Systems of Systems (SoS)

M.Sc. In Critical Computing Systems Engineering

ISEP/IPP – 2021/22, 2nd semester

FIWARE

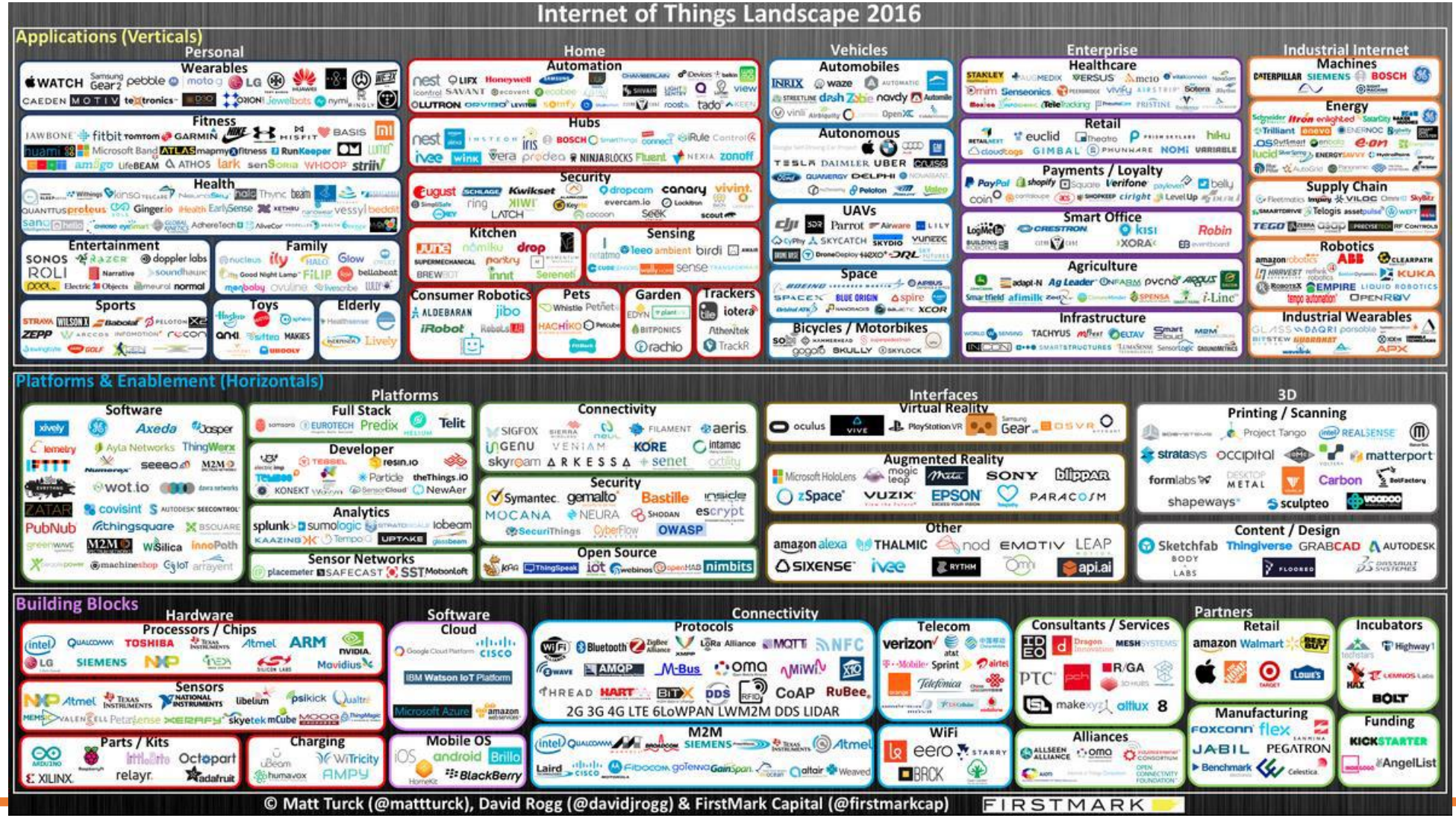
Pedro Santos

Outline

- What is FIWARE?
- Publisher-Subscriber Paradigm in FIWARE
- NGSI-LD

What is FIWARE?

Wide Range of IoT Standards



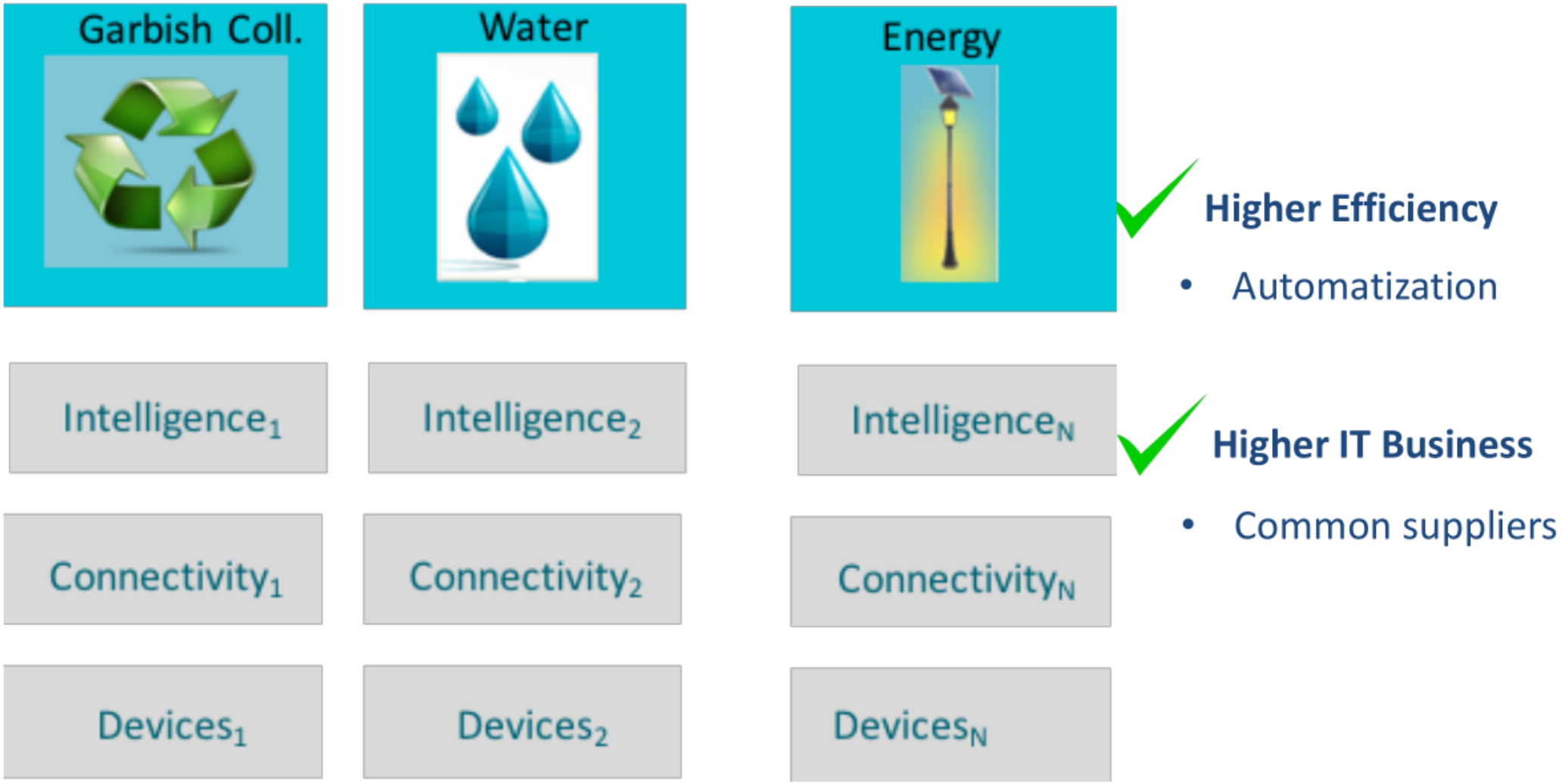
FIWARE

- Why do we need FIWARE?
  - Gathering, publishing, processing and analyzing private and open data at large scale
- What is it?
  - A curated framework of Open Source Platform components to accelerate the development of Smart Solutions
  - Advanced OpenStack-based cloud + rich library of Generic Enablers (GEs)
- Big push by European Commission towards promoting openness and sharing of data
- Built around the concept of CONTEXT!

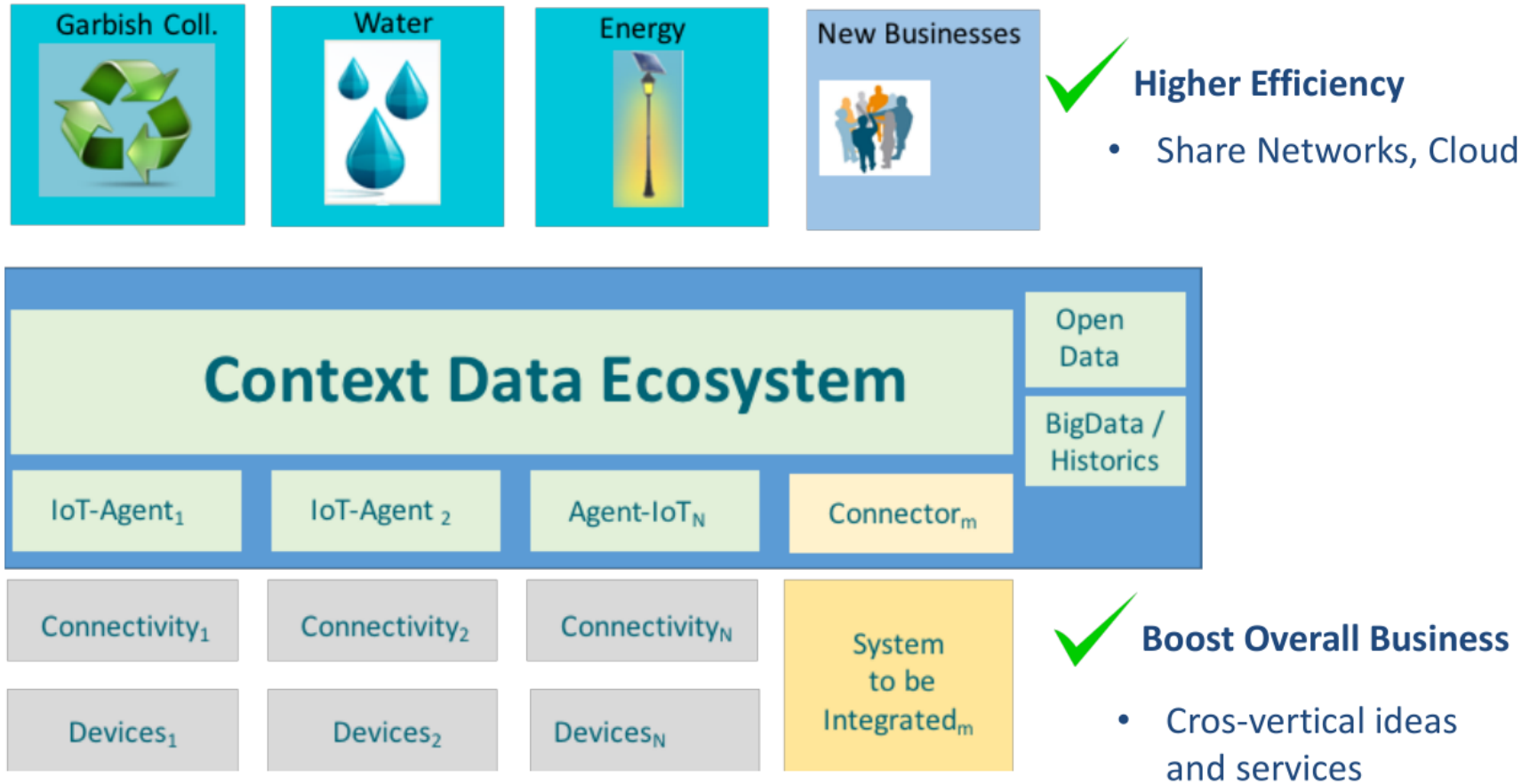
Context


APPs / Services / Data Scientist
NGSI API
Context Broker
Boiler
• Manufacturer
• Last revision
• Product id
• Temperature
• Actions
Users
• Name-Surname
• Birthday
• Location
• ToDo list
City
• OpenData
• Users Input
Street Devices
• Location
• Observations
• Commands
Public Bus T.System
• Location
• Arrival time

Previously: Silos or Verticals



FIWARE: Growth Engine for Local Ecosystems



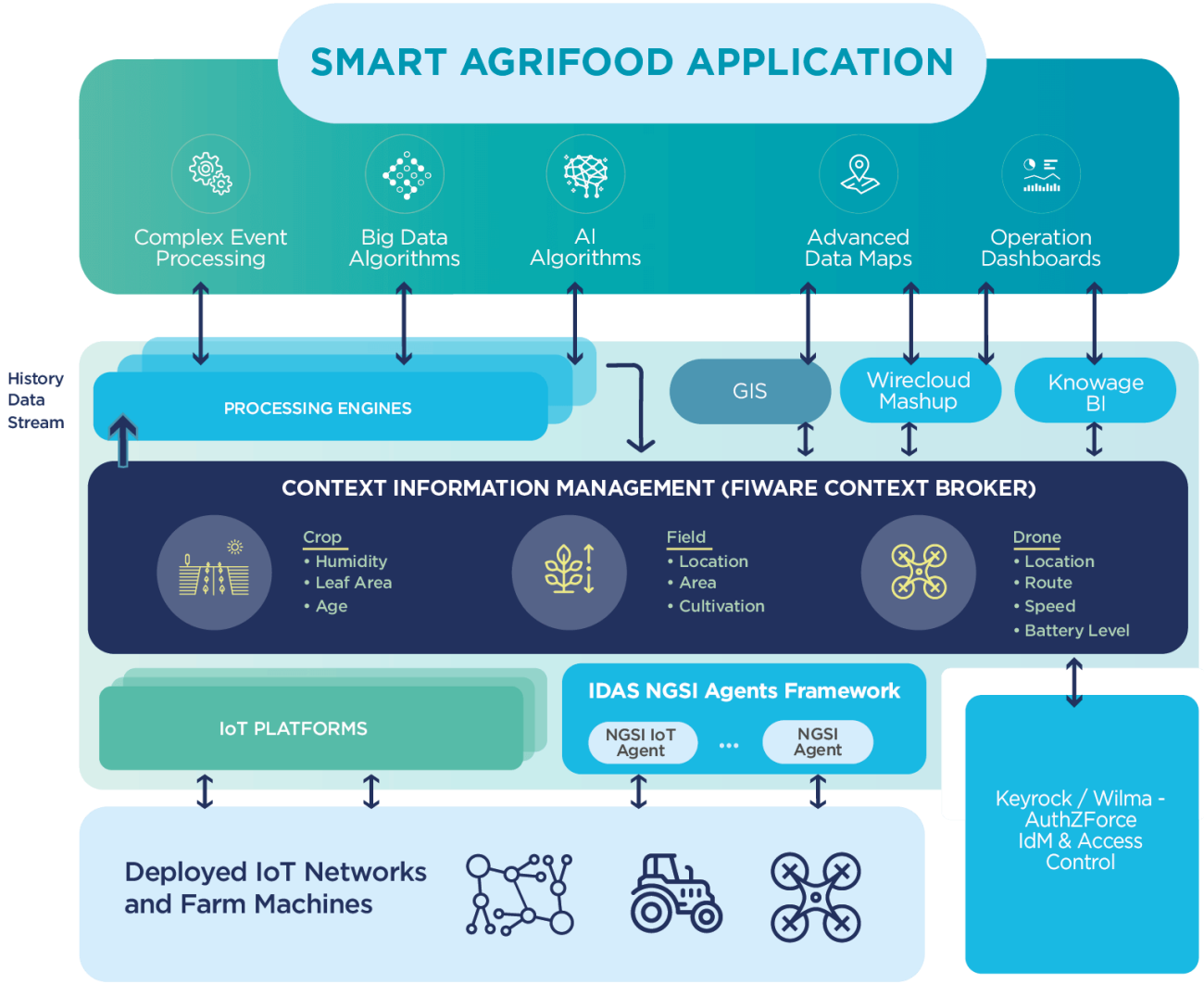
Higher Efficiency

- Share Networks, Cloud

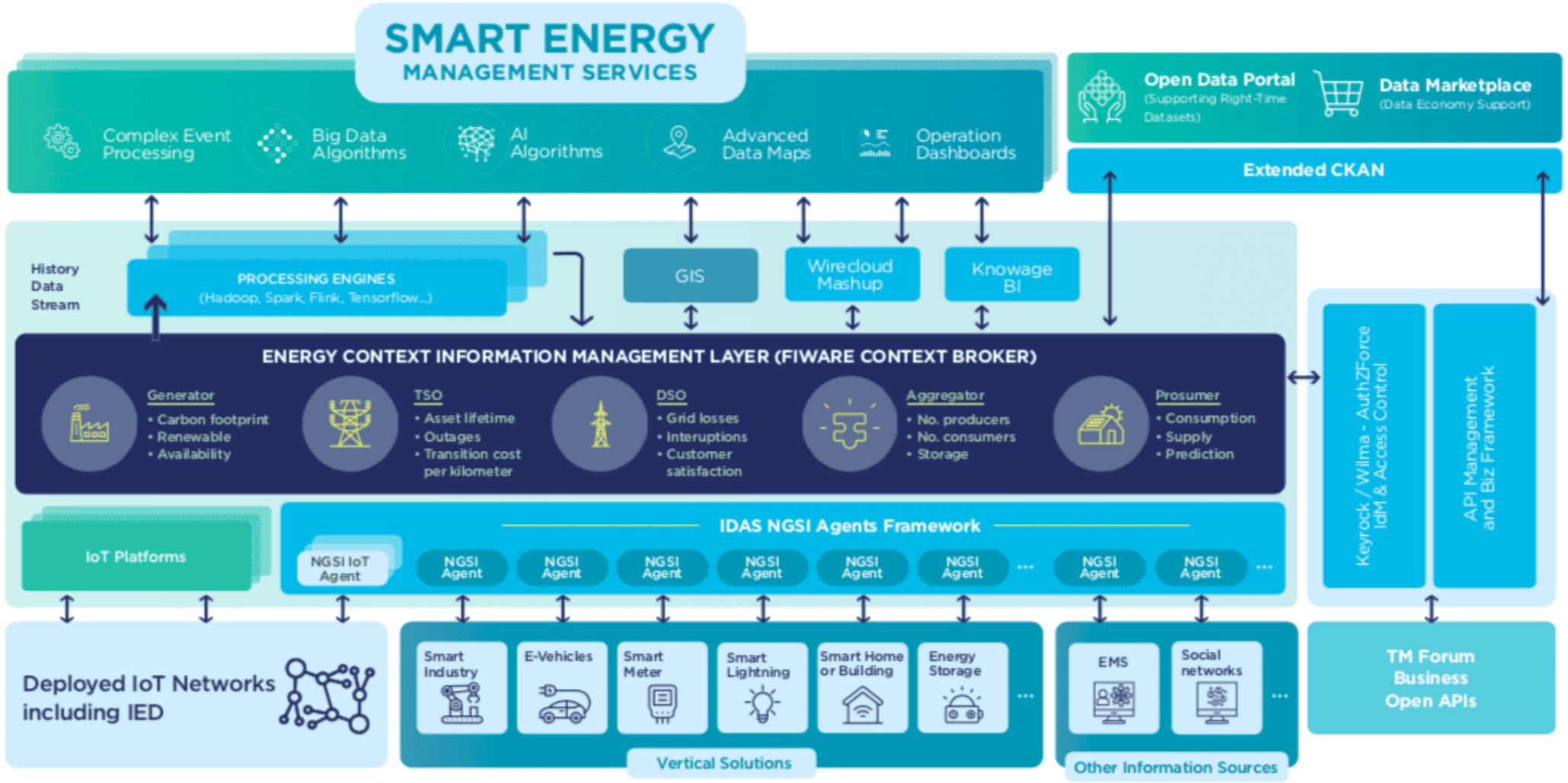
Boost Overall Business

- Cros-vertical ideas and services
- New Players
- Data Scientists
- Developers

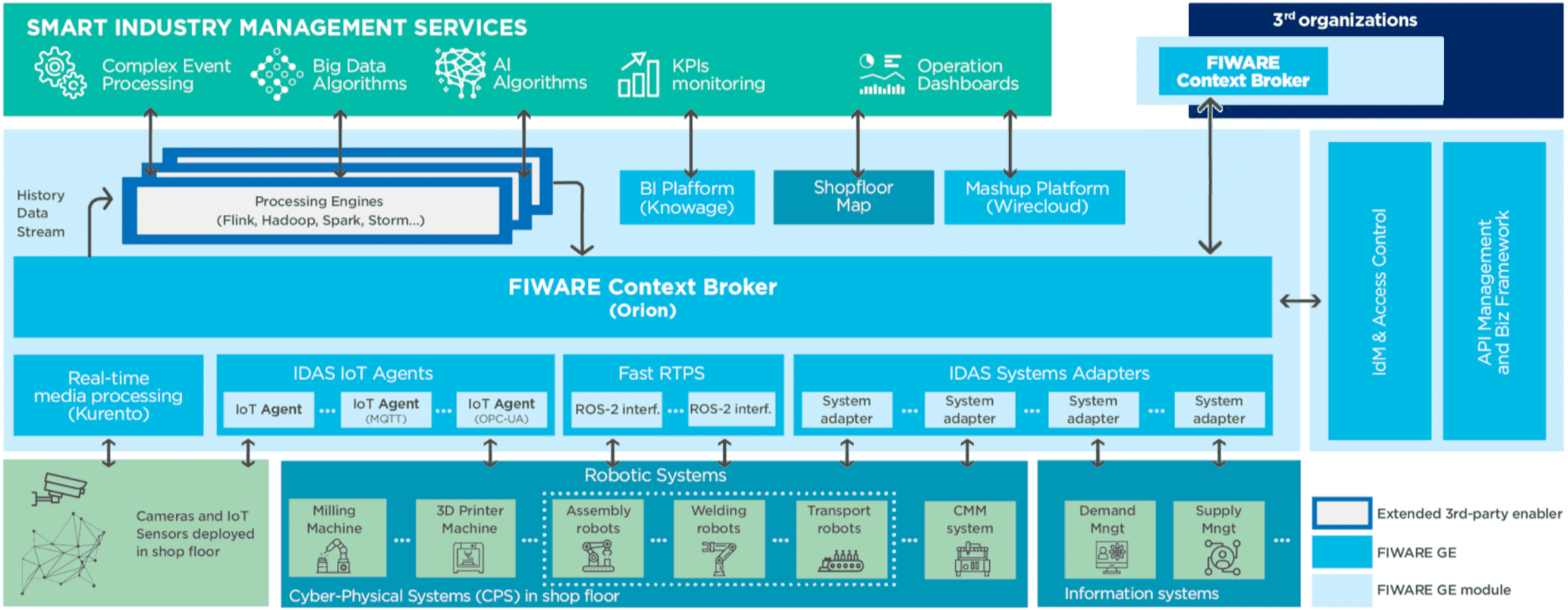
Smart Agrifood



Smart Energy

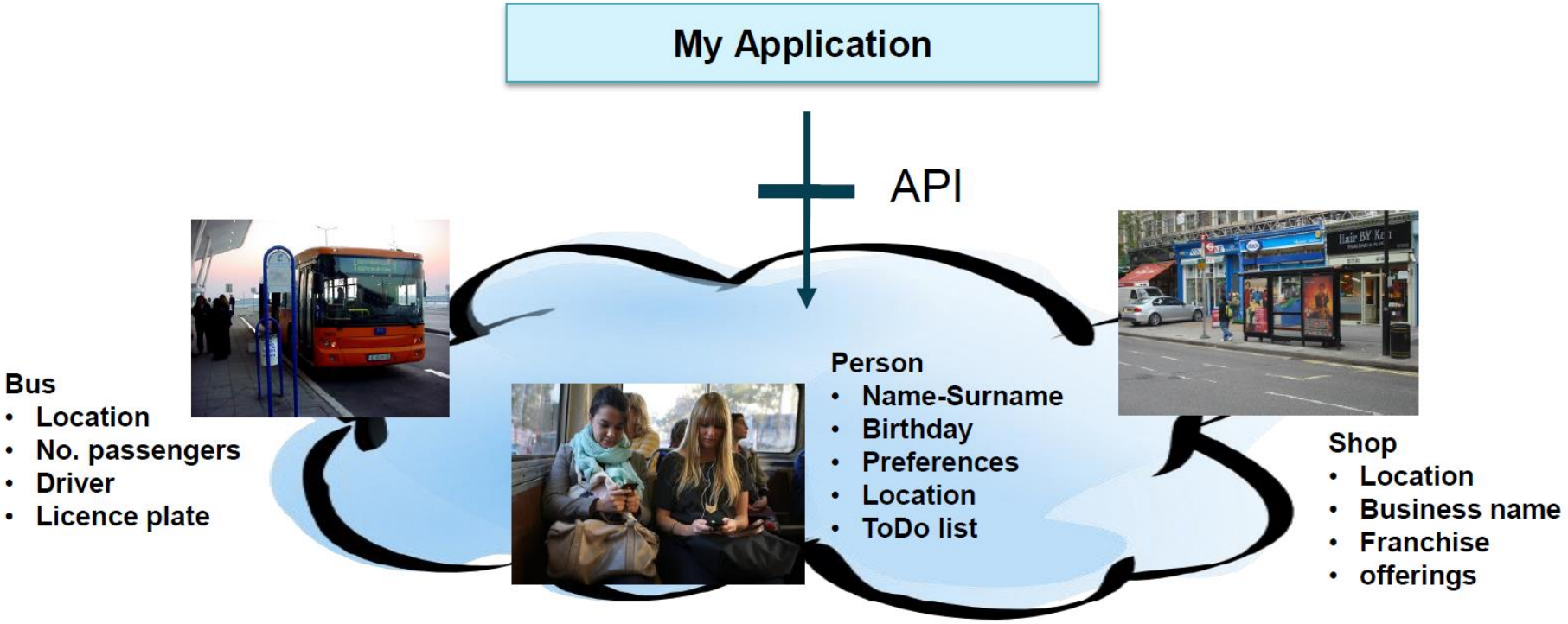


Smart Industry



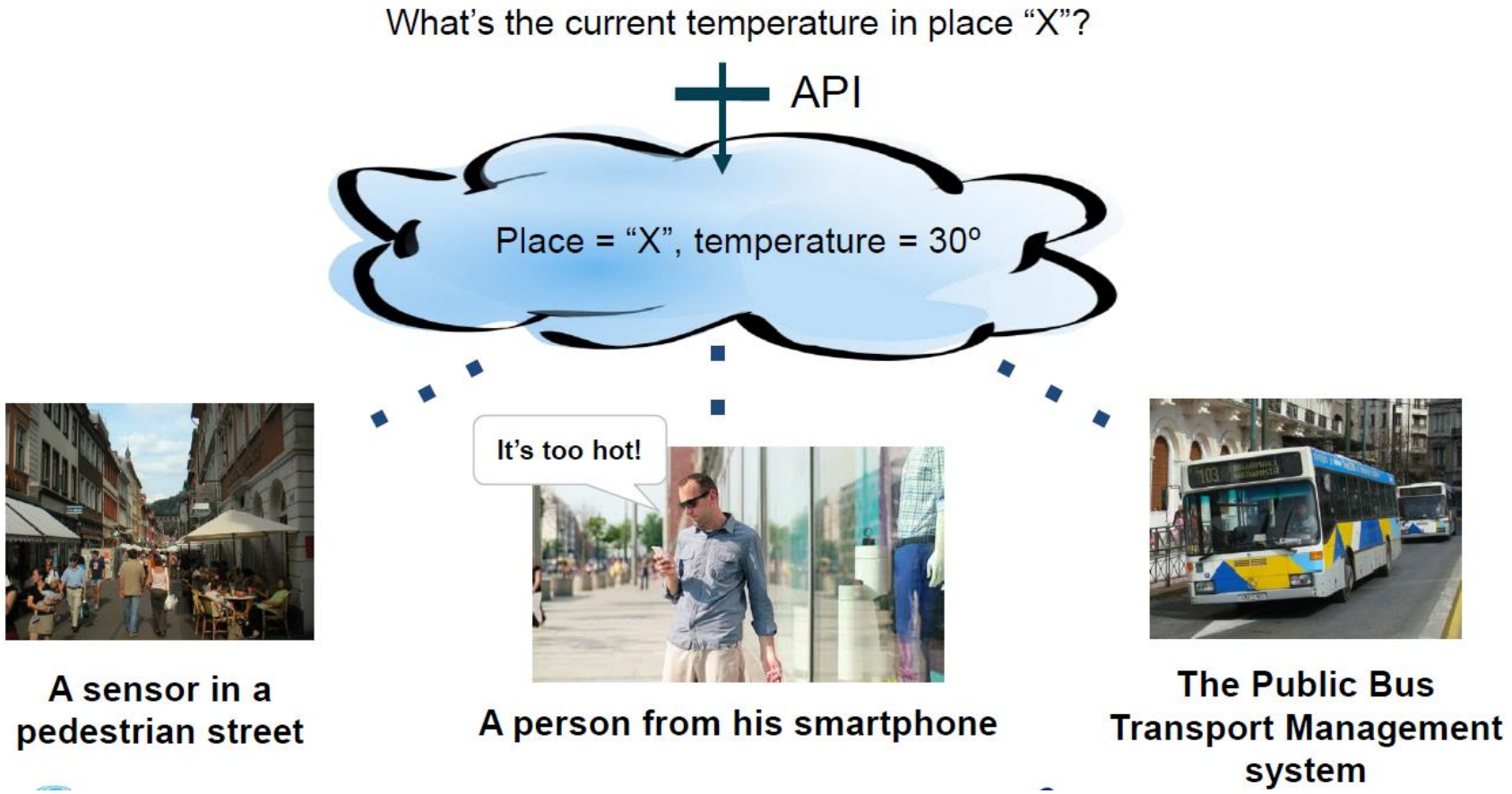
Context as the Key Driver of FIWARE

- A simple yet powerful standard API should be defined that helps programmers to manage Context information.
- Context information refers to the values of attributes characterizing entities relevant to applications



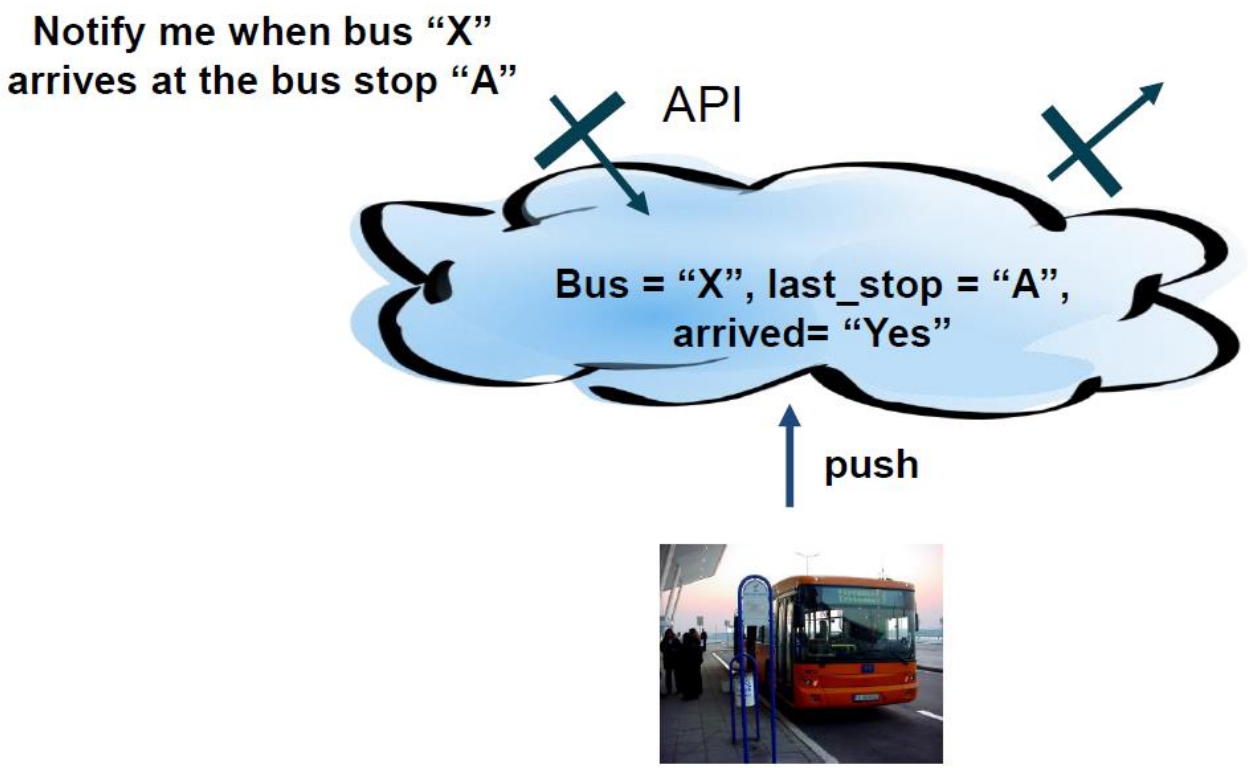
Context as the Key Driver of FIWARE

- Context information may come from many sources using different interfaces and protocols … but programmers should just care about entities and their attributes …



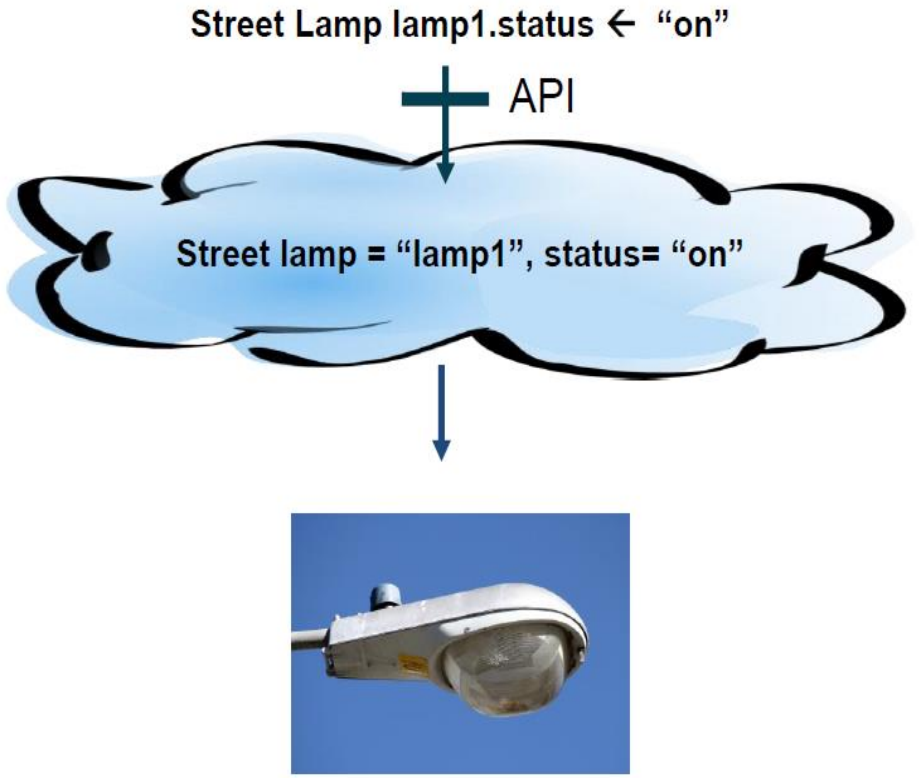
Context as the Key Driver of FIWARE

- Programmers may want to get notified when an update on context information takes place



Context as the Key Driver of FIWARE

- Acting on certain devices should be as easy as to change the value of attributes linked to certain entities



Why an open standard platform is required

- Avoid vendor lock-in:
  - Standard Southbound APIs for sensor providers.
  - Standard Northbound APIs offered to applications.
  - Portability among platform providers.
  - Interoperability of solutions enabled by the platform.
- Larger community of developers
  - True innovation.
  - Better prices.
- Not any standard is enough
  - Modularity.
  - Allow different business models.
  - Integration with standard open data platform.
  - Non-intrusive.

FIWARE Generic Enablers (GEs)

- A FIWARE Generic Enabler (GE):
  - Set of general-purpose platform functions available through APIs.
  - Building with other GEs a FIWARE Reference Architecture.
- FIWARE GE Specifications are open (public and royalty-free).
- FIWARE GE implementation (FIWARE GEi):
  - Platform product that implements a given GE Open Spec.
  - There might be multiple compliant GEisof each GE Open Spec.
- At least one open source reference implementation of FIWARE GEs (FIWARE GEris):
  - Well-known open source license.
  - Publicly available Technical Roadmap updated in every release.
- Available FIWARE GEis, GEris and incubated enablers published on the FIWARE Catalogue.

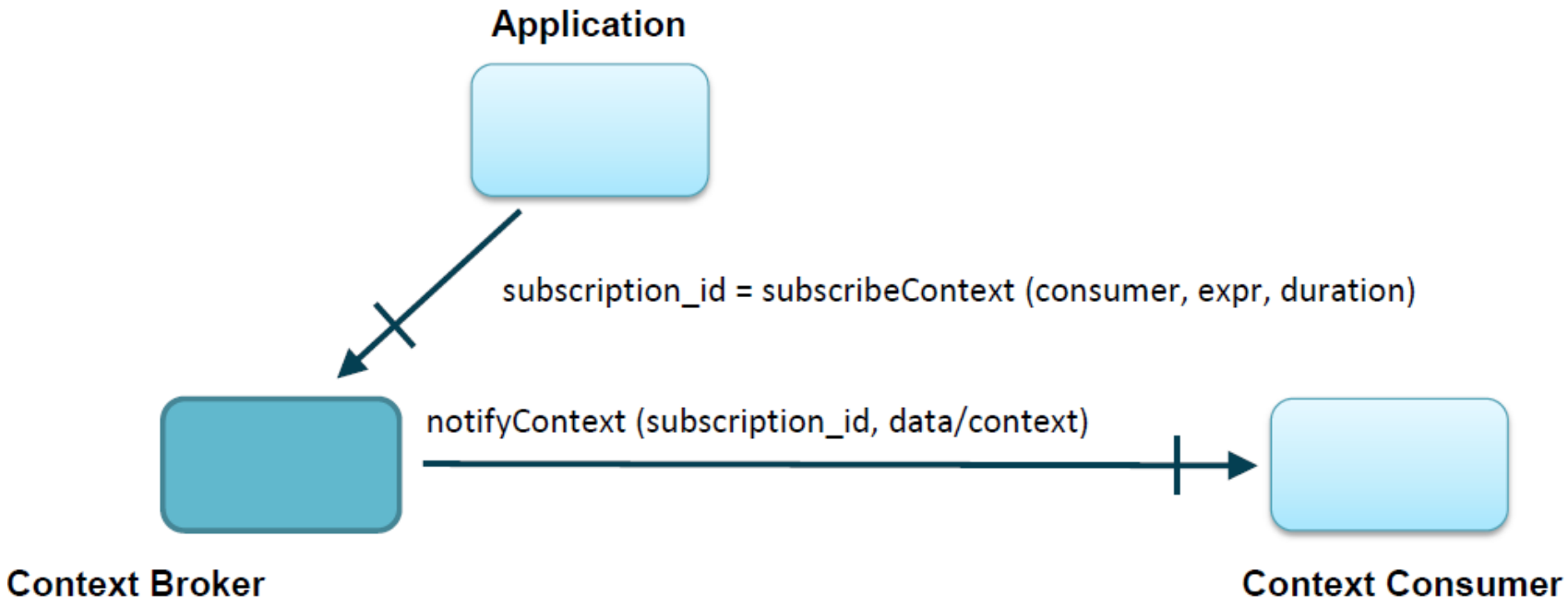
Publisher-Subscriber Paradigm in FIWARE

Basic Context Broker operations (1)

- **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

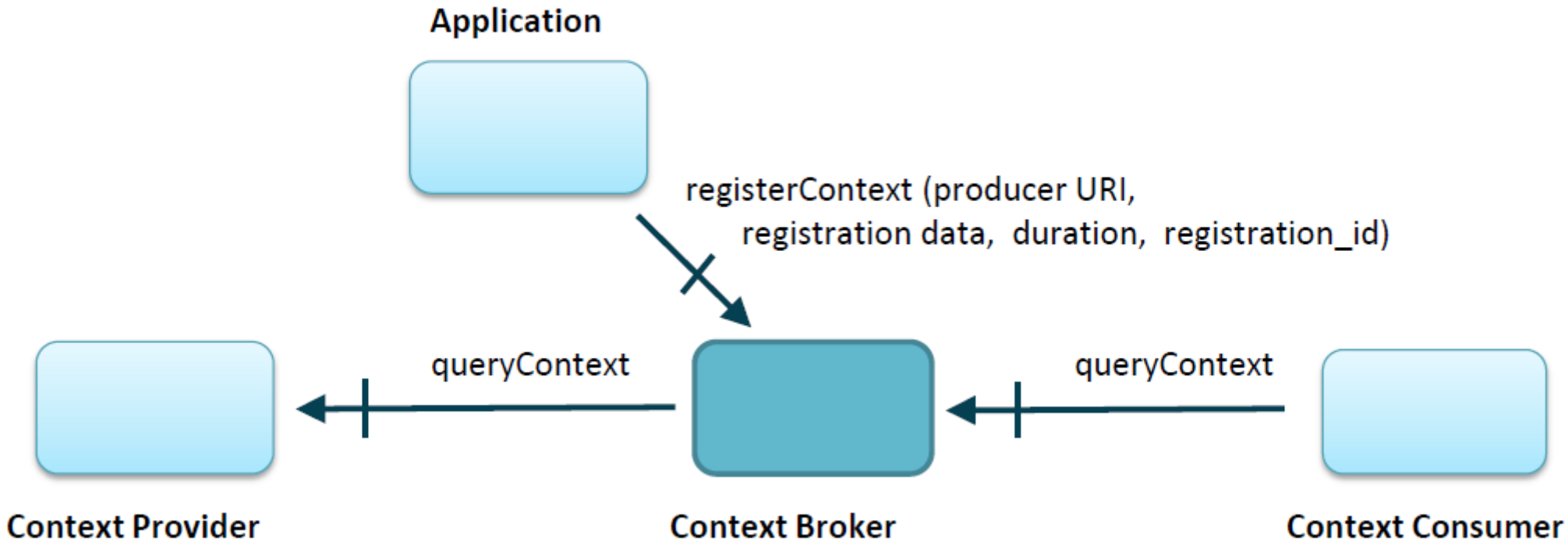
Basic entities and operations (2)

- **Context Consumers** can be subscribed to reception of context information complying with certain conditions, using the **subscribeContext** operation a Context Broker exports. 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.



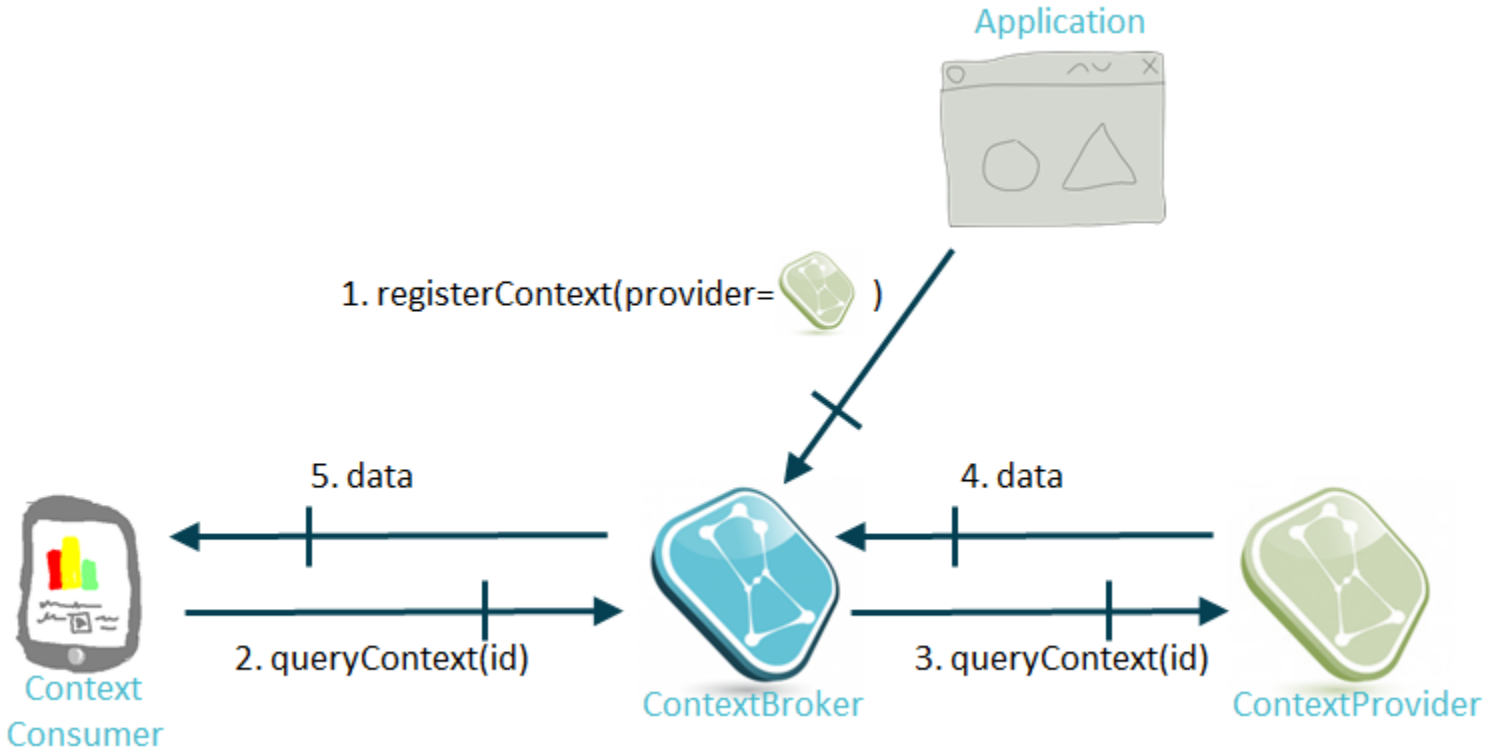
Basic entities and operations (3)

- Context Providers can be registered to the Context Broker linked to certain context information.
- A Context Broker will invoke the **queryContext** operation exported by Context Providers whenever they are queried for context information or have to notify updates in context information



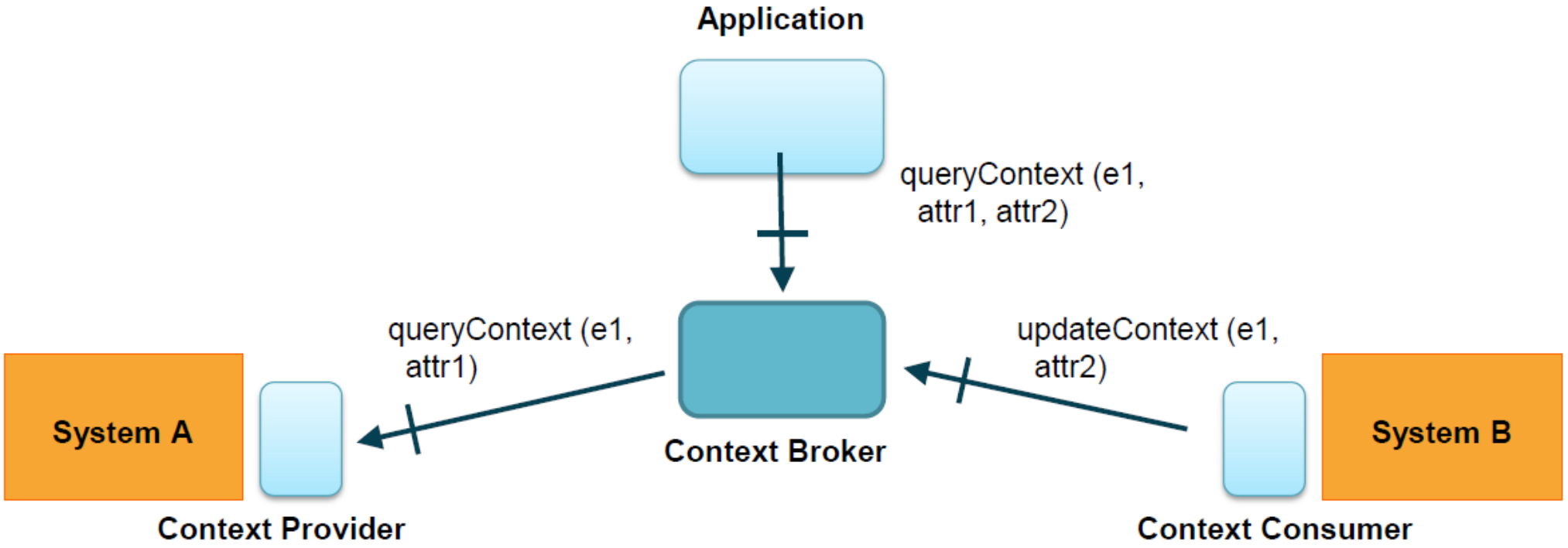
Basic entities and operations (3)

- If Orion receives a query or update operation (either in the standard or in the convenience family) and it cannot find the targeted context element locally (i.e. in its internal database) but a Context Provider is registered for that context element, then Orion will forward the query/update request to the Context Provider.
- In this case, Orion acts as a pure "NGSI proxy" (i.e. doesn't cache the result of the query internally)
- From the point of view of the client issuing the original request, the process is mostly transparent.



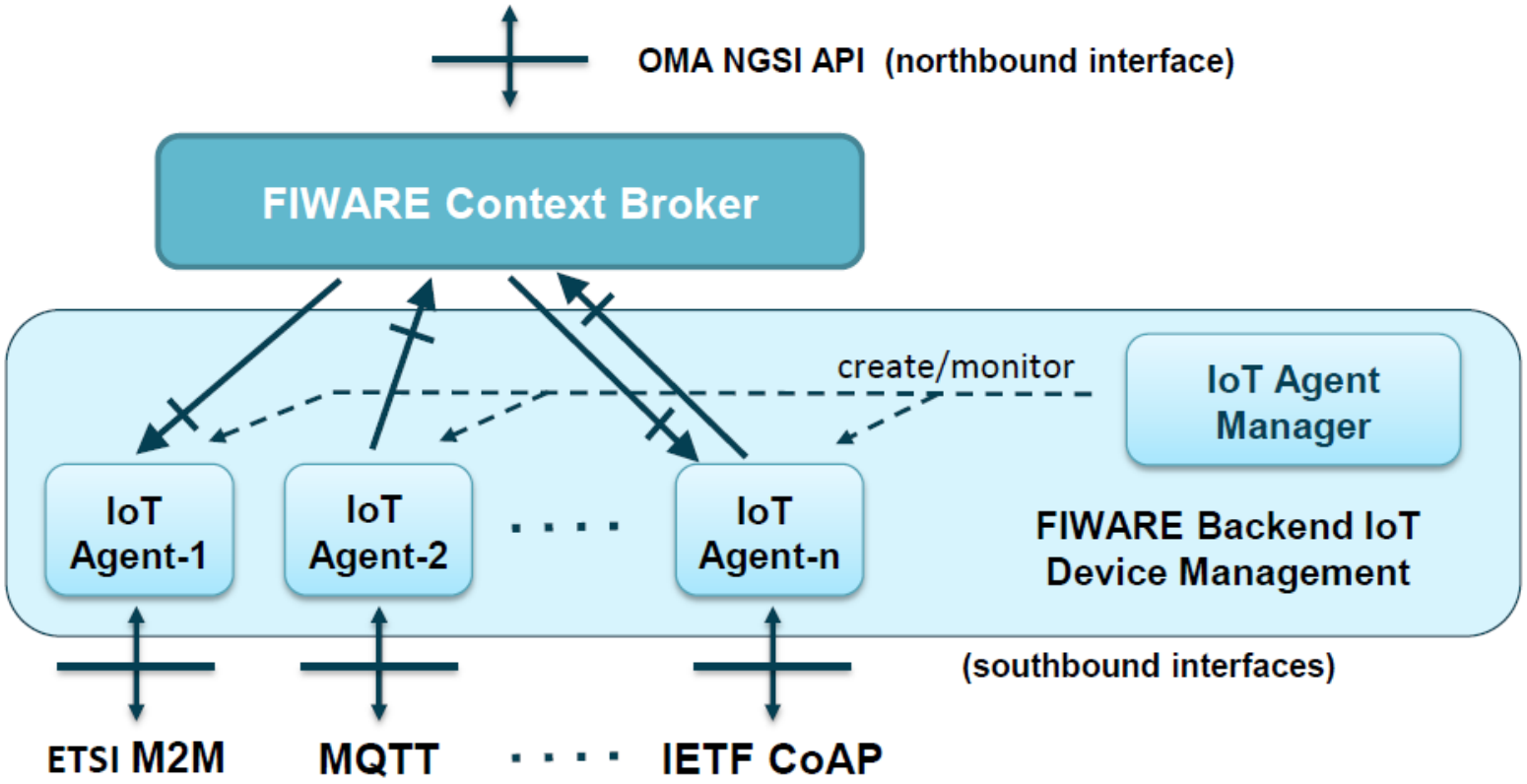
Integration with existing systems

- 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

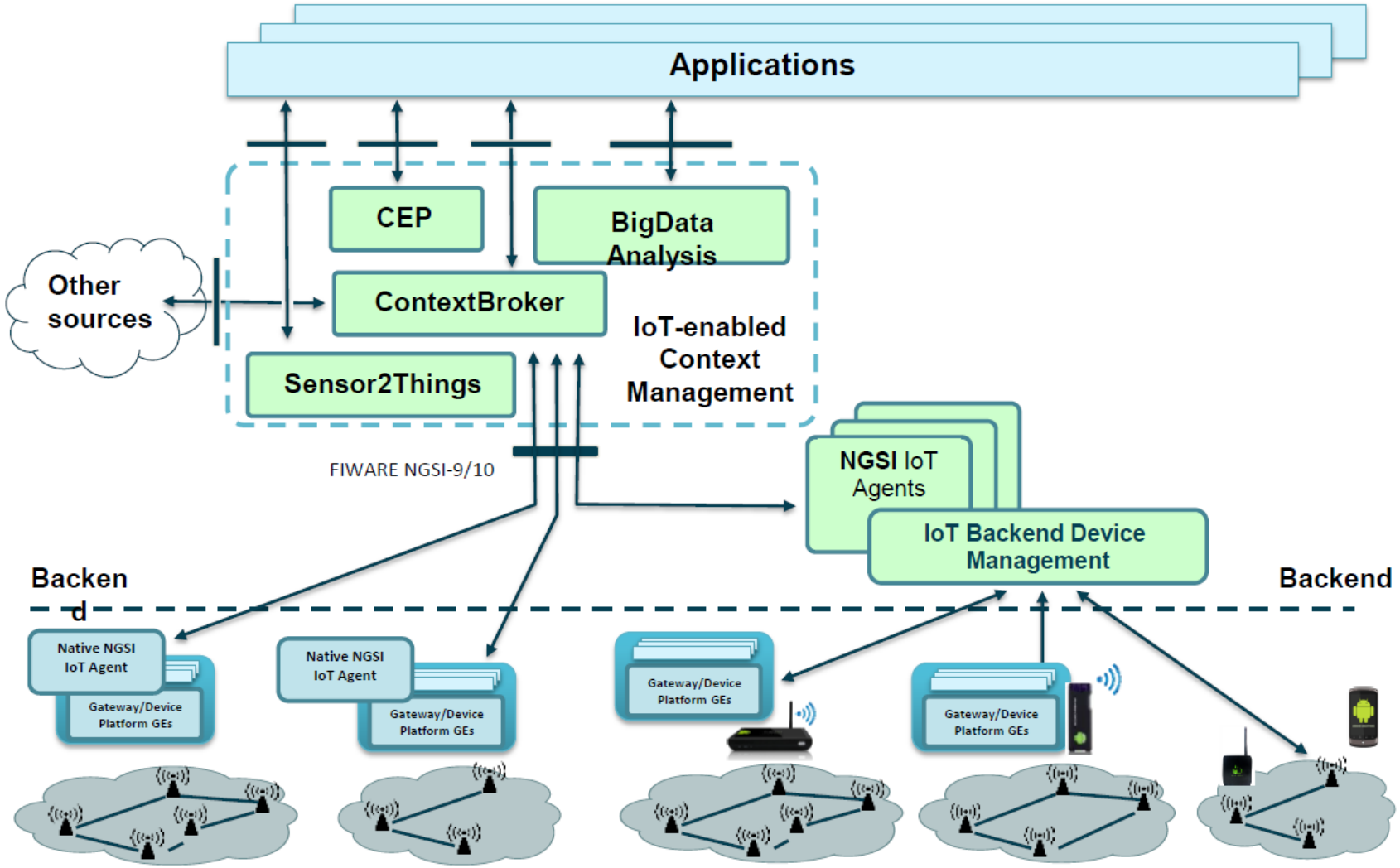


Integration with sensor networks

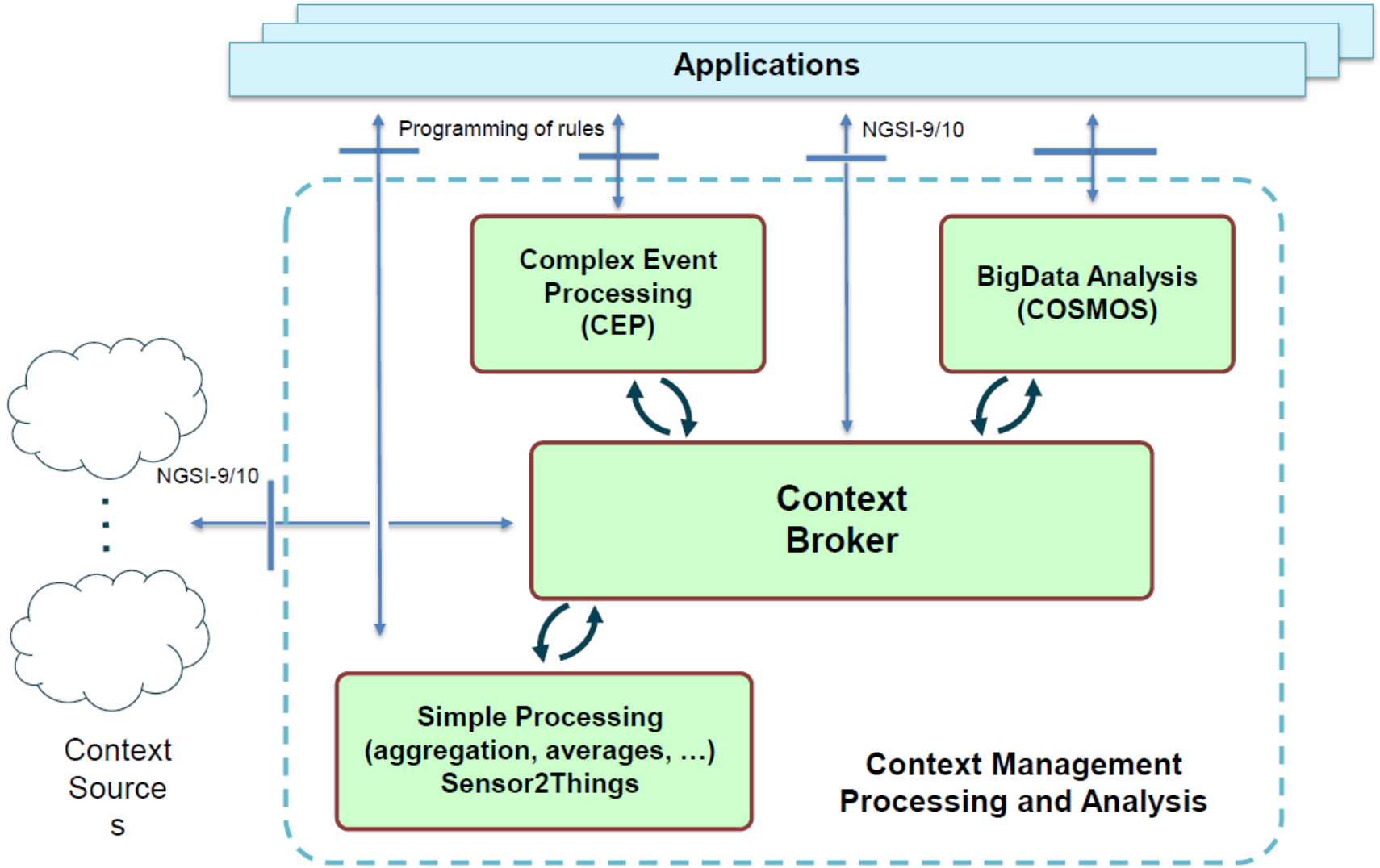
- The backend IoT Device Management GE enables creation and configuration of NGSI IoT Agents that connect to sensor networks
- Each NGSI IoT Agent can behave as Context Consumers or Context Providers, or both



FIWARE IoT-M2M & Context/Management altogether



Context Processing and Analysis



NGSI-LD
Next Generation Services Interface - Linked Data

What is NGSI-LD?

- NGSI-LD is an information model and API for publishing, querying and subscribing to context information.
- Goals:
  - Meant to facilitate the open exchange and sharing of structured information between different stakeholders.
  - Used across application domains such as Smart Cities, Smart Industry, Smart Agriculture, and more generally for the Internet of Things, Cyber-Physical Systems, Systems of systems and Digital Twins.
- Name:
  - The acronym NGSI stands for "Next Generation Service Interfaces", a suite of specifications originally issued by the OMA which included Context Interfaces.
  - The -LD suffix denotes this affiliation to the Linked Data universe.

Linked Data

- Linked Data
  - **Linked data** is structured data which is interlinked with other data so it becomes more useful through semantic queries.
  - It builds upon standard Web technologies such as HTTP, Resource Description Framework (RDF) and URIs.

- JSON-LD
  - A JSON-based method of encoding linked data, designed around the concept of a "context" to provide additional mappings from JSON to a Resource Description Framework model.
  - The context links object properties in a JSON document to concepts in an ontology.
  - In order to map the JSON-LD syntax to RDF, JSON-LD allows values to be coerced to a specified type or to be tagged with a language.

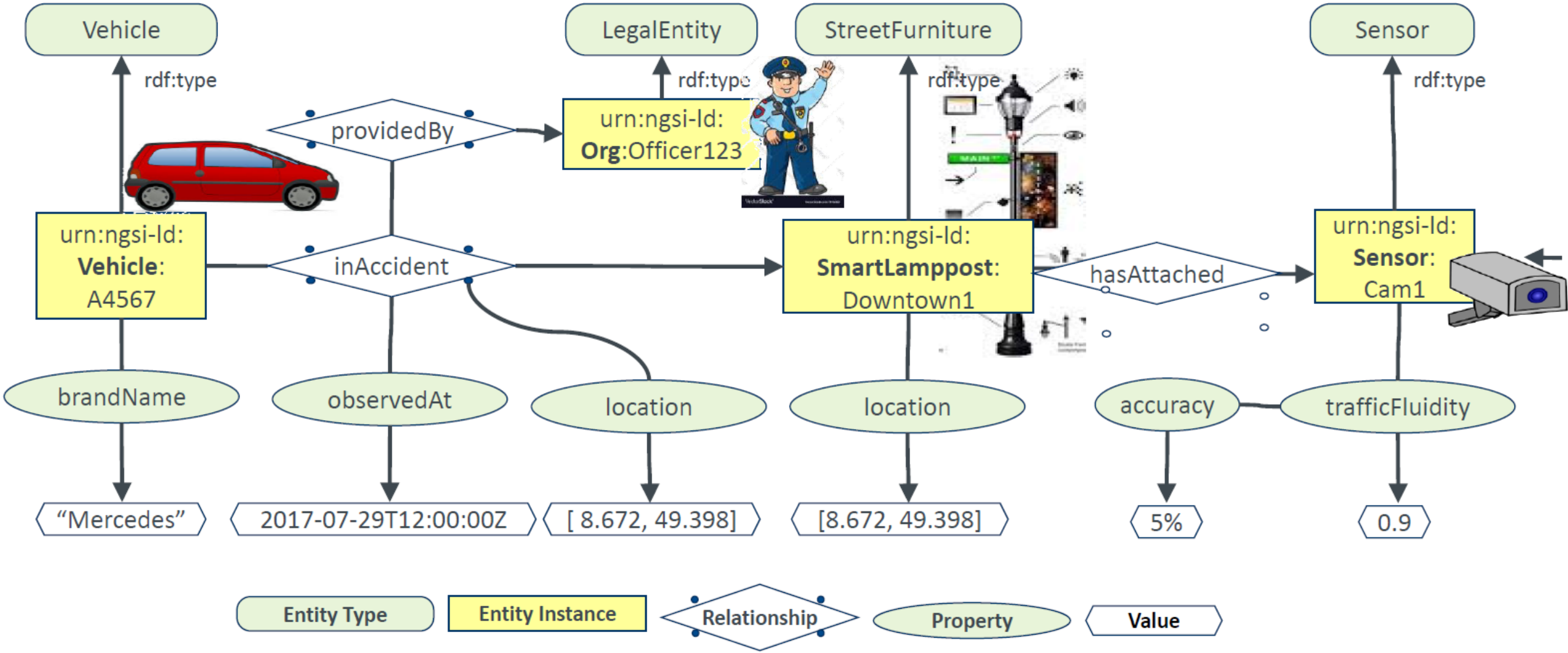
```
{
  "@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
    "homepage": {
      "@id": "http://xmlns.com/foaf/0.1/workplaceHomepage",
      "@type": "@id"
    },
    "Person": "http://xmlns.com/foaf/0.1/Person"
  },
  "@id": "https://me.example.com",
  "@type": "Person",
  "name": "John Smith",
  "homepage": "https://www.example.com/"
}
```

- Relationship NGSI-LD and Linked Data
  - NGSI-LD can be serialized using JSON-LD (in fact, the NGSI-LD specification states that NGSI-LD is based on JSON-LD).
  - The @context in JSON-LD is used to map terms provided as strings to concepts specified as URIs.
  - The Core NGSI-LD (JSON-LD) @context is defined as a JSON-LD @context which contains:
    - The core terms needed to uniquely represent the key concepts defined by the NGSI-LD Information Model
    - The terms needed to uniquely represent all the members that define the API-related Data Types

Information Model NGSI-LD

- The NGSI-LD information model represents Context Information as entities that have properties and relationships to other entities.
- The NGSI-LD meta-model formally defines these the following foundational concepts
  - **NGSI-LD Entity:** the informational representative of something (a *referent*) that is supposed to exist in the real world, outside of the computational platform using NGSI-LD.
  - **NGSI-LD Property:** an instance that associates a characteristic, an NGSI-LD Value, to either an NGSI-LD Entity, an NGSI-LD Relationship or another NGSI-LD Property.
  - **NGSI-LD Relationship:** a directed link between a subject (starting point), that may be an NGSI-LD Entity, an NGSI-LD Property, or another NGSI-LD Relationship, and an object (end-point), that is an NGSI-LD Entity.
  - **NGSI-LD Value:** a JSON value (i.e. a string, a number, true or false, an object, an array), or a JSON-LD typed value (i.e. a string as the lexical form of the value together with a type, defined by an XSD base type or more generally an IRI), or a JSON-LD structured value (i.e. a set, a list, or a language-tagged string).
  - **NGSI-LD Type:** an OWL class that is a subclass of either the NGSI-LD Entity, NGSI-LD Relationship, NGSI-LD Property or NGSI-LD Value classes defined in the NGSI-LD meta-model. NGSI-LD pre-defines a small number of types, but is otherwise open to any types defined by users.

Example: Combined data exchange using Property Graphs



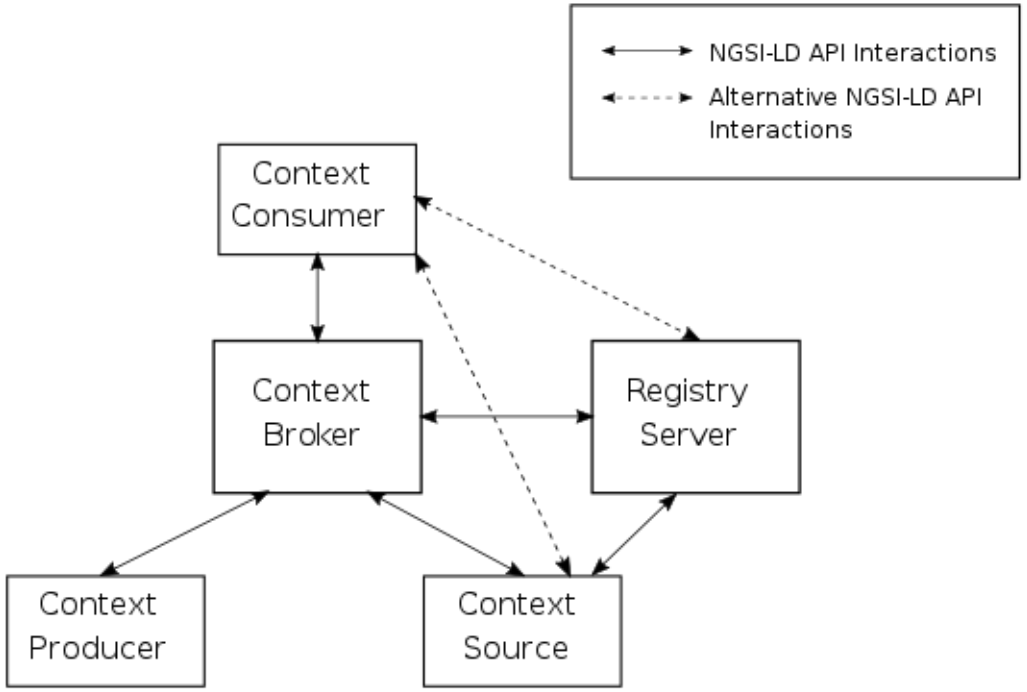
Example: Entity "Vehicle" and its @context in NGSI-LD

```
{
  "id": "urn:ngsi-ld:Vehicle:A4567",
  "type": "Vehicle",
  "brandName": {
    "type": "Property",
    "value": "Mercedes"
  },
  "inAccident": {
    "type": "Relationship",
    "object": "urn:ngsi-ld:SmartLamppost:Downtown1",
    "observedAt": "2019-05-29T12:14:55Z",
    "providedBy": {
      "type": "Relationship",
      "object": "urn:ngsi-ld:Org:Officer123"
    }
  },
}
```

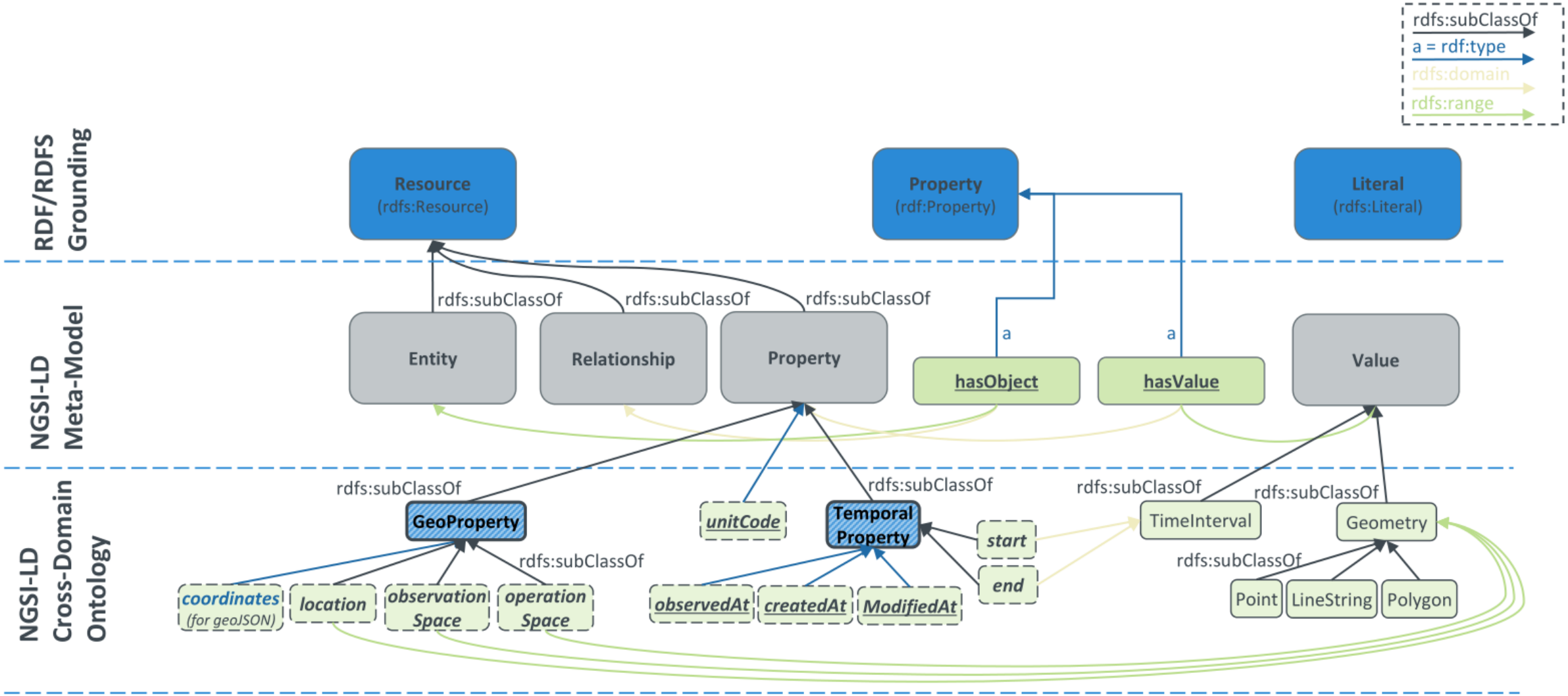
```
"@context": [
"https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld",
"https://example.org/vehicle/my-user-terms-context.jsonld"
]
```

Architecture of NGSI-LD

- **Context Consumer**: A Context Consumer consumes NGSI-LD Entities from a Context Broker (or possibly directly from a Context Source) using the Context Information Consumption functionalities of the NGSI-LD API.
- **Context Producer**: A Context Producer creates, updates and deletes NGSI-LD Entities, NGSI-LD Properties and NGSI-LD Relationships in the Context Broker using the Context Information Provision functionalities of the NGSI-LD API.
- **Context Source**: A Context Source makes NGSI-LD Entities available through the Context Information Consumption functionalities of the NGSI-LD API.
- **Context Broker**: A Context Broker acts as the primary access points to context information for Context Consumers.
- **Registry Server**: The Registry Server stores Context Source Registrations provided by Context Sources using the Context Source Registration functionalities of the NGSI-LD API.



NGSI-LD Information Model



Thank You