Green mobility data models and services for smart ecosystems

D4.1 GreenMov Reference Architecture and guidelines v1

Document Identification	
Contractual Delivery Date	31/08/2022
Actual Delivery Date	31/08/2022
Responsible Beneficiary	FIWARE Foundation
Contributing Beneficiaries	IMEC
Dissemination Level	PU
Version	1.0
Total Number of Pages:	27

Keydwords
Architecture, guidelines, federation, scalability, performance

This document is issued within the frame and for the purpose of the GreenMov project. This project has received funding from the European Union's Innovation and Networks Executive Agency – Connecting Europe Facility (CEF) under Grant AGREEMENT No INEA/CEF/ICT/A2020/2373380 Action No: 2020-EU-IA-0281. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

Document Information

Related Activity	Activity 4	Document Reference	D4.1
Related Deliverable(s)		Dissemination Level (*)	PU

List of Contributors	
Name	**Partner**
Alberto Abella	FIWARE Foundation
Brecht Van de Vyvere	IMEC

Document History			
Version	**Date**	**Change editors**	**Changes**
0.1	31/03/2022	Alberto Abella (FF)	Starting ToC
0.2	21/7/2022	Alberto Abella (FF)	Reference architecture described
0.3	17/8/2022	Alberto Abella (FF)	included more information in the architecture
0.4	22/8/2022	Alberto Abella (FF)	security requirements and FIWARE platform requirements
0.5	26/8/2022	Alberto Abella (FF)	References and answer to review's comments
0.6	28/8/2022	Alberto Abella (FF)	References and answer to review's comments (2)
0.9	28/8/2022	Alberto Abella (FF)	final review and remove comments
0.9	31/9/2022	María Guadalupe Rodriguez (ATOS)	Fix format and add information
1.0	31/8/2022	Clara Pezuela (ATOS)	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	**Who (Partner short name)**	**Approval Date**
Reviewer	Ignacio Elicegui (ATOS)	24/08/2022
Quality manager	María Guadalupe Rodriguez (ATOS)	31/08/2022
Project Coordinator	Clara Pezuela (ATOS)	31/08/2022

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
CSR	Context Source Registrations
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
JSON	JavaScript serialized object notation. It is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays (or other serializable values)
JSON schema	JSON Schema specifies a JSON-based format to define the structure of JSON data for validation, documentation, and interaction control.
JSON-LD	It is a method of encoding linked data using JSON.
LDES	Linked Data Event Stream
ODALA	ODALA is an initiative (European project) that aims to promote the use of Big Data to facilitate and speed up decision-making in public administrations. This initiative has a social focus and is designed to help the use of Smart Cities technology simply and practically.
SHACL	Shapes Constraint Language. Language for describing Resource Description Framework graphs
WP	Work Package

Executive Summary

This deliverable describes the initial version of the GreenMov common reference architecture proposed to implement the cities' use cases. It includes the combined approaches for improving scalability and a set of guidelines to be applied in specific scenarios by using the cities' pilots as examples.

The architecture is related to activity 3 as it supports its smart city services, with activity 5 through the implementation of the use cases and partially with activity 2 integrating the data sources and the entities stored in the different elements of the architecture.

The section 1 is an introduction to the purpose of this document, its relationship to other deliverables, its overall structure and the glossary of terms to understand its full content of the document.

Section 2 describes how to reach horizontal scalability in the deployments and the connection of different brokers to provide the data access and the data distribution services.

Section 3 presents the software architecture and how its different components get connected and orchestrated.

Section 4 introduces the data architecture and finally section 5, provides the conclusions of the document.

This document allows the reader to understand the underlying technologies behind the city services that support the integration, homogenisation and delivery processes that enable the implementation and deployment of the city use cases. Based on FIWARE enablers, it also provides the links to deploy the required components to support the local customization in each use case.

1 Introduction

1.1 Purpose of the document

This document describes the core elements of the technologies used by the use cases. Each use case will take these building blocks to customise their corresponding architecture instance according to the services and functionalities required to implement their final use cases.

1.2 Relation to other project work

This document is a first version of the GreenMov reference architecture, including software components and data flows. This is to be reviewed in 6 months (M18) when most stable implementations will be running, and further details/customisations can be provided.

This proposed architecture is related to activity 3 because the services here designed will be fed by the common software components (Context Brokers) and interfaces (NGSI-LD) shared across the different pilots. It is also related to activity 5 as it deploys the services in Activity 3 and build their final user applications on top of this architecture and their actual instantiations. Finally, is related to activity 2 through the design of the information architecture relying on the data brokers and the historical storage components provided by the GreenMov architecture.

1.3 Structure of the document

This document is structured in 5 major chapters, including this one **Chapter 1**.

Chapter 2 presents the methods to achieve horizontal scalability. For example, if we need more instanced nodes for providing the services or the integration of more extra data sources. It also shows how to retrieve the information when it is distributed across different nodes.

Chapter 3 introduces the different software components suggested and how they are interconnected.

Chapter 4 summarizes how the data will be coded and stored across the different elements of the architecture.

Chapter 5 presents the conclusions with the most remarkable elements of the deliverable

1.4 Glossary adopted in this document

- **Cygnus**. is a software package for managing the history of the context that is created as a stream of data which can be injected into multiple data sinks, including some popular databases like PostgreSQL, MySQL, MongoDB or AWS DynamoDB as well as BigData platforms like Hadoop, Storm, Spark or Flink [7]
- **Draco** is another software package to provide a data persistence mechanism for managing the history of context. It is based on Apache NiFi and is a dataflow system based on the concepts of flow-based programming. It supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic and also offers an intuitive graphical interface.[7]

- **Kurento**. is a component that enables real-time processing of media streams supporting the transformation of video cameras into sensors as well as the incorporation of advanced application functions (integrated audiovisual communications, augmented reality, flexible media playing and recording, etc). [7]
- **Linked data.** It is structured data, which is interlinked with other data, so it becomes more useful through semantic queries.[7]
- **Man-in-the-middle.** Cyberattack where the attacker secretly relays and possibly alters the communications between two parties who believe that they are directly communicating with each other. [7]
- **Orion.** Software solution for context information management compliant with NGSIv2 specification, created by Telefónica, FIWARE Foundation and other entities, available as a Generic enabler of the FIWARE platform [2].
- **Orion-LD**. Software solution for context information management compliant with NGSI-LD specification, created by Telefónica, FIWARE Foundation and other entities, available as a Generic enabler of the FIWARE platform [3].
- **Scorpio.** Software solution for context information management compliant with NGSILD specification, created by NEC and other entities, available as a Generic enabler of the FIWARE platform [4].
- **Stellio.** Software solution for context information management compliant with NGSILD specification, created by EGM and other entities, available as a Generic enabler of the FIWARE platform [5].

2 Reference elements

2.1 Summary of Serverless architecture for Context Broker horizontal scalability

At the beginning of the project, serverless infrastructure was mentioned to facilitate the deployment and to improve the scalability of the GreenMov solution. Indeed, serverless computing is an execution model where the code is executed by the cloud provider which is responsible to allocate the requested resources dynamically. It allows users to write and deploy code without worrying about the infrastructure. However, it has been found that among the different serverless infrastructure providers like AWS, Azure, GCP, none of them is independent, that's why it has been chosen not to use this solution for the GreenMov project.

Then, to work on scalability the decision has been made to load testing of the different existing FIWARE brokers: Orion, Orion-LD, Stellio and Scorpio. Here is a presentation of the four brokers, with the API they implement, the databases they used and if they provide temporal data API for retrieving historical data.

Table 1. Analysis of different context brokers' base software

	Orion	Orion-LD	Scorpio	Stellio
Context	Yes	Yes	Yes	Yes
NGSI API	V2	V2 + LD	LD	LD
Context DB	MongoDB	MongoDB	PostgreSQL + Postgis	Neo4j
Temporal Data	No	Experimental	Yes	Yes
Temporal Data DB		MongoDB	PostgreSQL + Postgis	PostgreSQL + Postgis
Version	3.6.0	1.1.0-PRE-683	2.1.9	1.9.1

To make the tests a Kubernetes environment hosted on OVH has been used. The cluster characteristics were: 2 CPU, 8 GB of RAM and 1 TB of storage. The test tool that has been chosen was Hyperfoil[1]. It is a web benchmark tool, released with Apache license. This tool allows the simulations of a lot of virtual users that can request the system totally independently. Furthermore, all request as asynchronous and the tool provides statistics at the end of each run. The first test that has been made consisted of 5000 virtual users per seconds, each of the users making 1 get request. The second test was 166 virtual users, each of the users making 10 updates. The duration of the two tests was 60 seconds each.

These are the results of the tests:

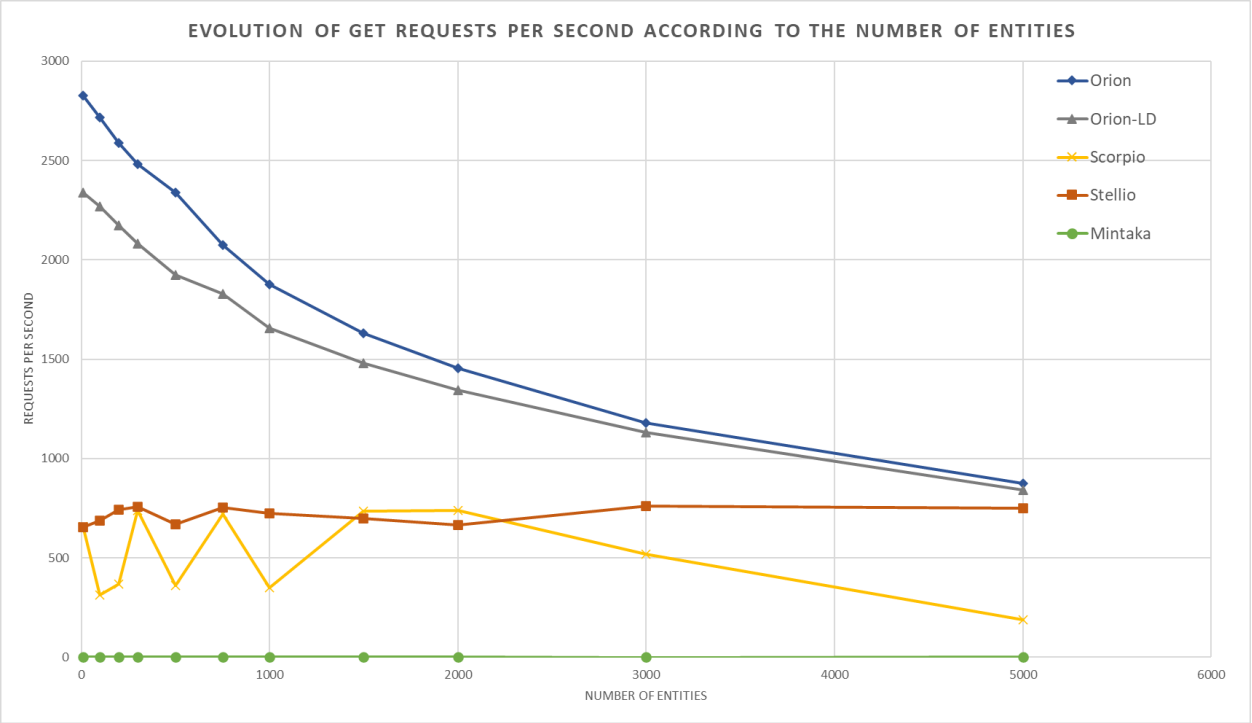

Figure 1. Performance of several context brokers for # of entities

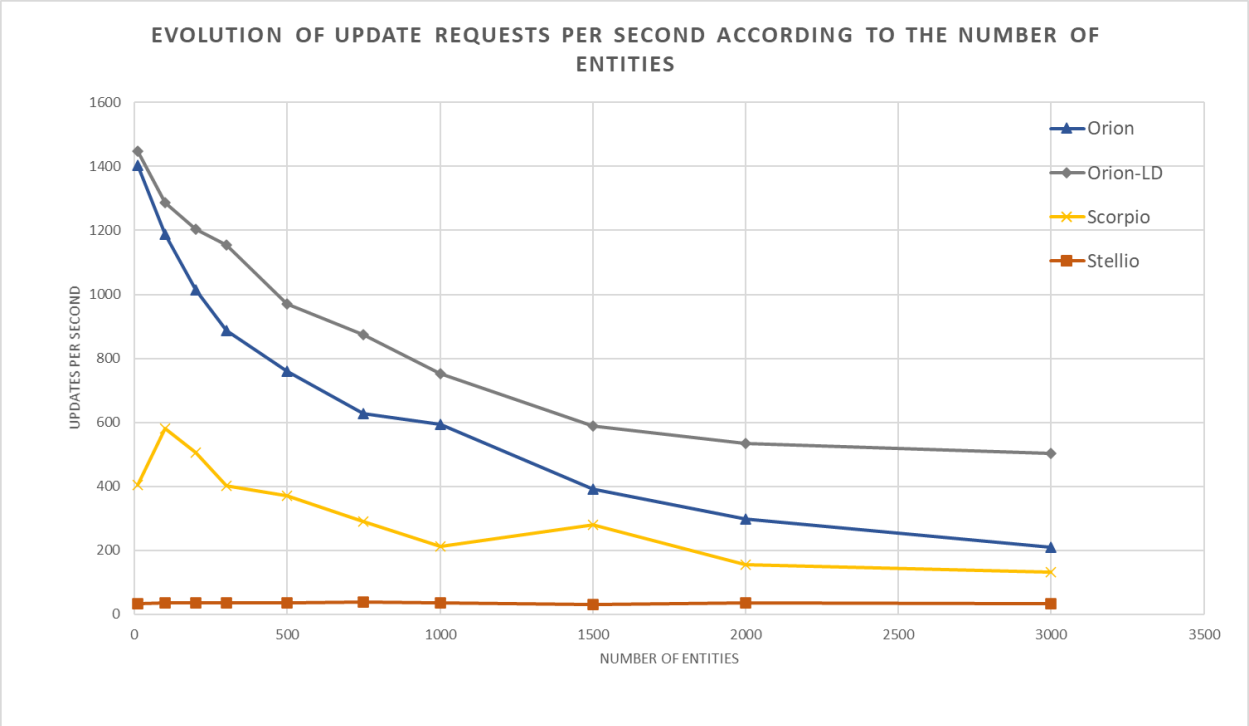

Figure 2. Performance for several context brokers for # requests

First, with these results, it can be found that Orion has good performance for both GET and UPDATE requests for a small number of entities whereas performance drops with the number of entities increasing. Then Orion-LD has almost the same results as Orion for GET requests, but it has slightly better performance for UPDATE requests. Next, Scorpio results are not as good as that of Orion-LD and performance seems to be unstable for a small number of entities into the broker. Finally, Stellio results are also not as good as that of Orion-LD, but the number of GET and UPDATE requests per second do not drop with the number of entities increasing.

Mintaka, the API used to retrieve historical data for Orion-LD broker, has also been tested and the results are not as good as those of the broker to which it is plugged. However, Mintaka is still in experimental state, so these tests will have to be redone when Mintaka is no longer in an experimental state.

To conclude this section on broker load testing, it has been seen that Orion and Orion-LD have the best load performance, but Orion is not implementing the LD interface needed in the GreenMov project. Furthermore, Orion-LD implements temporal data interface, but it is still in experimental state. Then, Stellio and Scorpio have performance not as good as that of Orion-LD for a small number of entities. However, they natively implement temporal data interface and Stellio has very stable results even when the number of entities in the broker is increasing so it may be considered for many entities.

The results of Scorpio and Stellio can be explained with the fact that tests have been made with basic configuration, no improvements have been made to increase performance. Furthermore, brokers replication have also not been tested., therefore the use cases have this information as a valuable input for customizing their respective architectures

2.2 Multi-broker source selection for federated query processing

In order to query over a federation of context brokers, applications need to understand what data is contained within the brokers' collections. Then, the application can prune context brokers that are irrelevant for its query in order to optimize federated querying. This process can be achieved in multiple ways, for example, by using the NGSI-LD Registry API (source selection on the server-side) or creating a specific index that describes what a data source contains (source selection on the client-side). In order that a third party can easily set up such an interface, replication and synchronization with a context broker is needed. This can be done through the NGSI-LD temporal query and subscription APIs. However, exposing these endpoints in an open environment where the number of requests can be significant is not desired (Fig. 1). Therefore, in GreenMov, we investigate how this source selection process can be achieved through the use of a Linked Data Event Stream (LDES) at the core of every context broker. This LDES is intended to raise the scalability of the context broker by increasing the broker's HTTP cacheability. We will first describe what LDES is and how this supports a data space. Then, we will describe in detail how this LDES supports source selection over a federation of context brokers.

An LDES allows publishers to expose their datasets as event sources. Every time a change happens in the dataset, a new, immutable event is created. This gives consumers of an LDES the full control over what it wants to achieve with the dataset (tracking historic changes of an object, have an up-to-date copy of the dataset). An LDES refers to the collection of all these immutable events and can be published in a fragmented way using the TREE specification. This also gives data publishers the opportunity to define a cost-effective data publishing strategy using hypermedia-driven fragmentations. The events that are contained in a fragment can be defined using a selector function. For example, all events that are generated between a certain time interval. When this

time interval lies in the past, immutable caching can be applied to the fragment lowering the publication cost and thus increasing the scalability of the context broker. In the ODALA and GreenMov project, NGSI-LDES has been developed as an LDES enabler for NGSI-LD systems.

Before federated queries can be solved across context brokers, a consumer needs to understand what data is contained within the broker's collections. First, this requires the analysis of the collection's schema (e.g., expressed in SHACL shapes). For example, when trying to do forecasting of bike sharing station capacities, there is no need to traverse across context brokers that do not have shared mobility information. Second, the coverage of the collection needs to be investigated. More specifically in GreenMov, we investigate how the geospatial coverage of context brokers can be exposed for applications that request *in-situ* data, e.g., based on their actual location or their destination. In the current NGSI-LD specification, a server-side Registry API is defined that allows consumers to perform source selection by retrieving and discovering Context Source Registrations (CSRs). In GreenMov, a client-side source selection building block will be created that has a two-fold approach. The first approach is to use the SHACL shapes of the NGSI-LDESs (1a) and the geospatial coverage index (1b). The second approach is to use the NGSI-LD Registry API. Based on the type of entity and location as input parameters, the component returns which sources are relevant using both approaches. A benchmark will be performed that loads both Coverage Index and Registry API from the NGSI-LDES, which aligns with the Data Space vision that is being deployed in Flanders.

3 Basic software components

The Reference Architecture should address how the integration of data and information across different systems on the use cases will be achieved, ensuring sustainable and efficient service provisioning. Interoperability should be supported in line with relevant standards (REST, NGSI, JSON-LD). In this respect, the Context Broker technology will play a cornerstone role for the integration, following a system of systems approach, of data and information across the systems implementing the different services for a given focus area (e.g., traffic management, environmental impact, or other system in relation to public services). Some of these systems will be provided globally as a service from public clouds. The Context Broker [6] building block has been the preferred technology for other EU programs.

The next figure shows a diagram with the main components based on a FIWARE reference architecture [7] for smart cities, where the requirements for the different use cases Murcia y Molina, Nice and Flanders has been taken into consideration.

The central element of this architecture is the context broker where the retrieval and sharing of the information happens.

It has four main levels [8]. the upper level is the one for the connection with the legacy systems of the city or other general applications. This level will be extremely customized for every use case.

The second level is the one with the context broker and some adapters for the connection with the legacy systems. It has to be noted that in right side of this level there are the identification and identity blocks.-the use and sharing of these elements in the second level allows the flexibility to provide solution to the different use cases but keeping the same basic software structure which allow scalability and shared knowledge. Besides this, the fact that this components come from a FIWARE reference architecture ensure a wide knowledge base to solve any issue of the use cases.

The third level of the schema is populated by the adapters with the IoT world, cameras, sensors, etc. Again, being based on the FIWARE components ecosystem ensures the availability of many solutions for the connection a large database of integration with other IoT elements and some share software frameworks for these cases in which a customization is required.

The lower level (fourth) includes those IoT elements and other data sources for the solution of the city that can be connected thanks to the previous level without much hassle. This level retrieve data from different sources that can be modelled into common data models, eventually allowing seamless data sharing between use cases or city's ecosystems and being stored into entities in the second level.

.

3.1 Global diagram

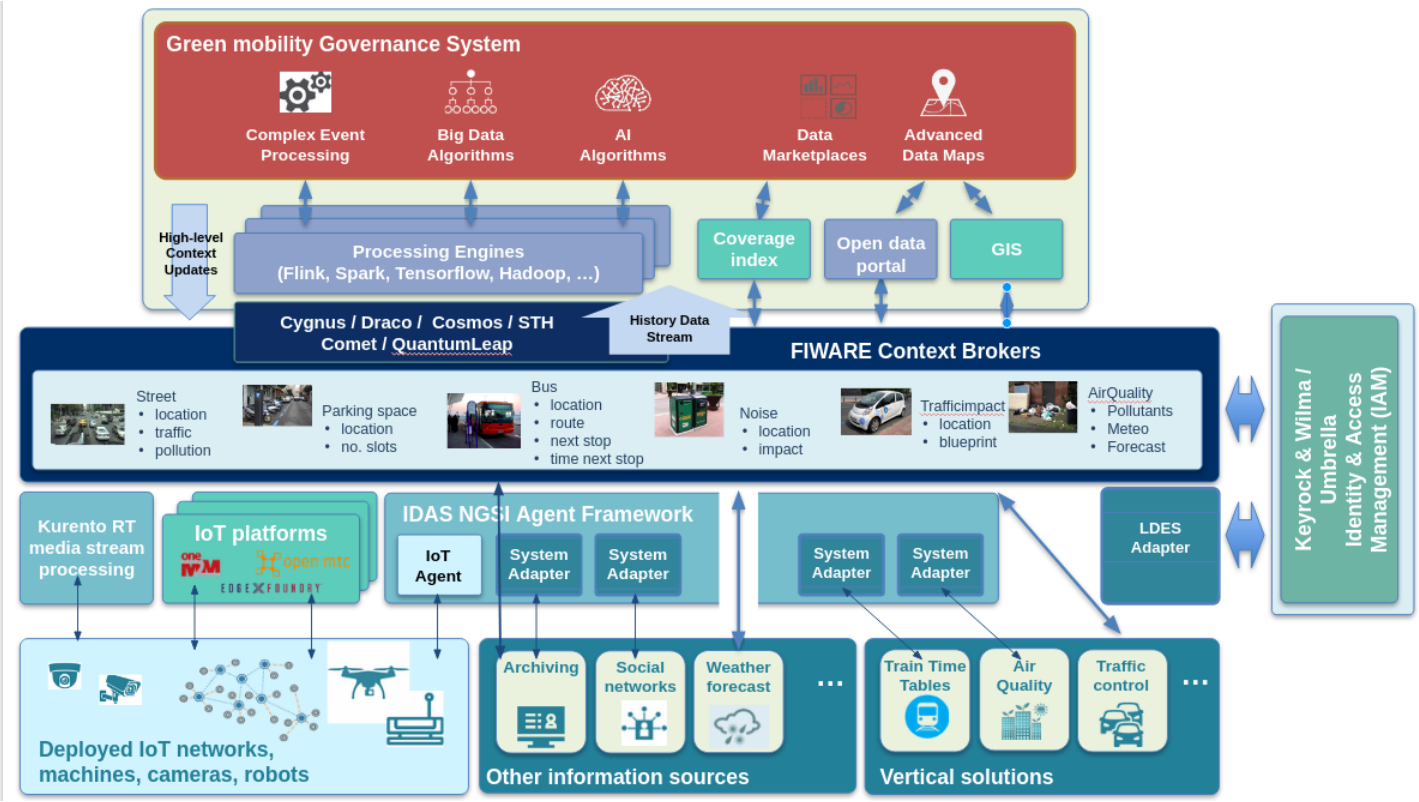

Figure 3. Global diagram for the reference architecture. source: FIWARE reference architecture for cities adapted for GreenMov.

- A Context Broker component[1], is at the core of the architecture, keeping a digital twin representation of the real-world objects and concepts relevant to the specific problem tackled: Environmental sensors, traffic sensors, noise sensor, bike stations, etc.
- Southbound to the Context Broker, the NGSI IoT Agents, available as part of the FIWARE IDAS framework, are used for connections to the different sensors or actuators, used for example to detect available bikes on the stations. They perform the necessary conversions between IoT protocols and NGSI. In addition, System Adapters developed based on the IDAS Agent library cope with the connection to the legacy systems of the city as described in the services architecture. FIWARE component Kurento is able to process the video streams of cameras deployed in the shop floor, which are helpful to detect potential obstacles or risky situations.
- Southbound to the Context Broker, data sources can be serialized in multiple formats (JSON, CSV , JSON-LD...) and published through heterogeneous APIs (JSON API, LDES, NGSI-LD).
- Northbound to the Context Broker, a number of tools are targeted to support real-time data processing of the streams of history data generated as context / digital twin information evolves over time. A combination of open-source components from third party products and advanced data maps for monitoring processes. A number of FIWARE Data Connectors (Cygnus, Draco, Cosmos, STH Comet, QuantumLeap) are available as part of FIWARE to facilitate transference of historic context / digital twin information to these tools.
- Transversal to all these layers, a number of FIWARE components support Identity and Access Management (e.g., Keyrock, Wilma, AuthZForce). They control the flow of data across the different layers. With regards to the access to the Context Broker, they enforce the policies establishing what users can update, query or subscribe to changes on context / digital twin data. Note that the flow of data is not only south to north in the picture. Northbound applications can perform updates on context data, which in turn will trigger changes in the sensors, actuators or systems that are connected southbound.
- Northbound to the Context Broker, the NGSI-LDES component creates a scalable interface for data sharing. The component allows applications to replicate and synchronize with the historic and real-time context of entities. Example of such an application is the Coverage Index or Registry API, which can be served by a Context Registry and can be used by a source selection component. NGSI-LDES requires that the temporal and types of interfaces are available on the Context Broker.
- An important point to highlight is that in this reference architecture, likewise the FIWARE reference architecture, that is based on, is not about taking it all or nothing. You are not forced to use all the components described in the schema, but you are free to use other third-party platform components as well to design the hybrid platform of your choice. Thus, for example, you may opt for using a concrete IoT platform instead of IDAS IoT Agents to interface with sensors and actuators as reflected in the picture. As long as it uses the FIWARE Context Broker technology to manage context information, your platform can be labelled as "powered by FIWARE" and solutions build on top as well. Thus, the different use cases involved in the project can customize this architecture to their own needs.

[1] According to Gartner, referenced by https://www.linkedin.com/pulse/what-context-brokers-alvaro-martin/ A context broker is a service that is designed to gather reachable context data of a variety of types, sources and velocity. It then applies conditioning, integration, rules and analytics to derive the reduced prepared context data, actionable at a point of business decision by a system or a human

3.2 Context broker

Orion context broker integrates information from sensors, systems and other machines in the different use cases breaking information silos. It allows not only the retrieval of information from heterogeneous sources but also the querying in time and the geoquerying in spatial dimension. Additionally, it also allows the users to subscribe to changes or updates and to receive notifications when the conditions are met.

This broker must support following entry points to be compatible with the NGSI-LDES component:

- the NGSI-LD temporal interface;
- the NGSI-LD types of interfaces.

Examples of Context Broker we see fit: Orion-LD + Mintaka, Scorpio, Stellio.

3.3 OPC UA agent

The OPC UA Agent [8] is a flexible and configurable software component in the FIWARE architecture. It is able to exploit Industrial IoT Data in Motion streams, together with other heterogeneous data sources from mobility, cities, the environment or meteorological data.

This software agent connects OPC UA servers with the Context Broker using the FIWARE NGSI standard API. This way. It enables the development of advanced context-aware services.

3.4 NGSI-LDES

NGSI-LDES outputs Linked Data Event Streams (LDESs) from an NGSI-LD context broker. Per type of entity, a separate LDES is published using a hierarchical fragmentation strategy. Data is not replicated as the component works as an adaptor on the temporal interface. Also, the NGSI-LD types of interface is required on the Context Broker to generate a DCAT catalogue out of the box.

3.5 Coverage index

A Coverage Index allows clients to efficiently discover the coverage of data sources on certain properties. In GreenMov, we provide a Coverage Index that exposes the geospatial coverage of context brokers for applications that request *in-situ* data, e.g., based on their actual location or their destination.

3.6 Source selection building block

- DCAT + SHACL as input
- Coverage index
- NGSI-LD Registry API

The source selection building block returns which sources (e.g., context brokers) are relevant based on the type of entity (e.g., Bike Station) and location (e.g., a certain part in the city) as input parameters.

Source selection interfaces can be made by third parties from the NGSI-LDES (Fig. 4), which aligns with the Data Space vision that is being deployed in Flanders.

The first approach is to use the DCAT description and SHACL shapes of the NGSI-LDESs (1a) and the geospatial coverage index (1b). The second approach is to use the NGSI-LD Registry API. Based on the type of entity and location as input parameters, the component returns which sources are relevant using both approaches. A benchmark will be performed between NGSI-LDES/Coverage Index and Registry API.

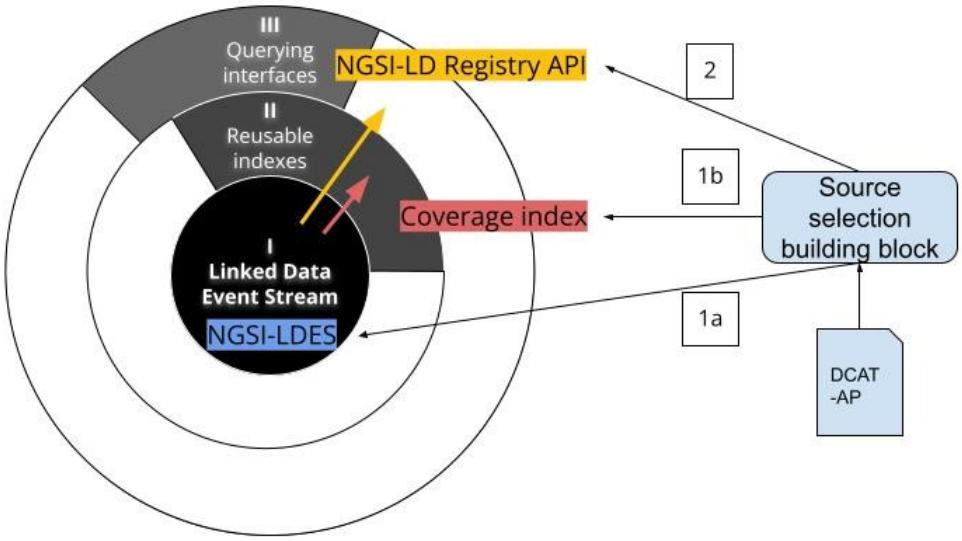

Figure 4. The source selection building block uses the NGSI-LDES+Coverage Index (1) or NGSI-LD Registry API (2) to retrieve the sources that are relevant for a certain type of entity and location

3.7 Security

The security of the overall system is a multilayer building, and the precise requirements should be described for each implementation of the reference architecture.

anyhow some elements to build the security of the system has to be in place.

- A proper identification and permission system has to be implemented across the system. This mechanism has to be able to restrict access to non-authorized people to the restricted operations.
- For each component a precise description of the function has to be available.
- Additionally, a feedback mechanism must be in place, and it has to be proved that it is regularly checked, and the raised issues addressed and eventually solved.
- The process for contribution of new features or fixes has to be clearly explained and the mechanism for contribution also available and clear. Some standards for contribution to be accepted should be also documented (in order to prevent low quality contributions that can drag the overall security of a component). Complementary, a discussion mechanism for new features or approaches to fixes has to be available.
- It is required to have updated documentation of the software
- Last but not least the overall project has to be maintained. (e.g. for basic software updates)
- A unique version number has to identify different version and a proper naming method for versioning should be clearly described and adopted.

- the different software blocks must have an automatic testing mechanism and explanations on how to run it to check proper running.
- Cryptographic protocols and algorithms have to be implemented whenever necessary to ensure security. The default security mechanisms within the software produced by the project MUST NOT depend on broken cryptographic algorithms
- Whenever applicable every element of the project MUST use a delivery mechanism that counters man-in-the-middle attacks.

4 Data Architecture

4.1 Data Storage architecture and technical format

As long as NGSI-LD is the chosen common standard within GreenMov services and components for interchanging information between the main systems (but for the sensors) the common standard for document sharing and eventually for some of the data storage will be JSON (and specifically JSON-LD)

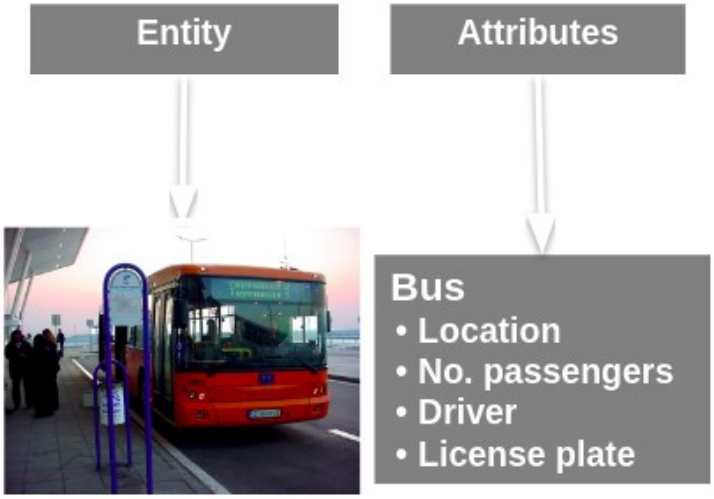

Figure 5. Entity and attributes

This entity[2] could be represented in a JSON payload this way

Table 2. JSON representation of an entity and its attributes

```
{“id”: ”ngsi-ld:BUS:001”,
“type”: “Bus”
“location”: [215, 33.4],
“driver”: ”ngsi-ld:DRIVER:002”,
“licensePlate”: “4536KVM”
}
```

The information will be split into elements named entities that can be a single JSON payload. the payload is a series of keywords (attributes) and an attached value that can be a single value (e.g. string, data-time, number)

[2] This entity is described here only for explanatory purposes and it does not correspond with any of the real ones in the use cases. worth to be noted that the value for the attribute driver can be the pointer to a different entity of the type of drive with their own attributes, allowing to create relationships between the entities stored in the context broker.

or complex values like an array or a more complex object. These entities have a unique attribute, the identifier or '*id*', that allows to reference them uniquely, and another attribute, the *type*, which determines their class.

Entities belonging to the same class can have different internal structures because do not need to have proper values for all the attributes These elements can include references to other elements store across the different systems.

4.2 Basic data classes / entities

Although the different pilots would share the same technical architecture there would be quite limitation if they do not share the data structures. Thus in activity 2 a set of shared data models have been defined so every entity can be shared between the different pilots.

The list of entities is defined in deliverable 2.2, section chapter *4 Data Models for GreenMov use cases*.

5 Requirements for a generic enabler

The official site for the complete list of requirements and the process to achieve it is available at the FIWARE site[9]. Main ones are summarized in the coming sections.

5.1 Licensing and open SSF Best practices signature

Every Generic Enabler MUST comply with the Licensing and IPR Management requirements. Summarizing.

- The source code of the product **MUST** be licensed under one of the well-recognized open source licenses approved by the Open Source Initiative.
- The open-source license under which source code of the product is licensed **MUST** be clearly mentioned in a first-level section of the README.md file included in the main GitHub repository.
- When using a copyleft open-source license, a specific explanatory paragraph of legal opinion MUST be added in the section where the open-source license is mentioned
- The legal opinion paragraph above **SHOULD** be accompanying the text describing the adopted open-source license in the headers of all source code files for the product.
- Every enabler MUST be open to third party contributions. All offered contributions MUST be reviewed within a "reasonable" time frame.
- There MUST be a document (CONTRIBUTING.md guidelines) clearly describing the terms under which the IPR of contributions to the source code of the product will be managed. Such document MUST be made accessible in (or map to) a first-level section of the README.md file included in the associated GitHub repositories.
- The CONTRIBUTING.md guidelines **MUST** include the template of the Contribution License Agreement for individuals and entities contributing code to the component. As a reference for producing these templates, the following templates derived from the Harmony Agreements project are provided:
    - Individual CLA
    - Entity CLA
- When using a copyleft open-source license, IPR Management rules for contributions **MUST** include clauses as follows:
    - There should be at least one organization which can exercise IPRs on the whole software.
    - There is a commitment to transfer to the FIWARE Foundation the IPRs on the whole software in case that the software is no longer supported by the organization(s) that currently own(s) IPR on the whole software.

5.2 General requirements

A FIWARE Generic Enabler **MUST** fit well in the architecture of a *"Powered by FIWARE"* solution:

- Integrate well with architectures where context management is cornerstone and addressed using FIWARE NGSI (currently FIWARE NGSIv2, compliant with ETSI NGSI-LD in the future).
- Be able to fit within one of the defined FIWARE chapters.

5.3 Code control tool requirements and public backlog

GitHub and GitHub Issue tracking **MUST** be used.

5.4 Documentation requirements

A generic enabler to be accepted needs to have accurate, current Documentation **MUST** be available on Read the Docs and as GitHub content. To guarantee that documentation is of high quality, development related **documents MUST** be **peer-reviewed and QA verified.** See Documentation Guidelines [8] for the best documentation practices. Should you want to benefit from automatic documentation generation systems, namely, Read the Docs [9], you **MUST** use an approved markup notation:

- Markdown [10] is preferred for simple documents.
- Restructured text [11] is an acceptable alternative for complex documentation.

5.5 Development requirements

Every Generic Enabler must sign-up to the OpenSSF Best Practices Badge Program [12] and display the badge.

The Open Source Security Foundation (OpenSSF) [13] Best Practices badge is a way for Free/Libre and Open Source Software (FLOSS) projects to show that they follow best practices. Projects can voluntarily self-certify, at no cost, by using this web application to explain how they follow each best practice.

API Specifications MUST be provided. Preferred format is OpenAPI [14], a.k.a. Swagger, format.

6 Conclusions

This document provides the basic description of a reference architecture that has to be customized in each implementation of the different use cases.

It identifies and lists the different components potentially usable at the different pilots and sets some conditions on its implementation. The reference architecture is based on existing projects, therefore the integration of most of its elements has been already tested. An analysis of the performance of the context brokers, one of the core components of the architecture, is included showing different scenarios that every use case can take as an input for setting up its specific implementation.

The description of the main building blocks of the architecture is included in chapter 3 together with a diagram showing the connection between the elements.

Some requirements in terms of security are also listed and more can be found in the references for becoming a approved generic enabler of the FIWARE platform.

Finally, the document describes the basic data architecture and references to the deliverable 2.2, where the precise definition of the different data models is described for its use in the different use cases.

GreenMov project is creating some ad hoc solutions that can be useful beyond the scope of the project. Thus the requirements for becoming an official component of the FIWARE platform are listed in chapter 5.

7 References

[1] https://hyperfoil.io/

[2] https://github.com/telefonicaid/fiware-orion

[3] https://github.com/FIWARE/context.Orion-LD

[4] https://github.com/ScorpioBroker/ScorpioBroker

[5] https://github.com/stellio-hub/stellio-context-broker

[6] FIWARE Catalogue: https://www.fiware.org/catalogue/

[7] Smart Cities Brochure. https://www.fiware.org/wp-content/uploads/Smart-Cities-Brochure-FIWARE.pdf

[8] FIWARE for Data Spaces. https://www.fiware.org/wp-content/uploads/FF_PositionPaper_FIWARE4DataSpaces.pdf

[9] https://fiware-requirements.readthedocs.io/en/latest/

[10] https://fiware-requirements.readthedocs.io/en/latest/development/index.html#documentation

[11] https://readthedocs.org/

[12] https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

[13] https://github.com/ralsina/rst-cheatsheet/blob/master/rst-cheatsheet.rst

[14] https://bestpractices.coreinfrastructure.org/en/signup

[15] https://openssf.org/

[16] https://github.com/OAI/OpenAPI-Specification