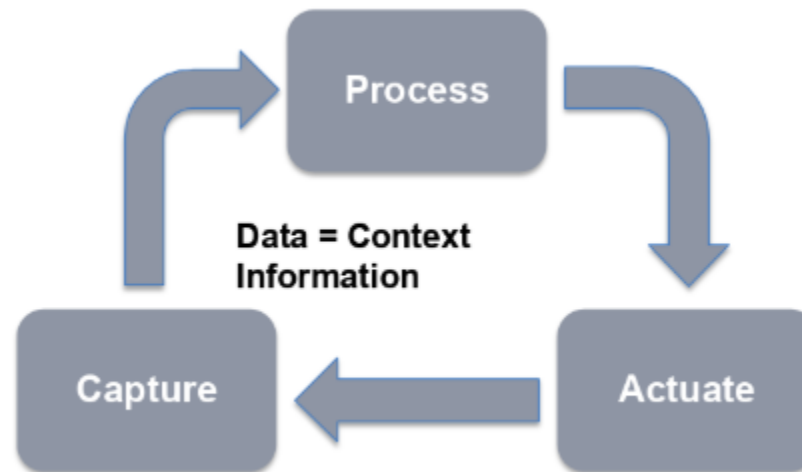


Internet of Things

Intro to FIWARE

Smart solutions

- **Smart solutions** gather data from many different sources (including but not limited to IoT) to build a **digital twin representation** of the real world (also referred as **context representation**) which is constantly analyzed and processed in order to automate certain processes or bring support to smart decisions.



Challenges

- **Different data formats** between verticals.
- **Company policies** do not support the provision and exchange of data.
- Information coming from **multiple sources** must be accessible in **real-time** creating a **digital continuum**.
- Breaking down **boundaries between domains** will enable the exchange of relevant data across multiple applications.

Context creates a digital continuum, blurring the frontiers between application domains
Breaking the current silos of information



Source: <https://www.slideshare.net/FI-WARE/fiware-wednesday-webinars-fiware-overview>

What is context data?

- An **entity** represents the state of a physical or conceptual object which exists in the real world.
- The **context data** of that entity defines the state of that real-world object at a given moment in time.
- **Context data** describes what is going on, where, when, why...

Modeling context

... in Cities



Shop

- Location
- Business name
- Franchise
- offerings



Citizen

- Birthday
- Preferences
- Location
- ToDo list



Bus

- Location
- No. passengers
- Driver
- License plate

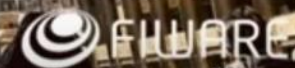
Entities

Attribute



Incident / claim

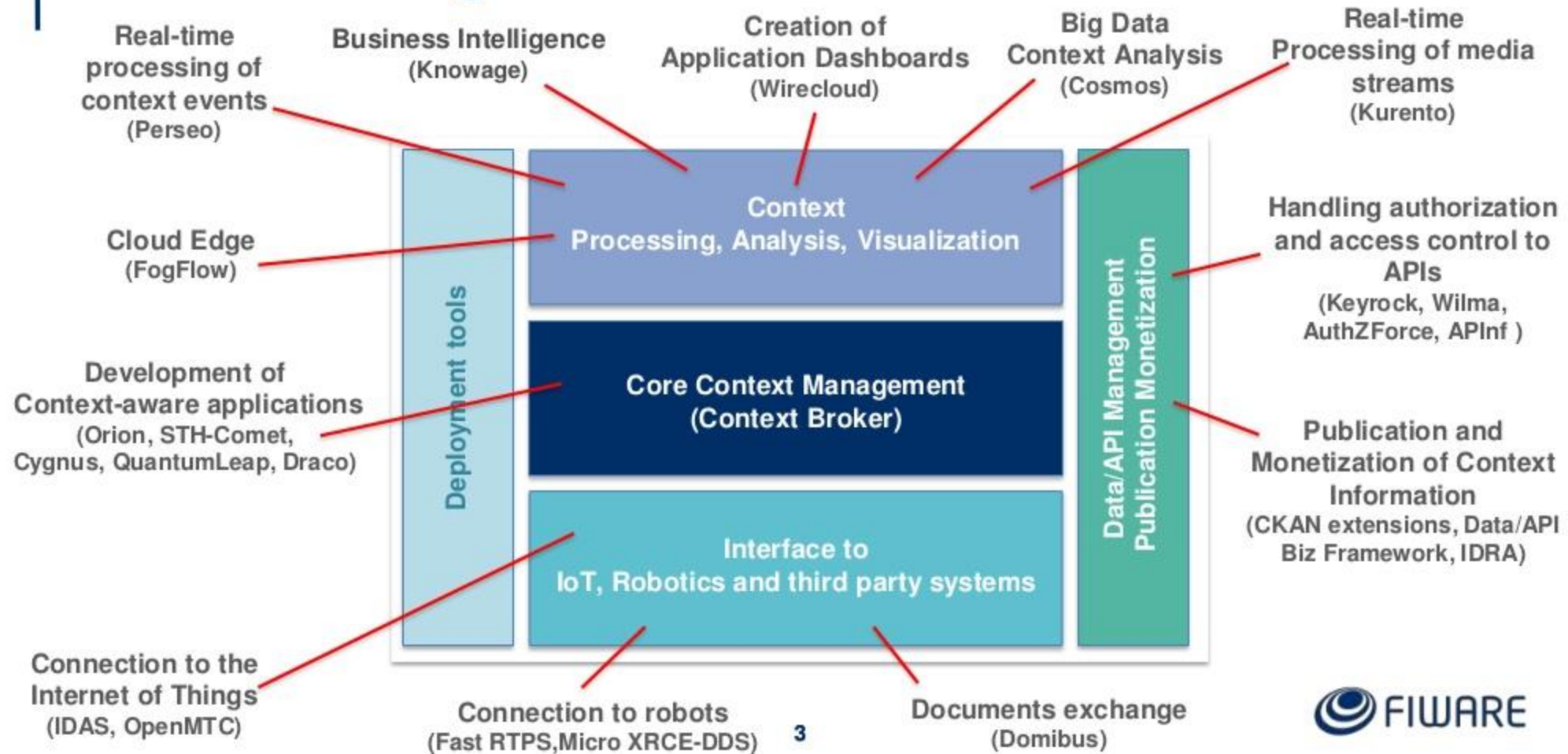
- Date
- Location
- Type
- Issuer
- Description



What is FIWARE?

- An **open-source initiative** defining a **universal set of standards for context data management** which facilitate the development of smart solutions for different domains such as smart cities, smart industry, smart agrifood and smart energy.
- Provides a **framework** of open-source **platform components** and **standards** to access and manage heterogeneous context information through **open APIs**.
- **Generic enablers** and solutions to provide smart services, with the **Context Broker** as the main component.
- A **standard** for exchange of context information:
NGSI (Next Generation Service Interface)

FIWARE Catalogue



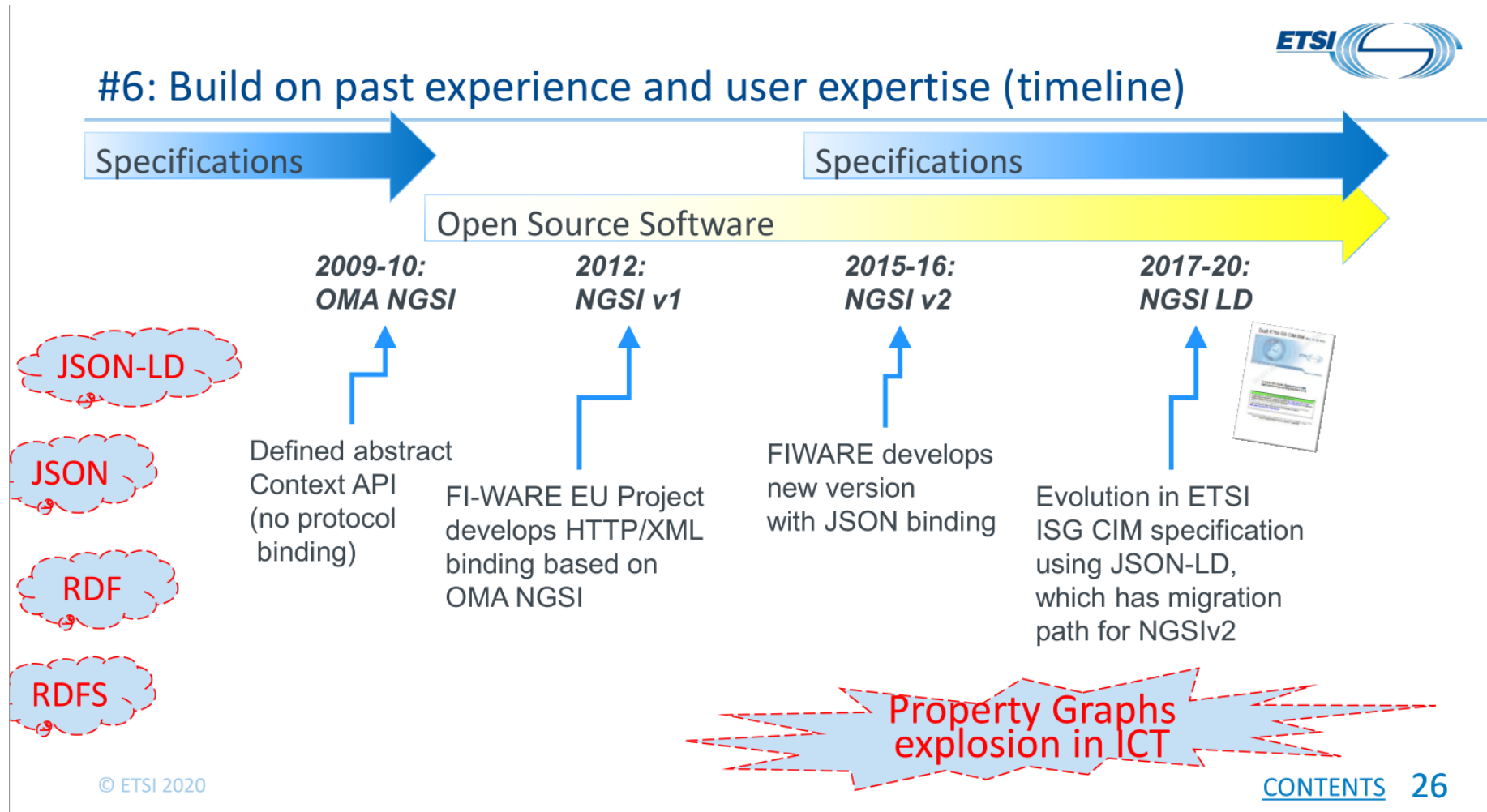
Context Broker

- The **Context Broker** is the **core** and **mandatory** component of any “**Powered by FIWARE**” platform or solution.
- Has been chosen as a **CEF (Connecting Europe Facility) Building Block** by all European member states.
<https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Context+Broker>
- It enables the system to perform **updates** and **access** the current state of the context.
- Only holds the **current state** – it has **no memory**.

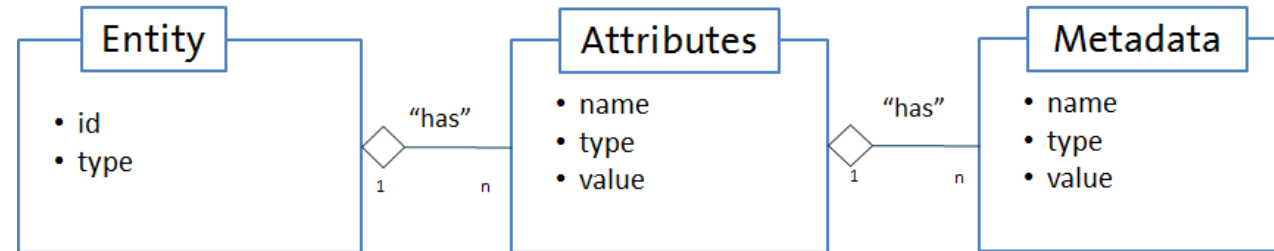
NGSI API

- **NGSI-LD** is the **formal standard API** for Context Information Management (**ETSI**).
https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.06.01_60/gs_CIM009v010601p.pdf
- All interactions between applications or platform components and the Context Broker take place using the **NGSI RESTful API**.
- Enables the **integration** of components and provides the basis for the **interoperability** and **portability** of smart solutions.

NGSI API Timeline



NGSI v2



Source: <https://fiware.github.io/specifications/ngsiv2/stable/>

- Each **entity** must have a unique **id** and **type** attribute.
- Additional **attributes** are optional.
- Each additional **attribute** should also have a defined **type** and a **value** attribute.
- **Metadata** is additional data used to describe properties of the attribute value itself (accuracy, provider, timestamp, etc.).
- **Relationships** can be defined using NGSI v2, but only so far as giving the attribute an appropriate attribute name (e.g., starting with `ref`, such as `refManagedBy`) and assigning the attribute `type=Relationship`.
This is purely a naming convention with no real semantic weight.

NGSI v2 Example

```
{
  "id": "entityId",
  "type": "entityType",
  "attr1": {
    "type": "attrType",
    "value": "attrValue",
    "metadata": {
      "metadata1": {
        "type": "metaType",
        "value": "metaValue"
      }
    }
  }
}
```

```
{
  "id": "device-9845A",
  "type": "Device",
  "batteryLevel": {
    "type": "Number",
    "value": 0.75
  },
  "controlledAsset": {
    "type": "Relationship",
    "value": [
      "wastecontainer-Osuna-100"
    ]
  },
  "value": {
    "type": "Text",
    "value": "l%3D0.22%3Bt%3D21.2"
  }
}
```

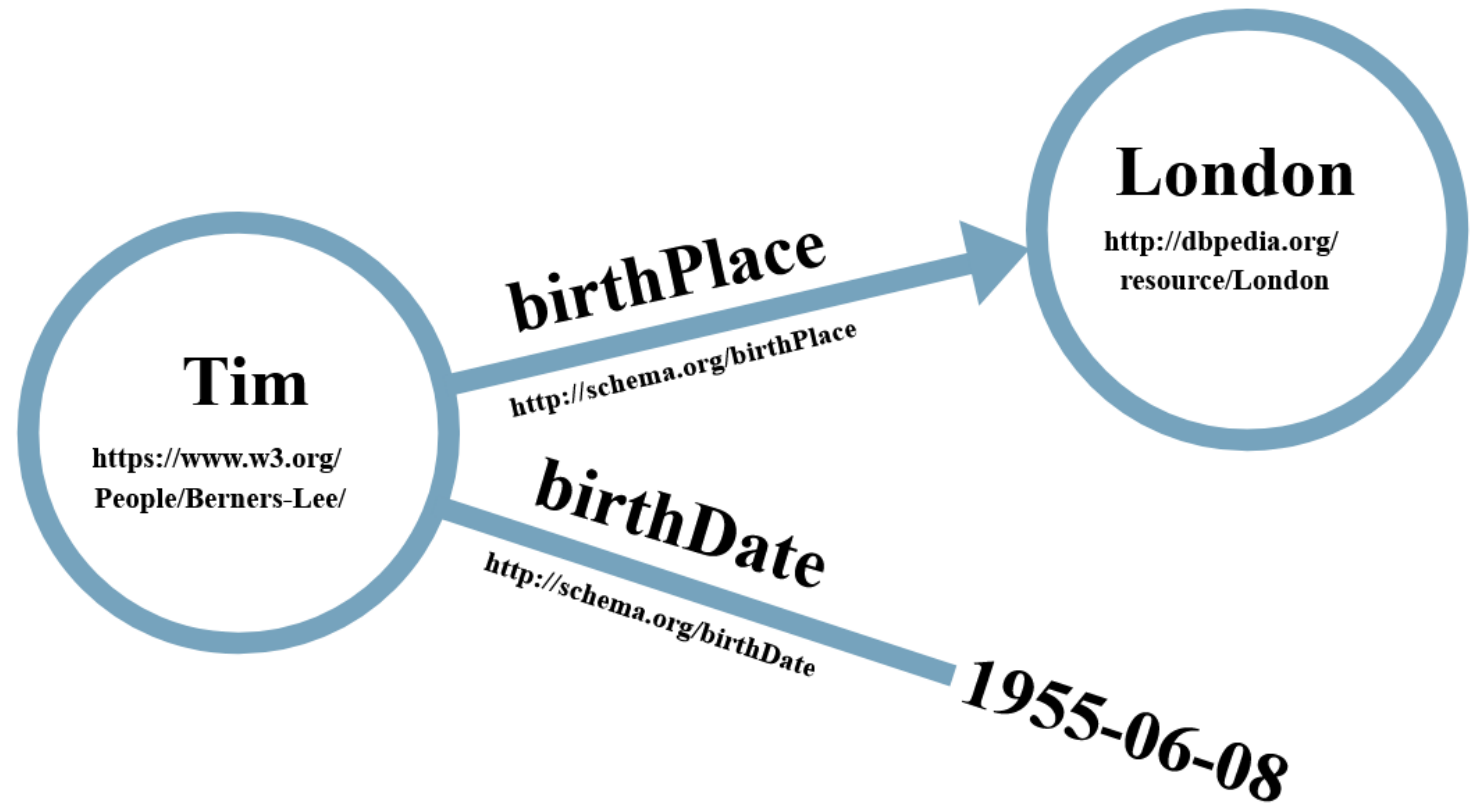
Source: <https://github.com/smart-data-models/dataModel.Device/blob/master/Device/doc/spec.md#device-ngsi-v2-normalized-example>

Linked Data

- **Linked Data** is a way to create a **network** of standards-based, **machine-readable data** across websites.
- Creating a system of readable links for computers **requires** the use of a **well-defined data format** (JSON-LD) and assignation of **unique IDs** (URLs or URNs) **for both data entities and the relationships between entities**, so that semantic meaning can be programmatically retrieved from the data itself.

Linked Data Example

Subject	Predicate	Object
Tim	birthPlace	London
Tim	birthDate	1955-06-08



Source: <https://ontola.io/what-is-linked-data/>

JSON-LD

- **JSON-LD** is a method of encoding **Linked Data** using JSON.
- It is a standard way of **avoiding ambiguity** when expressing Linked Data in JSON so that the data is structured in a format which is **parsable by machines**.
- **URLs and data models** are used to remove ambiguity by allowing attributes to have both a short form (such as name) and a fully specified long form (such as <http://schema.org/name>).
- JSON-LD is designed around the concept of a "**context**" which provides additional mappings, allowing the computer to interpret the data with more clarity and depth.
- The **@id keyword** can be used to give a node a URI. This URI identifies the node and can be used to reference it.
- The **@type keyword** can be used to associate a well-defined data model to the data itself.

JSON-LD Example

```
{
  "@context": "http://schema.org/",
  "@type": "Person",
  "address": {
    "@type": "PostalAddress",
    "addressLocality": "Seattle",
    "addressRegion": "WA",
    "postalCode": "98052",
    "streetAddress": "20341 Whitworth Institute 405 N. Whitworth"
  },
  "colleague": [
    "http://www.xyz.edu/students/alicejones.html",
    "http://www.xyz.edu/students/bobsmith.html"
  ],
  "email": "mailto:jane-doe@xyz.edu",
  "image": "janedoe.jpg",
  "jobTitle": "Professor",
  "name": "Jane Doe",
  "telephone": "(425) 123-4567",
  "url": "http://www.janedoe.com"
}
```

```
{
  "@type": "http://schema.org/Person",
  "http://schema.org/address": {
    "@type": "http://schema.org/PostalAddress",
    "http://schema.org/addressLocality": "Seattle",
    "http://schema.org/addressRegion": "WA",
    "http://schema.org/postalCode": "98052",
    "http://schema.org/streetAddress": "20341 Whitworth Institute 405 N. Whitworth"
  },
  "http://schema.org/colleague": [
    {
      "@id": "http://www.xyz.edu/students/alicejones.html"
    },
    {
      "@id": "http://www.xyz.edu/students/bobsmith.html"
    }
  ],
  "http://schema.org/email": "mailto:jane-doe@xyz.edu",
  "http://schema.org/image": {
    "@id": "janedoe.jpg"
  },
  "http://schema.org/jobTitle": "Professor",
  "http://schema.org/name": "Jane Doe",
  "http://schema.org/telephone": "(425) 123-4567",
  "http://schema.org/url": {
    "@id": "http://www.janedoe.com"
  }
}
```

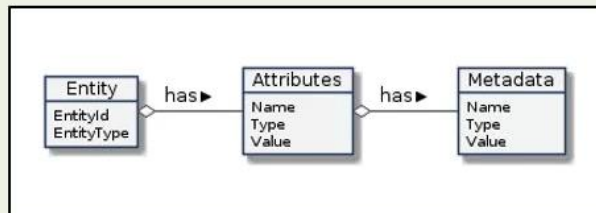
NGSI-LD

- **NGSI-LD** is an evolution of the **NGSI v2**, which has been modified to improve support for **Linked Data** by exploiting the capabilities offered by JSON-LD.
- Creating proper **machine-readable Linked Data** is fundamental to NGSI-LD.

NGSI-LD Data Model

The NGSI LD data model is more complex; the definitions of use are more rigid which lead to a navigable knowledge graph.

NGSI v2

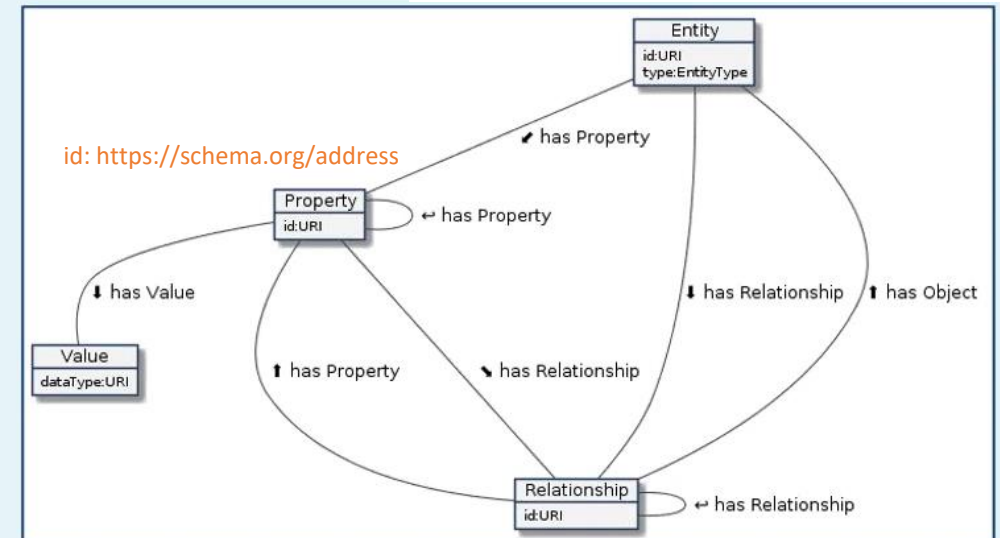


- Entities
- Attributes
- MetaData

NGSI-LD

- Entities
- Properties
- Relationships
- Values

plus ...



id: urn:ngsi-ld:Building:store001
type: https://uri.fiware.org/ns/data-models#Building

managedBy
urn:ngsi-ld:Person:bob-the-manager

- Properties of Properties
- Properties of Relationships
- Relationships of Properties
- Relationships of Relationships

plus ...

- Properties of Properties of Properties
- Relationships of Properties of Properties
- Properties of Properties of Relationships
- Relationships of Properties of Relationships
- Properties of Relationships of Properties
- Relationships of Relationships of Properties
- Properties of Relationships of Relationships
- Relationships of Relationships of Relationships

etc...

NGSI-LD Data Model

The Entity	Example	Notes
Has an id	<code>urn:ngsi-ld:Building:store001</code>	URI/URN. id must be unique.
Has a type .	<code>https://uri.fiware.org/ns/data-models#Building</code>	<ul style="list-style-type: none">Fully qualified URI of a well defined data modelShort-hand strings for types, mapped to fully qualified URIs through the JSON-LD @context.
Has a series of properties	name, address, category etc.	This can be expanded into <code>http://schema.org/address</code> , which is known as a fully qualified name (FQN).
Has a series of properties-of-properties	a verified field for the address	This is the equivalent of NGSI v2 metadata
Has a series of relationships	managedBy	The object corresponds to the URI/URN of another data entity. Equivalent of NGSI v2 refXXX
Has a series of properties-of-relationships	managedBy.since	Holds additional information about a relationship. This is the equivalent of metadata about a refXXX property
Has a series of relationships-of-relationships	managedBy.subordinateTo	holds the URI/URN of another relationship.

Source: <https://www.slideshare.net/FI-WARE/fiware-wednesday-webinars-introduction-to-ngsild-236464309>

NGSI v2 Example	NGSI-LD Example
<pre>{ "id": "device-9845A", "type": "Device", "batteryLevel": { "type": "Number", "value": 0.75 }, "controlledAsset": { "type": "Relationship", "value": ["wastecontainer-Osuna-100"] }, "value": { "type": "Text", "value": "I%3D0.22%3Bt%3D21.2" } }</pre>	<pre>{ "id": "urn:ngsi-ld:Device:device-9845A", "type": "Device", "batteryLevel": { "type": "Property", "value": 0.75 }, "controlledAsset": [{ "type": "Relationship", "object": "urn:ngsi-ld:wastecontainer-Osuna-100" }], "value": { "type": "Property", "value": "I%3D0.22%3Bt%3D21.2" }, "@context": ["https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld", "https://raw.githubusercontent.com/smart-data-models/dataModel.Device/master/context.jsonld"] }</pre>

Smart Data Models

- Although each data entity within the context will vary according to the use case, the **common structure** within each data entity **should be standardized** to promote **reusability**.
- The **FIWARE Foundation** and **TM Forum** are leading a joint collaboration program to support the adoption of a **reference architecture** and compatible **common data models** that underpin a digital market of **interoperable and replicable smart solutions**.
- Available at: <https://github.com/smart-data-models/>

Core Context Management: Hello World!

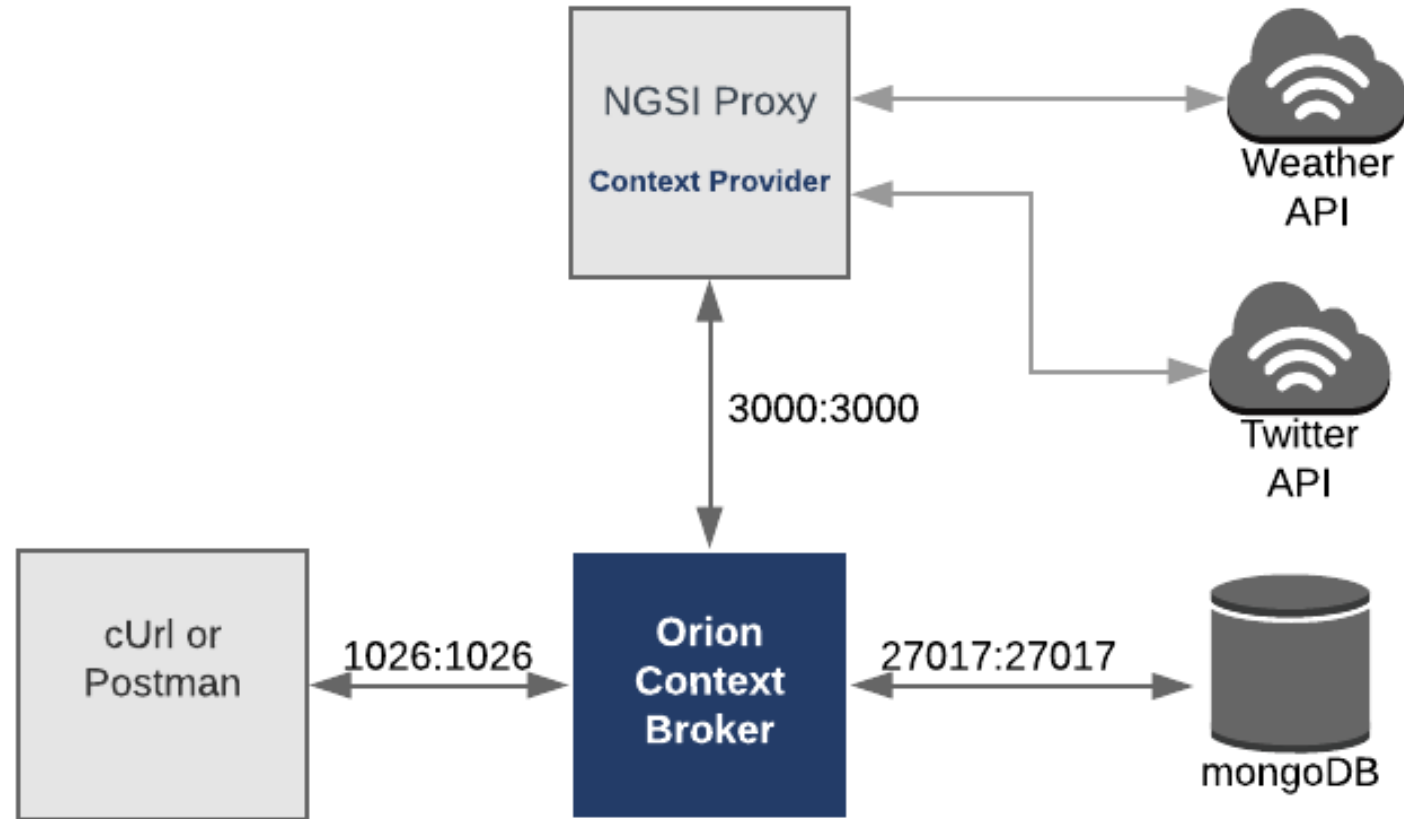


Source: <https://github.com/FIWARE/tutorials.Getting-Started/tree/NGSI-v2>

Context Providers

- There is another class of **context data** which is much more **dynamic** (temperature, humidity etc.).
- This information is **always changing** and if it were statically held in a database, the data would always be out-of-date.
- The FIWARE platform makes the gathering and presentation of real-time context data transparent. Each NGSI request will return the latest context by combining the data held within its database along with real-time data readings from any registered external **context providers**.

Context Providers



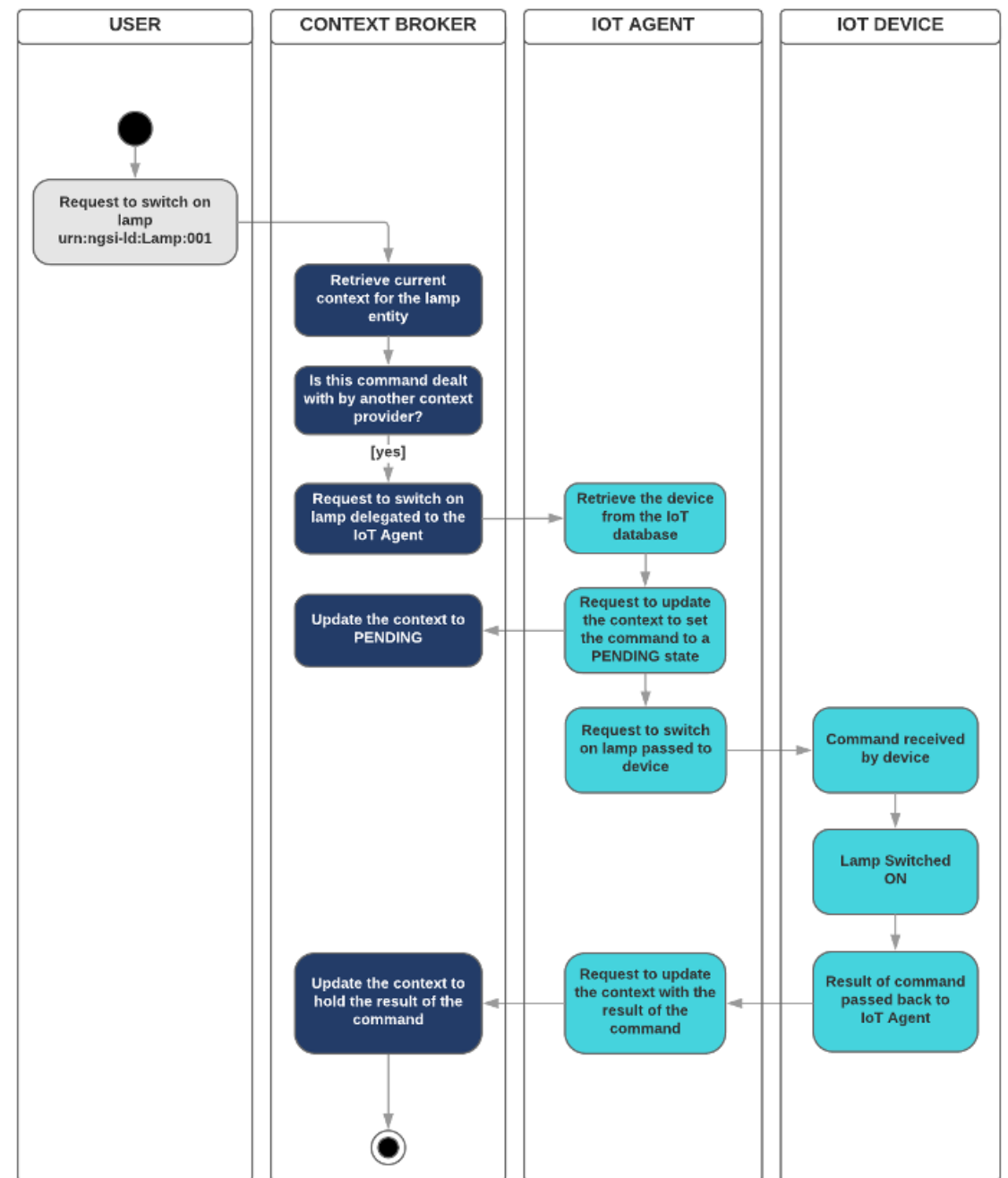
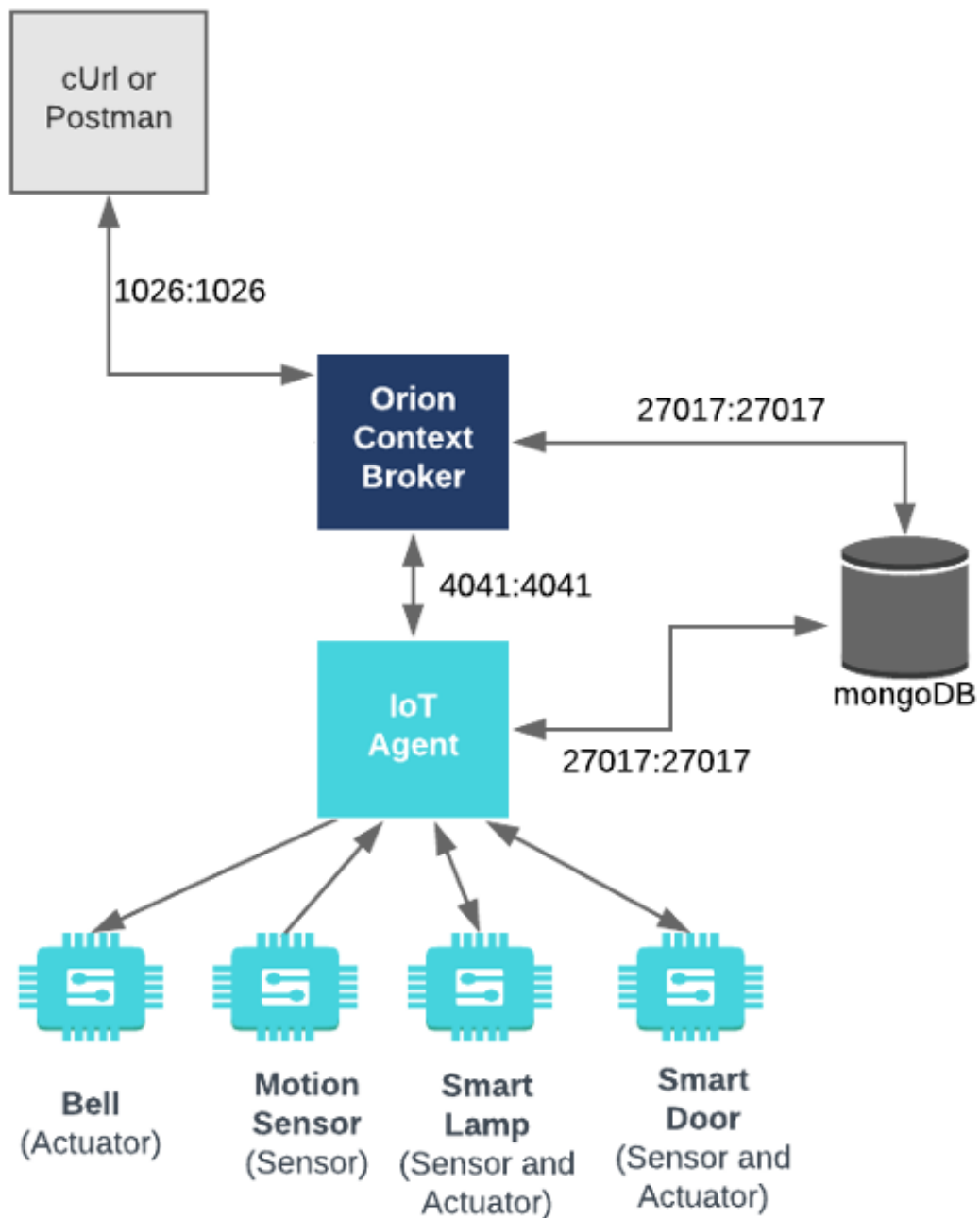
Source: <https://github.com/FIWARE/tutorials.Context-Providers/tree/NGSI-v2>

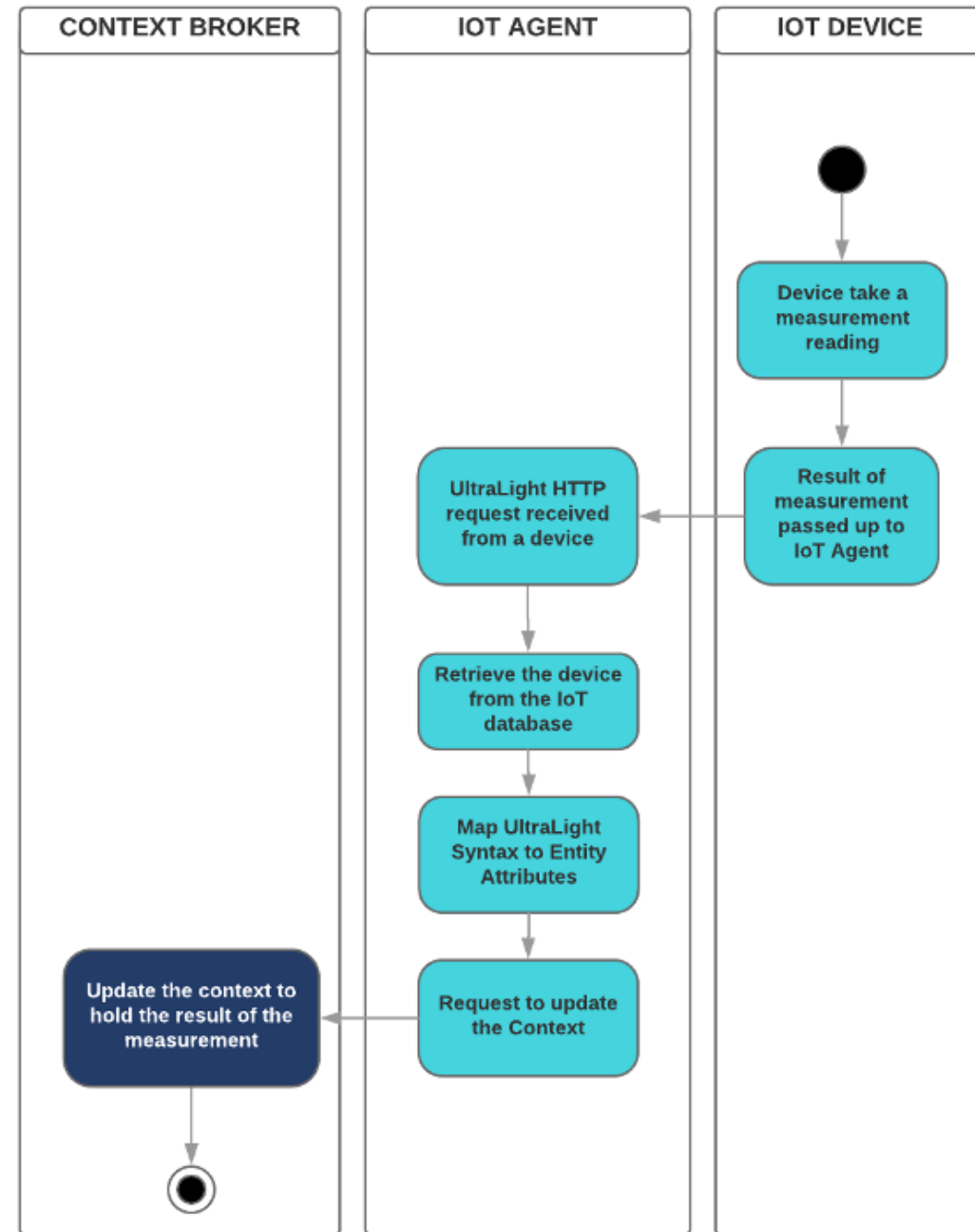
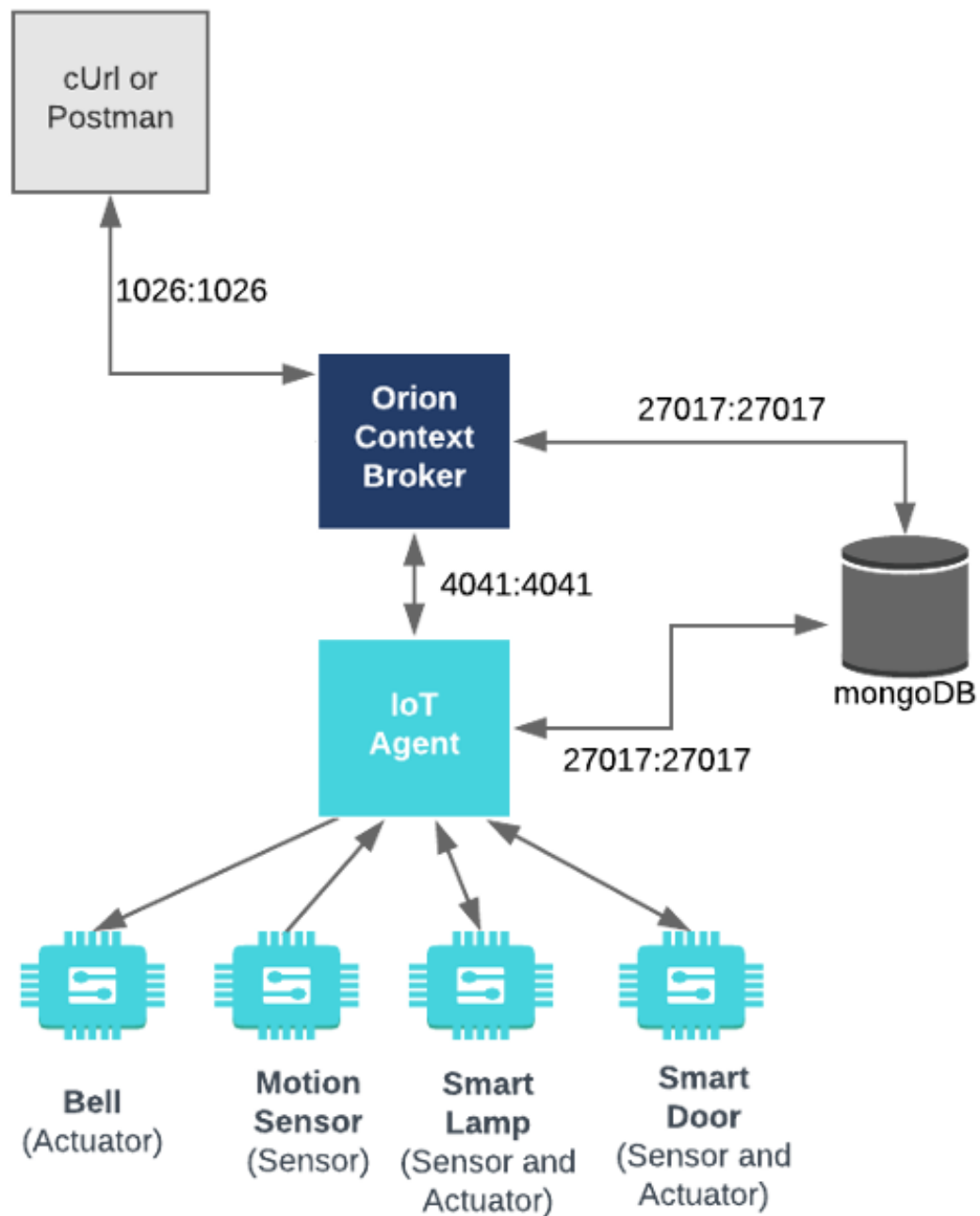
Subscriptions

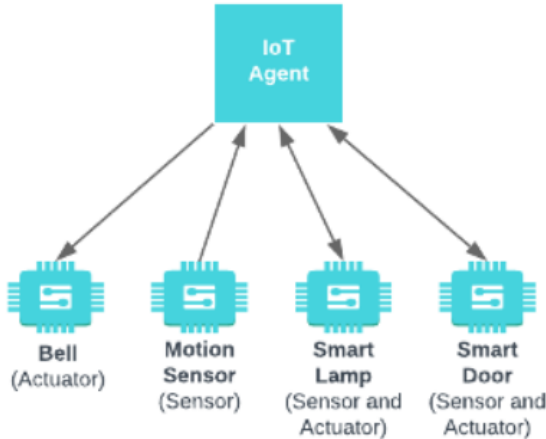
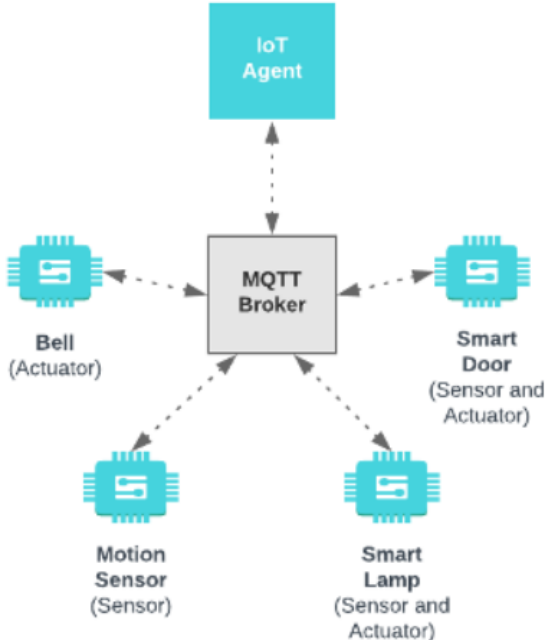
- Until now all the operations used to change the state of the system have been synchronous. Changes have been made directly by a user or application and they have been informed of the result.
- The **Context Broker** offers an **asynchronous notification mechanism**. Applications can subscribe to changes of context information so that they can be informed when something happens. This means the application does not need to continuously poll or repeat query requests.

IoT Agents

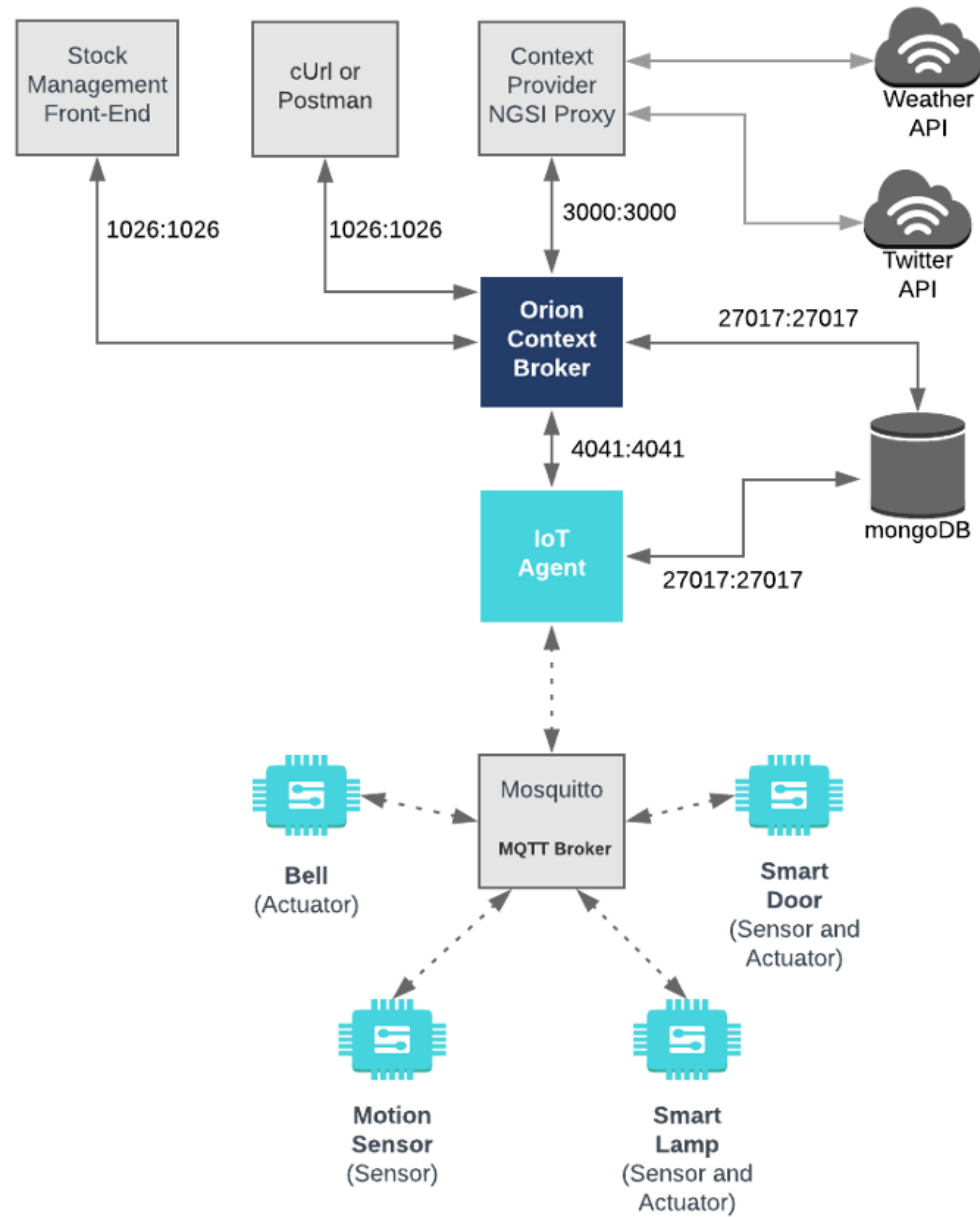
- An **IoT Agent** is a component that lets a group of devices interact with a Context Broker using their own native protocols.
- Each IoT Agent provides a **north port NGSI interface** which is used for Context Broker interactions. All interactions beneath this port occur using the native protocol of the attached devices.
- This brings a **standard interface** to all IoT interactions at the context information management level.
- Already implemented IoT Agents include the following:
 - **IoTAgent-JSON** - a bridge between **HTTP/MQTT** messaging (with a **JSON** payload) and **NGSI**
 - **IoTAgent-LWM2M** - a bridge between the **Lightweight M2M** protocol and **NGSI**
 - **IoTAgent-UL** - a bridge between **HTTP/MQTT** messaging (with an **UltraLight2.0** payload) and **NGSI**
 - **IoTAgent-LoRaWAN** - a bridge between the **LoRaWAN** protocol and **NGSI**





HTTP Transport	MQTT Transport
 <p>The diagram shows an IoT Agent at the top, connected by solid arrows to four IoT devices below: Bell (Actuator), Motion Sensor (Sensor), Smart Lamp (Sensor and Actuator), and Smart Door (Sensor and Actuator). This represents a direct, request-response communication model.</p>	 <p>The diagram shows an IoT Agent at the top, connected by a dashed arrow to a central MQTT Broker. The MQTT Broker is then connected by dashed arrows to four IoT devices: Bell (Actuator), Motion Sensor (Sensor), Smart Lamp (Sensor and Actuator), and Smart Door (Sensor and Actuator). This represents an indirect, publish-subscribe communication model.</p>
IoT Agent communicates with IoT devices directly	IoT Agent communicates with IoT devices indirectly via an MQTT Broker
Request-Response Paradigm	Publish-Subscribe Paradigm
IoT Devices must always be ready to receive communication	IoT Devices choose when to receive communication
Higher Power Requirement	Low Power Requirement

Source: <https://github.com/FIWARE/tutorials.ioT-over-MQTT/tree/NGSI-v2>



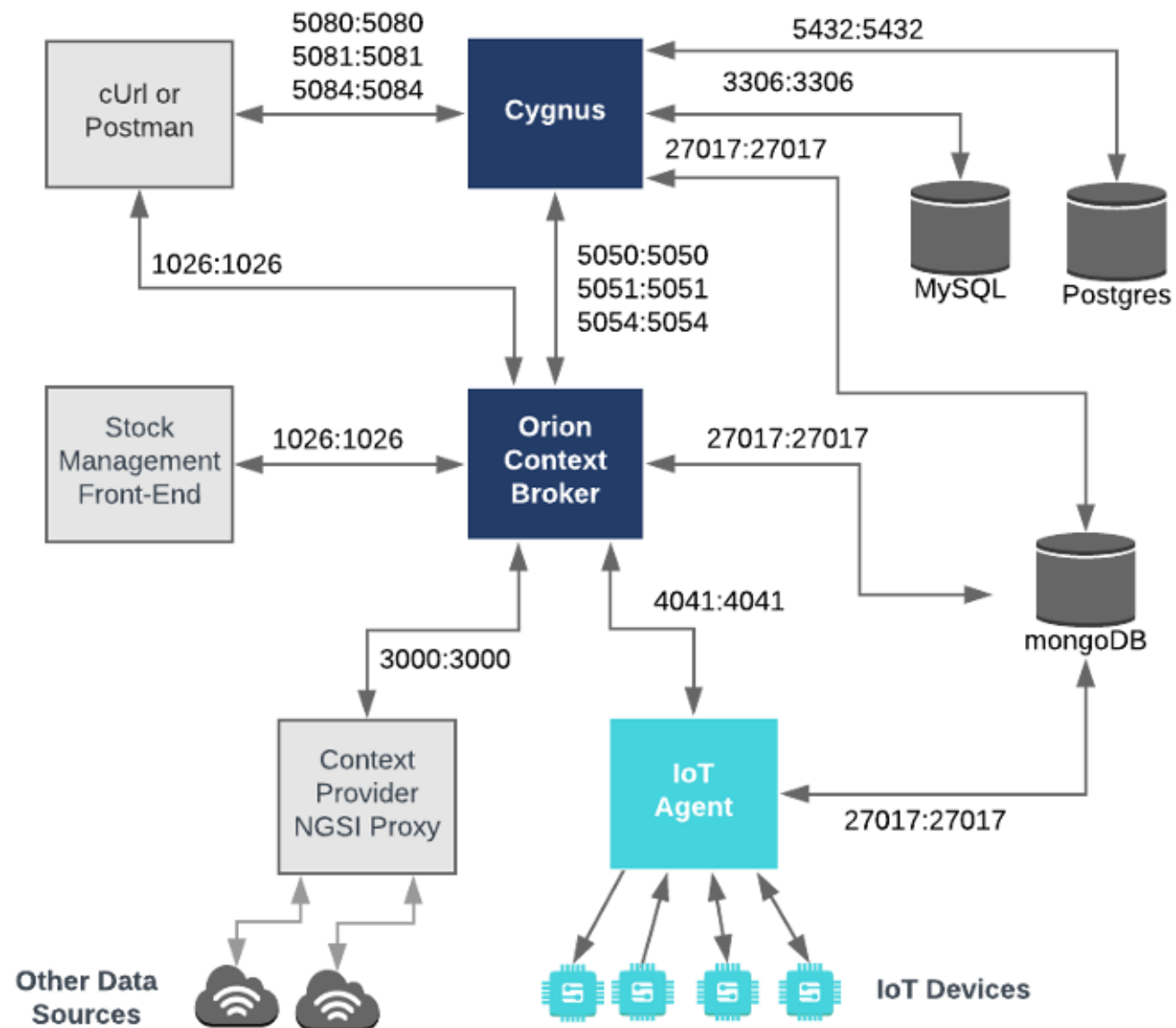
Source: <https://github.com/FIWARE/tutorials.ioT-over-MQTT/tree/NGSI-v2>

Data Persistence

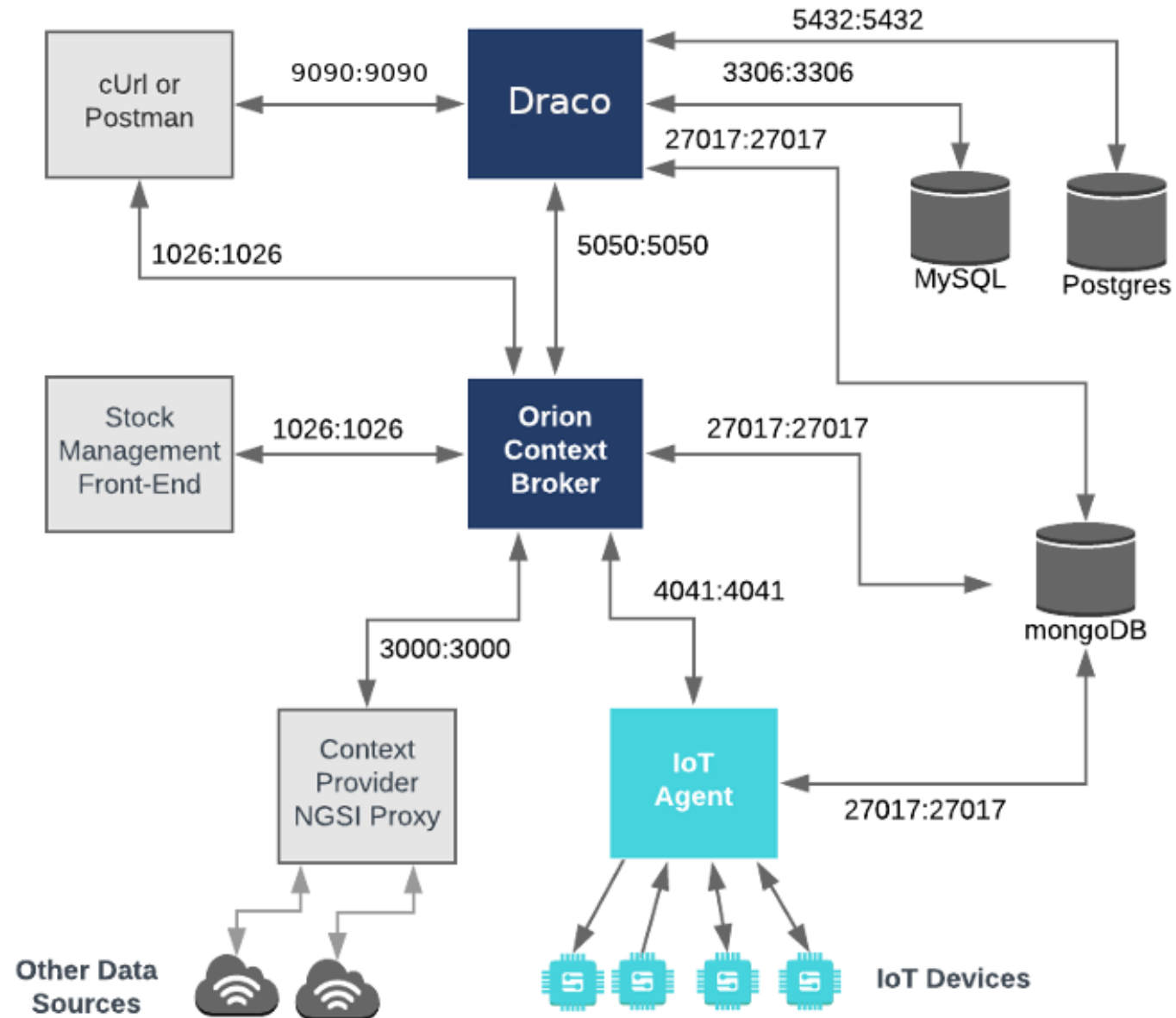
- **Context Brokers** only hold the **latest value** of context data and **should not be used as databases**.
- **Data persistence** gives "**memory**" to context data.
- Since business **requirements differ** from application to application, there is **no one standard use case for historical data persistence**.
- Therefore, rather than overloading the Context Broker with the job of historical context data persistence, this role has been assigned to **separate, highly configurable components**.

Data Persistence

- The **FIWARE Catalogue** contains two components to facilitate data persistence:
 - **Cygnus** – based on **Apache Flume**
 - **Draco** – based on **Apache NIFI**
- **Cygnus** is the **older** and **more mature** data persistence component within the FIWARE Catalogue.
- **Draco** offers a **graphical interface** to set up and monitor the procedure.



Source: <https://github.com/FIWARE/tutorials.Historic-Context-Flume>



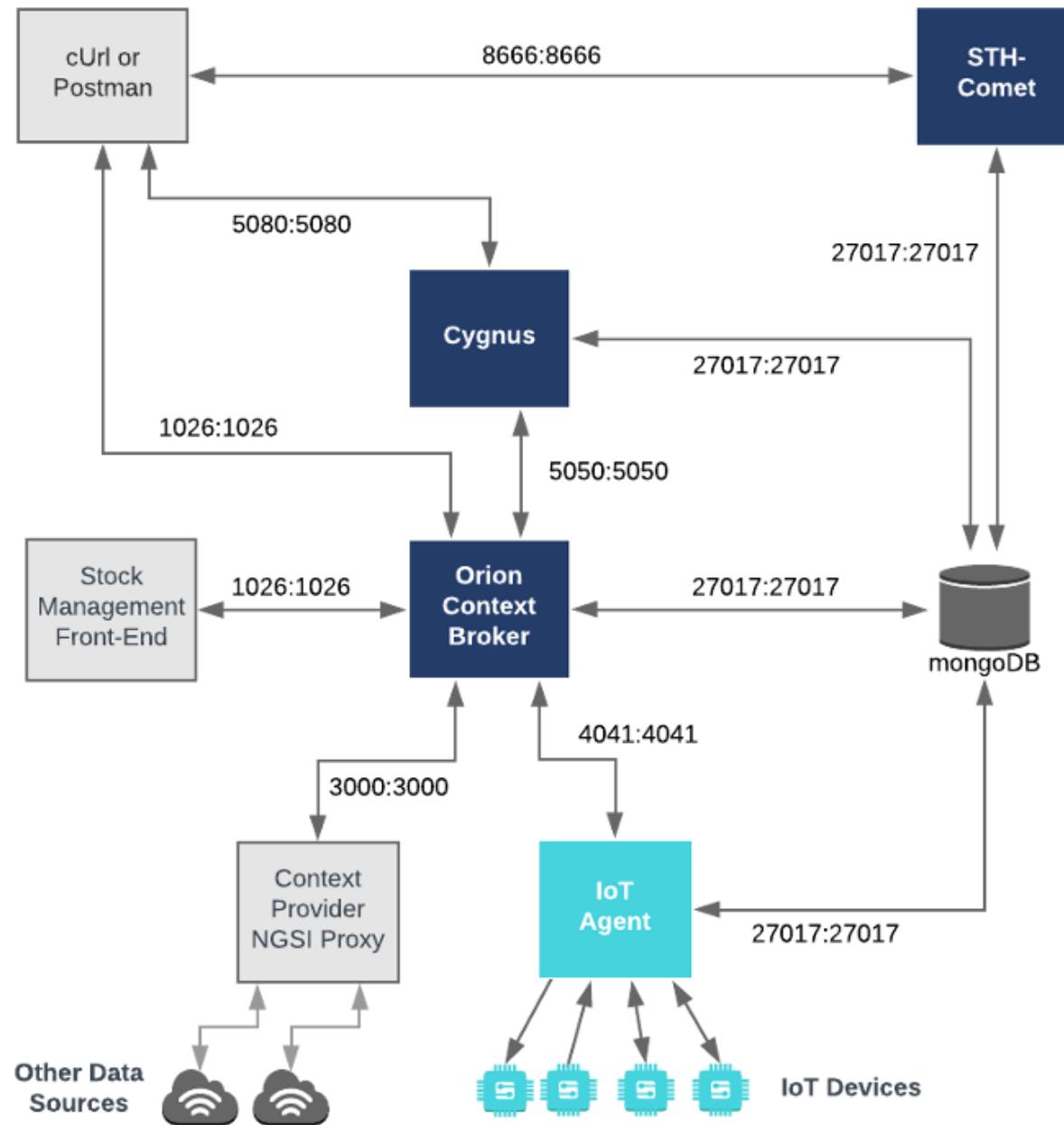
Source: <https://github.com/FIWARE/tutorials.Historic-Context-NIFI>

Short Term History

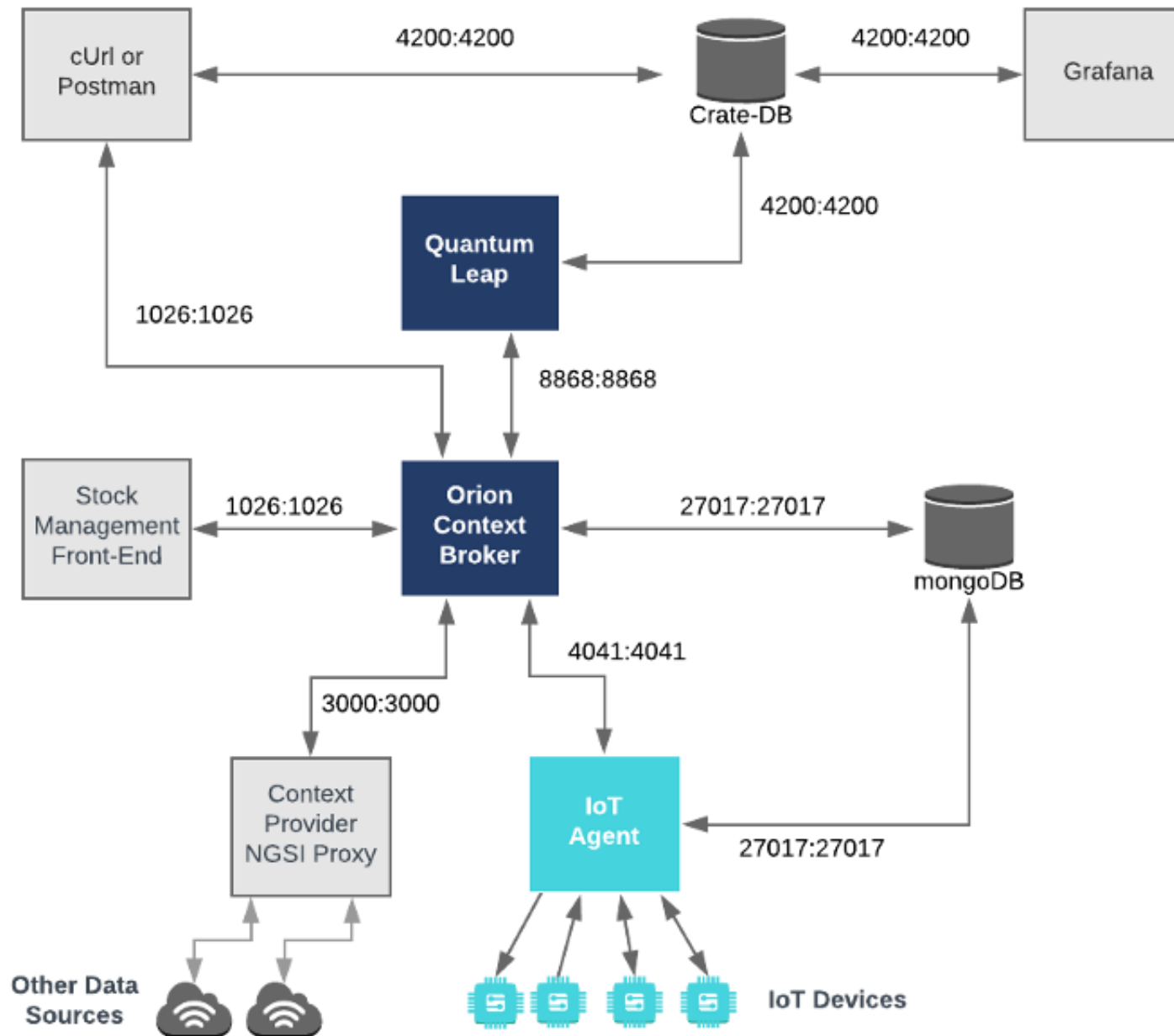
- **Historical context data** can be persisted to a database creating a **series of data points**.
- Each time-stamped data point represents the state of context entities **at a given moment in time**.
- The individual data points are relatively meaningless on their own. It is only through **combining** a series of data points that **meaningful statistics** such as maxima, minima and trends can be observed.

Short Term History

- The **FIWARE Catalogue** contains two components to deal with short term history:
 - **STH-Comet**
 - **QuantumLeap**
- Each component places **buckets** of time-series data into a database and offers retrieval via an API.
- Results can be displayed using **graphing tools**, such as Grafana.

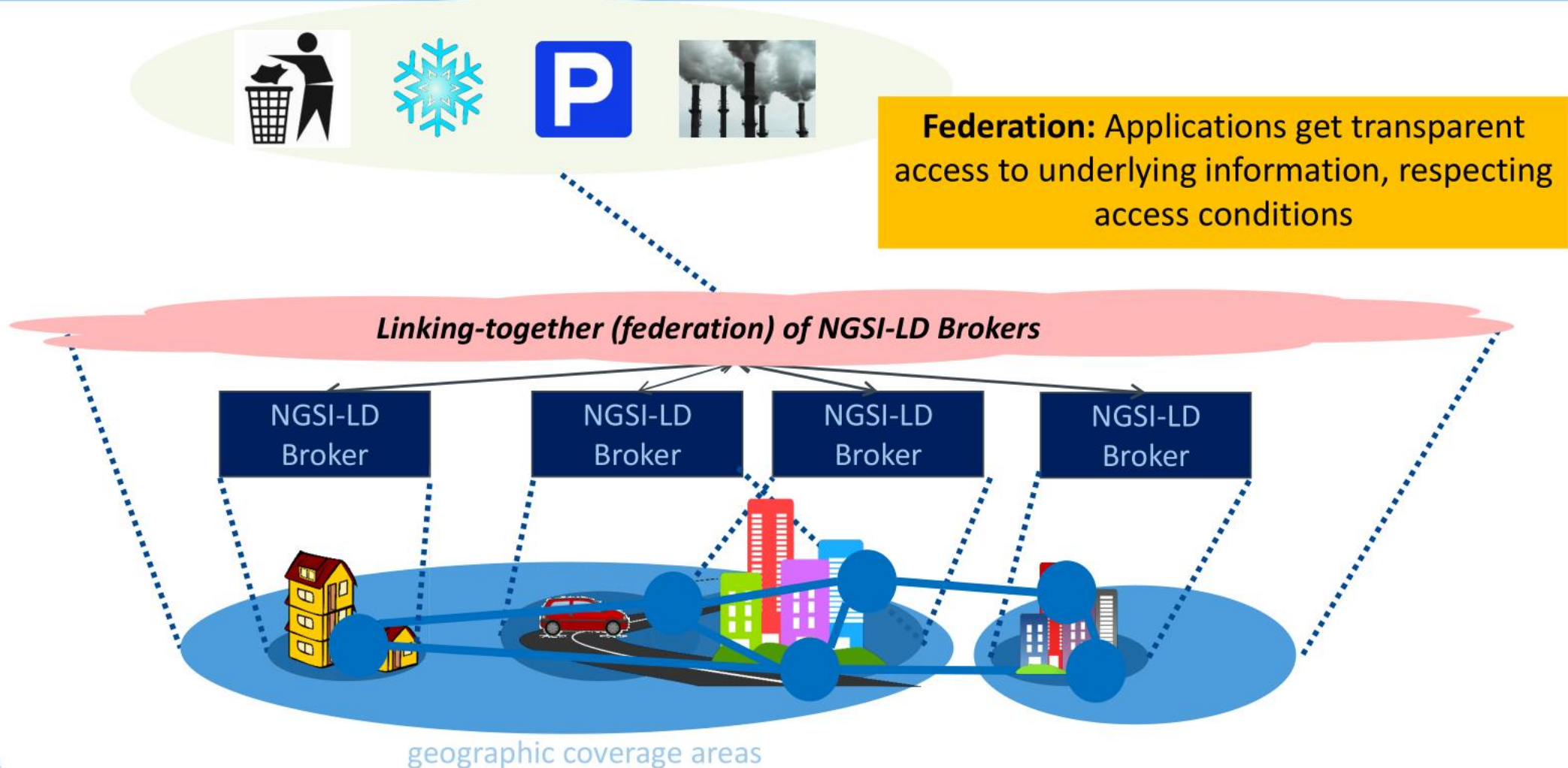


Source: <https://github.com/FIWARE/tutorials.Short-Term-History/tree/NGSI-v2>

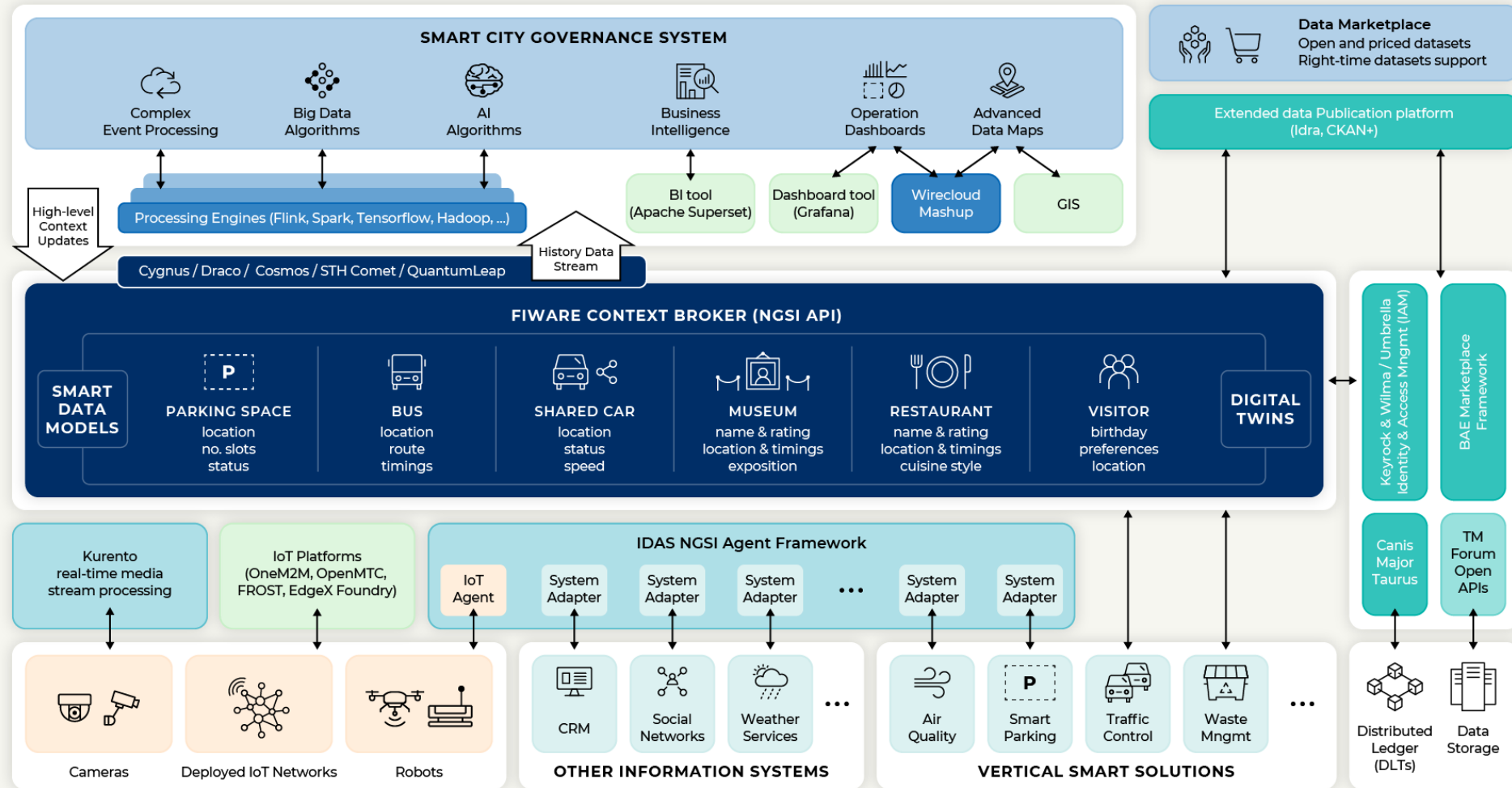


Source: <https://github.com/FIWARE/tutorials.Time-Series-Data/tree/NGSI-v2>

#4: Adapt to Local, Central or Federated Sources



The FIWARE Smart Cities Reference Architecture



FIWARE: Standardization for Smart Cities on a global scale



ETSI published on January 24th, 2019 "**NGSI-LD**" the new Context Information Management Standard API. The rationale is to reinforce the fact that this specification leverages on the ... **FIWARE NGSIv2** to incorporate the latest advances from Linked Data.



Joint Collaboration Program:
Front-runner Smart Cities

- to support the adoption of a reference architecture and compatible common data models
- Using FIWARE NGSI and TM Forum Open APIs
- Smart City Common Data Models will be public and royalty-free
- Initial cities: Vienna, Nice, Genoa, Utrecht, Porto, Santander, Valencia, Gothenburg, La Plata, Montevideo



FIWARE Context Broker

Technology has been chosen in 2018 as new CEF (Connecting Europe Facility) Building Block by all European member states.

Existing CEF Building Blocks so far:

- eDelivery
- eInvoicing
- eID
- eSignature
- eTranslation.