": "Room1",
            "isPattern": "false",
            "type": "Room"
         },
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ]
}
```

Std Op Example: Room QueryContext (1)

```
POST <cb_host>:1026/v1/queryContext
...
{
   "entities": [
      {
         "type": "Room",
         "isPattern": "false",
         "id": "Room1"
      },
      "attributes": [
         "temperature"
      ]
   ]
}
```

```
200 OK
...
{
   "contextResponses": [
      {
         "contextElement": {
            "attributes": [
               {
                  "name": "temperature",
                  "type": "float",
                  "value": "25"
               }
               ],
            "id": "Room1",
            "isPattern": "false",
            "type": "Room"
         },
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ]
}
```

Std Op Example: Room QueryContext (2)

```
POST <cb_host>:1026/v1/queryContext
...
{
   "entities": [
      {
         "type": "Room",
         "isPattern": "false",
         "id": "Room1"
      }
   ]
}
```

```
200 OK
...
{
   "contextResponses": [
      {
         "contextElement": {
            "attributes": [
               {
                  "name": "temperature",
                  "type": "float",
                  "value": "25"
               },
               {
                  "name": "pressure",
                  "type": "integer",
                  "value": "720"
               }
            ],
            "id": "Room1",
            "isPattern": "false",
            "type": "Room"
         },
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ]
}
```

Std Op Example: Room Create (2)

POST localhost:1026/v1/updateContext

```
...
{
   "contextElements": [
      {
         "type": "Room",
         "isPattern": "false",
         "id": "Room2",
         "attributes": [
         {
            "name": "temperature",
            "type": "float",
            "value": “29"
         },
         {
            "name": "pressure",
            "type": "integer",
            "value": "730"
         }

         ]
      }
   ],
   "updateAction": "APPEND"
}
```

200 OK

```
...
{
   "contextResponses": [
      {
         "contextElement": {
            "attributes": [
               {
                  "name": "temperature",
                  "type": "float",
                  "value": ""
               },
               {
                  "name": "pressure",
                  "type": "integer",
                  "value": ""
               }
            ],
            "id": "Room1",
            "isPattern": "false",
            "type": "Room"
         },
         "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
         }
      }
   ]
}
```