D2.3 Extension of FIWARE for supporting water management and quality monitoring use-cases

Author: Lluis Echeverria (EUT)

Co-Authors: Marc Ribalta (EUT), Aitor Corchero (EUT), Fernando López (FF), Alberto Abella (FF), Benoit Orihuela (EGM), Chris Pantazis (NTUA), Siddharth Seshan (KWR), Gareth Lewis (UNEXE)

May 2022

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant agreement No. 821036.

Disclaimer

Intellectual Property Rights

Project Consortium

Executive Summary

The Fiware4Water project reaches its end, and the four Demo Cases have constructed the corresponding digital platforms and smart applications for intelligent water management on top of the F4W Reference Architectures designed in WP2.

The present document represents the last step and efforts in the definition and development of the proposed FIWARE-enabled F4W Reference Architectures, and closes the working loop started in deliverables D2.1 and D2.2. In this context, the Southbound and Northbound RAs' interfaces are described, representing the RA integration with i) Context Information Producers and IoT devices and ii) high-level services for data storage, manipulation and visualization, respectively. Thanks to the patterns, mechanisms and technologies identified and designed in previous project stages (D2.1 and D2.2) its development, based on FIWARE components, has resulted in a set of powerful digital platforms able to respond to the F4W Demo Cases requirements and needs. In this scenario, critical features such as real-time data analytics, Artificial Intelligence and Machine Learning models integration, scalability, fault tolerance or high performance are provided.

Furthermore, standardized and bidirectional communication and interoperability between water systems, including IoT sensors and devices, smart applications, or legacy systems among many other water-related actors, is enabled thanks to the definition and development of a set of harmonized water data models. Water domains such as Water Consumption, Water Quality monitoring, Wastewater or Water Distribution are covered and many entities have been modelled and standardized in close collaboration with other water-domain experts belonging to the ICT4Water cluster or the DigitalWater2020 synergy group, in which five H2020 projects are involved. Furthermore, semantic reasoning capabilities have been included in the F4W solutions by linking the data models to SAREF and SAREF4WATR ontologies. The proposed data models define the so-called **Common Information Model for Digital Water Management**, which can be reused and upscaled in any other digital water management domain case as it has been done and demonstrated in the Fiware4Water project. As a result, the F4W standardized data models have been included in the Smart Data Models Program (https://smartdatamodels.org), an open, collaborative and non-profit initiative for the curation of data models across different domains, which will ensure its maintenance and extension, and will provide visibility to the digital water society. In this scenario, this document provides several guidelines, mechanisms and resources to dynamically create new water-related data models and easily deploy and use them in a FIWARE powered ecosystem.

Additionally, contributions to the ETSI ISG-CIM standardization group are also reported, which include efforts to extend the NGSI-LD API specification or the implementation of the Digital Twin concept with NGSI-LD support.

Finally, with the aim of addressing the critical security concerns identified during the first stage of the project regarding Open Source developments, several activities have been carried out which include i) Static analysis of vulnerabilities in the deployment infrastructure (Docker), ii) Detecting and preventing hardcoded secrets (SAST), iii) Checks for common best practices in production deployments (Docker), iv) Configuration of GitHub Actions, and v) Reporting of issues.

The EU added value and policies recommendations related to this document are detailed in the Conclusion sections.

Related Deliverables

D1.4 – "Gap analysis and final Requirements", which concludes with the final requirements that need to be addressed.

D2.1 - "Specification of system architecture for water consumption and quality monitoring", which defines the general architecture and serves as a base for the different extensions.

D2.2 – "Extensions of FIWARE ecosystem with Big Data and AI frameworks", which provides big data and ai capabilities to the ecosystem, enabling the proper integration of components.

D4.1, D4.2, D4.4 and D4.5, from WP4, which describe the deployment and integration of the smart applications developed within each demo case during WP3.

Document Information

Programme	H2020 – SC0511-2018
Project Acronym	**Fiware4Water**
Project full name	FIWARE for the Next Generation Internet Services for the WATER sector
Deliverable	**D2.3: Extension of FIWARE for supporting water management and quality monitoring use-cases**
Work Package	**WP2: Architecture/Data/Ontology/API/Legacy links/Standards**
Task	Task 2.3: Common information models for water management
Lead Beneficiary	EUT
Author(s)	Lluis Echeverria (EUT)
Contributor(s)	Marc Ribalta (EUT), Aitor Corchero (EUT), Alberto Abella (FF) , Fernando López (FF), Benoit Orihuela (EGM), Chris Pantazis (NTUA), Siddharth Seshan (KWR), Gareth Lewis (UNEXE)
Quality check	Elad Salomons (EAB)
Planned Delivery Date	31/05/2022
Actual Delivery Date	30/05/2022
Dissemination Level	Public

Revision history

Version	Date	Author(s)/Contributor(s)	Notes
Draft1	01/04/2022	Alberto Abella (FF), Marc Ribalta (EUT)	ToC
Draft2	02/05/2022	Marc Ribalta (EUT), Aitor Corchero (EUT), Alberto Abella (FF)), Fernando López (FF), Benoit Orihuela (EGM), Chris Pantazis (NTUA), Siddharth Seshan (KWR), Gareth Lewis (UNEXE)	Main contributions
Draft3	20/05/2022	Lluis Echeverria (EUT), Fernando López (FF), Alberto Abella (FF)	
Review	26/05/2022	Elad Salomons (OptiWater)	
Final	27/05/2022	Lluis Echeverria (EUT)	Final

Table of content

List of figures

List of tables

List of Acronyms/Glossary

AI	Artificial Intelligence
API	Application Programming Interface
DC	F4W Demo Case
DW2020	DigitalWater2020 synergy group
DWC	Digital Water City project
ETSI	European Telecommunications Standards Institute
F4W	Fiware4Water project
LD	Linked Data
ML	Machine Learning
NGI	Next Generation Internet *The Next Generation Internet (NGI) initiative, launched by the European Commission in the autumn of 2016, aims to shape the future internet as an interoperable platform ecosystem that embodies the values that Europe holds dear: openness, inclusivity, transparency, privacy, cooperation, and protection of data.*
NGSI	Next Generation Service Interfaces
RA	Reference Architecture
RDF	Resource Description Framework
WPL	Work Packages Leaders

Introduction

The Fiware4Water smart applications are one of the most important F4W project outcomes, and have demonstrated the potential of the FIWARE ecosystem, a smart solution platform funded by the European Commission (2011-2016), in the digital water management domain.

In this context, the F4W WP2 has been focused on the materialization of the Fiware4Water Reference Architectures to provide digital support to the four F4W Demo Cases where the smart applications have been developed. Initially, the F4W RA was specified by translating requirements and identified gaps into platform functionalities and, later, such digital design was extended with BigData and AI frameworks to provide the full support to the F4W smart applications. The corresponding efforts were reported in D2.1 and D2.2 respectively.

Finally, now, at the end of the F4W project, the F4W RA have been implemented, and the F4W smart applications have been built on top of them. The present document describes this process and how the F4W RAs have been developed, taking a special focus on the following key topics:

- Systems interoperability through new standardized water data models.
  Several data models have been developed to enable standardized communication between water digital actors, thus defining the Common Information Model for Digital Water Management, which has been adopted under the Smart Data Models Program umbrella. Additionally, the implemented data models have been linked to SAREF and SAREF4WATR ontologies to extend the F4W RA functionalities with semantic reasoning capabilities, and several contributions to the ETSI ISG-CIM standardization group have been done in this context.
- RA integration with Context Information Producers and IoT devices, defined as the Southbound API.
  The Southbound interface is considered the connector bridge to the IoT platforms at the lower layers, which are the Context Information Producers. The final purpose of this interface is to translate the context information into a commonly defined data model and exchanged it through a standard open API. Security aspects are also considered.
- RA integration with high-level services for data storage, manipulation and visualization, defined as the Northbound API.
  The Northbound interface is considered the connector bridge that provides harmonized access to higher levels of an IT system, encapsulating both functionalities and services of the lower layers. Security aspects are also considered.

This document is organized as follows. Section I describes the Common Information Model for Digital Water Management and all the water data models developed in the project. Section II presents the Smart Data Models Program initiative, and how it contributes to the maintenance, curation and continuation of water data models. Section III reports on efforts devoted to adapting and aligning F4W data models to the standardized SAREF and SAREF4WATR ontologies, and section IV introduces the contributions realized to the ETSI ISG-CIM standardization group. Section V and Section VI document the Southbound and Northbound APIs specification and development respectively, for each DC, and Section VII reports efforts devoted to cybersecurity aspects. Section VIII reviews D1.4 recommendations and actions developed in this context. Finally, a Conclusion section summarises general aspects and results.

In Annex I, II, III and IV of this document, the final version of the Fiware4Water Reference Architectures are provided for the Greek, French, Amsterdam and UK Demo Cases respectively. Only the modifications introduced after D2.2 are reported.

I. The Common Information Model for a Digital Water management

Interoperability, defined as the ability of two or more software components to cooperate despite differences in language, interface, and execution platform [1], can be divided into two levels of abstraction. The first one is related to access to the data in the different resources, and the second one refers to interoperability at the semantic level. In the Fiware4Water project, the first level is led by the NGSI-LD information standard model and, the second one, by the F4W water common information model. The NGSI-LD model defines the API mechanisms to define access to water-domain data, while the F4W water common information model provides the instruments and elements for data interchange between systems. In this scenario, the common information model must define the different attributes of the data across the entities, the different attributes across the different uses, their data types, and their definitions, independently of the use case or water domain, either for wastewater management, water provisioning, etc.

I.1. Relationship between models

The common information model is structured into different (sub) data models, representing groups of attributes that encapsulate different magnitudes and elements of the water management reality. The relationship between the models is represented in two ways. On the one hand, the same attributes can be present in different data models (e.g., the location of a network element, pipe, valve, tank, etc.) and, therefore, can be queried by the users. On the other hand, the different models can have attributes representing the relationship between different elements. For example, a tank can have different sensors in it, and the identifiers of these sensors are included in the values of the tank entity.

The following sections present the set of water-related data models developed during the Fiware4Water project.

I.2. Water consumption

The water consumption data model is related to end-users' use and consumption of water. Regarding the Fiware4Water project, this data model can store not only the consumption but also the potential supply, leaks, and their associated alarms. It has been developed in the UK demo case for the control of the issues with the consumers.

It can be found in the subject dataModel.WaterConsumption, and the included data models is:

- WaterConsumptionObserved. The Smart Water Meter model captures water consumption, customer-side leak alarms, and associated flow rate originating from the smart water meters. The specification is available in the /doc folder.

I.3. Quality monitoring

Water quality monitoring is related to the control of the different parameters of potable water. These parameters include physical, biological, and chemical parameters. It has been developed in the UK DC for the control of the issues with the consumers.

It can be found in the subject dataModel.WaterQuality, and the included data models is:

- WaterQualityObserved. The Water Quality data model is intended to represent water quality parameters at a certain water mass (river, lake, sea, etc.) section. The specification is available in the /doc folder.

The developments of these data models have received contributions and feedback from other H2020 projects under the DigitalWater2020 synergy group, such as Digital Water City (DWC) and aqua3S.

I.4. EPANET water network

EPANET[1,2] [2] is a well-established software for the modelling of a water distribution network. F4W has identified and developed the required data models for the integration with such a tool, and the collection of its outputs. It has been developed in the UK DC for the integration with EPANET simulations. It can be found in the subject dataModel.WaterDistributionManagementEPANET, and the included data models are:

- Curve. This entity contains a harmonized description of a generic curve made for the Water Network Management domain. This entity is primarily associated with the water management vertical and is related to IoT applications. The specification is available in the /doc folder.
- Junction. This entity contains a harmonized description of a generic junction made for the Water Network Management domain. This entity is primarily associated with the water network management vertical and is related to IoT applications. The specification is available in the “/doc” folder.
- Network. This entity contains a harmonized description of a generic network made for the Water Network Management domain. This entity is primarily associated with the water network management vertical and is related to IoT applications. The specification is available in the doc folder.
- Pattern. This entity contains a harmonized description of a generic pattern made for the Water Network Management domain. This entity is primarily associated with the water management vertical and is related to IoT applications. The specification is available in the /doc folder.
- Pipe. This entity contains a harmonized description of a generic pipe made for the Water Network Management domain. This entity is primarily associated with the water management vertical and is related to IoT applications. The specification is available in the /doc folder.
- Pump. This entity contains a harmonized description of a generic pump made for the Water Network Management domain. This entity is primarily associated with the water management vertical and is related to IoT applications. The specification is available in the /doc folder.
- Reservoir. This entity contains a harmonized description of a generic Reservoir made for the Water Network Management domain. This entity is primarily associated with the water management vertical and is related to IoT applications. The specification is available in the /doc folder.
- SimulationResult. This entity contains a harmonized description of a generic simulation result made for the Water Network Management domain. This entity is primarily associated with the water network management vertical and is related to IoT applications. The specification is available in the /doc folder.
- SimulationScenario. This entity contains a harmonized description of a generic simulation scenario made for the Water Network Management domain. This entity is primarily associated with the water network management vertical and is related to IoT applications.

[1] https://www.epa.gov/water-research/epanet

[2] https://github.com/OpenWaterAnalytics/EPANET

- Tank. This entity contains a harmonized description of a generic tank made for the Water Network Management domain. This entity is primarily associated with the water management vertical and related IoT applications. The specification is available in the /doc folder.
- Valve. This entity contains a harmonized description of a generic Valve made for the Water Network Management domain. This entity is primarily associated with the water management vertical and is related to IoT applications. The specification is available in the /doc folder.

I.5. OpenChannel

OpenChannelManagement is related to the provisioning of water by using open channels. It has been developed in the Athens DC for the provisioning of water through open channels.

It can be found in the subject dataModel.OpenChannelManagement, and the included data models are:

- CrossSection. This entity contains a harmonized description of a generic Cross-Section made for the Raw-Water (Open Channels) System Management domain. A CrossSection defines any point of the system where raw-water properties are monitored by a device and/or computed via simulation.
- OpenChannel. This entity contains a harmonized description of a generic Channel made for Raw-Water (Open Channels) System Management domain.
- OpenChannelCurve. This entity contains a harmonized description of a generic curve made for Raw-Water (Open Channels) System Management domain.
- OpenChannelFlowRegulation. This entity contains a harmonized description of a generic simulation of a series of independent regulation structures to establish specific flow conditions in the conveyance system. It is made for the Raw-Water (Open Channels) System Management domain.
- OpenChannelJunction. This entity contains a harmonized description of a generic Junction made for the Raw-Water (Open Channels) System Management domain. A Junction defines a location where the characteristics of the channel change, two or more channels come together or split apart, amounts of water are abstracted or inserted into the system, etc.
- OpenChannelSystem. This entity contains a harmonized description of a generic system made for the Raw-Water (Open Channels) System Management domain. This entity represents either a system composed of different components (e.g., channels, junctions, cross-sections etc.) or just a component (e.g., a SluiceGate).
- RegulationStructure. This entity contains a harmonized description of a generic Regulation Structure made for the Raw-Water (Open Channels) System Management domain. The Regulation structure represents a junction-type object, controlling the water flow in the raw-water system.
- RegulationStructureSimulation. This entity contains a harmonized description of a data model for regulation structure simulation for the Raw-Water (Open Channels) System Management domain.
- SluiceGate. This entity contains a harmonized description of a generic Sluice Gate made for the Raw-Water (Open Channels) System Management domain.
- Spillway. This entity contains a harmonized description for a generic Spillway made for the Raw-Water (Open Channels) System Management domain. A Spillway represents a junction-type object, controlling the release of water from a dam or regulation structure downstream.

I.6. Wastewater

Wastewater data models refer to those models associated with the gathering and treatment of wastewater, including also the digital simulation of the treatment. It has been developed in Amsterdam DC for the control of wastewater integration with its legacy systems.

It can be found in the subject dataModel.WasteWater, and the included data models are:

- Blower. This entity contains a harmonized description of a Blower made for the Wastewater treatment domain. The entity represents a Blower that is used for aeration purposes in the wastewater treatment process. Important parameters are measured to regulate and measure the amount of airflow being provided to the aeration tank in the bioreactor. The energy consumption of a blower is also important information for real-time control and optimization of the wastewater treatment plant.
- OffGasStack. This entity contains a harmonized description of a generic Off-gas Stack made for the Wastewater treatment domain. This entity represents stacks that are present in some wastewater treatment plants where the emissions, greenhouse gases included, are emitted.
- WasteWaterJunction. This entity contains a harmonized description of a generic Junction made for the Wastewater treatment domain. Junctions could be in place in certain sections of the treatment plant. For wastewater treatment purposes, the junction is most useful if it is a location of a sensor that measures a specific variable.
- WasteWaterPlant. The data model for the wastewater treatment plant.
- WasteWaterSimulationResult. This entity contains a harmonized description of a WasteWaterSimulationResults made for the Wastewater treatment domain. The entity contains properties that are parameters which have been predicted or forecasted by models through a simulation.
- WasteWaterTank. This entity contains a harmonized description of a generic Tank made for the Wastewater treatment domain. For a given type of tank, all possible variables that can be measured are listed as properties. In the description property, the type of tank (anaerobic, pre-denitrification, nitrification etc.) can be defined.

I.7. Machine Learning

With the aim of ML algorithms integration in the data pipeline coming from legacy systems, the French DC has developed the following models:

- MLModel. The data model for compilation of the elements of an ML model.
- MLProcessing. The data model for compilation of the elements about the execution of a ML model.
- SubscriptionQuery. Subscription Query model for ML models

I.8. Additional data models

During the execution of the project, other data models have been identified and standardized. They were somehow complementary for the use cases of the Fiware4Water project, but created by the relationship with other water projects belonging to the ICT4Water cluster or the DW2020 synergy group.

I.8.1. Risk management

For the management of risks related to the water domain, but generalized enough to be used elsewhere, the subject dataModel.RiskManagement has been created and the included data models are:

- Asset. An item of value to stakeholders. An asset may be tangible (e.g., a physical item such as hardware, firmware, computing platform, network device, or other technology component) or intangible (e.g., humans, data, information, software, capability, function, service, trademark, copyright, patent, intellectual property, image, or reputation). The value of an asset is determined by stakeholders in consideration of loss concerns across the entire system life cycle. Such concerns include but are not limited to business or mission concerns.
- CyberAnalysis. An entity that represents analysis performed by digital tools to detect, for example, network traffic anomalies
- Exposure. This entity contains a harmonized description of a generic Exposure Entity made for the Risk Assessment domain.
- GISData. This entity contains a harmonized description of generic GISData made for the Risk Assessment domain.
- Hazard. This entity contains a harmonized description of a generic Hazard entity made for the Risk Assessment domain.
- Measure. Specific measure translated into actions to be performed in the different systems.
- Mitigation. The mitigation of consequences reduces the risk after an event has occurred. Therefore, this risk reduction measure is not suitable for the reduction of the likelihood of events but for the reduction of the negative consequences. Examples of consequence mitigation measures could be e.g., the construction of connection pipes to the neighbor water supplier(s) to get water from them in case of a breakdown of the own water supply, the construction of wells for an emergency supply, or signing of contracts with organizations providing small mobile emergency water treatment plants.
- NetworkServiceAlert. Data model for the qualification and reporting of those alerts in a network service and its relation to the corresponding mitigation measures.
- Risk. Effect of uncertainty on objectives. An effect is a deviation from the expected—positive and/or negative. Objectives can have different aspects (such as financial, health and safety, and environmental goals) and can apply at different levels (such as strategic, organization-wide, project, product, and process). Risk is often characterized by reference to potential events and consequences, or a combination of these. Risk is often expressed in terms of a combination of the consequences of an event (including changes in circumstances) and the associated likelihood of occurrence. Uncertainty is the state, even partial, of deficiency of information related to, understanding, or knowledge of, an event, its consequence, or likelihood.
- Vulnerability. This entity contains a harmonized description of a generic Vulnerability Entity made for the Risk Assessment domain.

I.8.2. Social media

For the dissemination and control of the water-related issues across different social networks, the datamodel.SocialMedia has been created and standardized. The included data models are:

- SMAnalysis. This entity contains a harmonized description of a generic SMAnalysis made for the Social Media domain. This entity is primarily associated with the process of analysis of Social Media applications' posts.

- SMCollection. This entity contains a harmonized description of a generic SMCollection made for the Social Media domain. This entity is primarily associated with the process of collection of Social Media posts (primarily on Twitter).
- SMPost. This entity contains a harmonized description of a generic SMPost made for the Social Media domain.
- SMRefLocation. This entity contains a harmonized description of a generic SM Reference Location (SMRefLocation) made for the Social Media domain.
- SMUser. This entity contains a harmonized description of a generic SMUser made for the Social Media domain. This entity is primarily associated with the description of a user of Social Media applications.

I.8.3. Satellite Imagery

Additionally, the ground observation required by the water management through different satellite sources led to the creation of the subject dataModel.SatelliteImagery where the following data models were identified.

- EOAnalysis. This entity contains a harmonized description of a generic EOAnalysis made for the Satellite Imagery domain. This entity is primarily associated with the process of analysis of Earth Observation applications.
- EODataHub. This entity contains a harmonized description of a generic EOInstrument made for the Satellite Imagery domain. This entity is primarily associated with the data hub related to Earth Observation Analysis applications.
- EOGeoDataLayer. This entity contains a harmonized description of a generic EOGeoDataLayer made for the Satellite Imagery domain. This entity is primarily associated with the output data layers related to Earth Observation Analysis applications.
- EOInstrument. This entity contains a harmonized description of a generic EOInstrument made for the Satellite Imagery domain. This entity is primarily associated with satellite instruments related to Earth Observation Analysis applications.
- EOProduct. This entity contains a harmonized description of a generic EOProduct made for the Satellite Imagery domain. This entity is primarily associated with satellite products related to Earth Observation Analysis applications.
- EOSatellitePlatform. This entity contains a harmonized description of a generic EOSatellitePlatform made for the Satellite Imagery domain. This entity is primarily associated with the Satellite platforms related to Earth Observation Analysis applications.

I.8.4. Anomaly

It was necessary to update the anomaly data model for the detection of issues/outliers in the values of the water data.

- Anomaly. This entity contains a harmonized description of an anomaly.

II. Smart Data Models Programme

The water common information model developed for the Fiware4Water project is a valuable outcome of the project. Like any project, it has a start and an end. Therefore, preserving this outcome requires an initiative that can curate these assets after the end of the life span of the project.

The Smart Data Models Program is a non-profit initiative for the curation of data models across different domains, specifically including the domain for water management. Thus, the different data models developed across the Fiware4Water activities have contributed to this program, so this can not only be curated but also disseminated and evolved according to the market needs. This dissemination includes the listing of the different data models, the dissemination of the use cases adopting them, and the recognition of their authors. Further details about the services included in this program that will keep the outcomes of the project are included in the next sections.

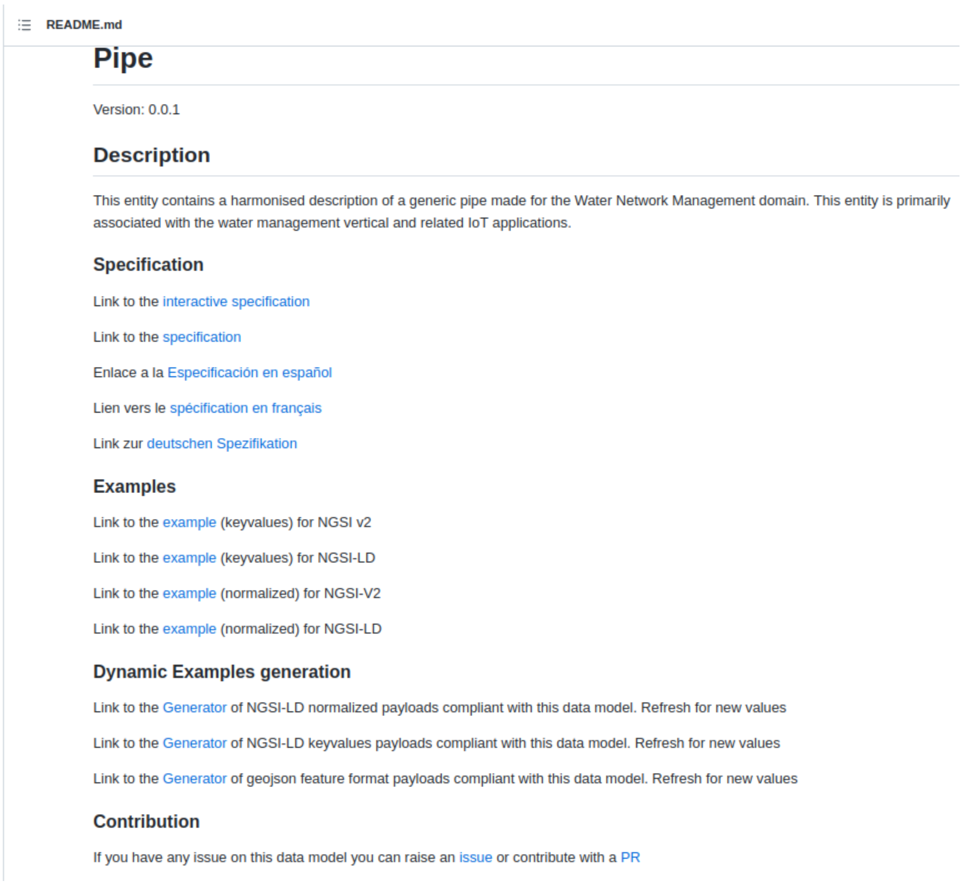
II.1. Exploitation services

For each one of the data models presented in the previous section, on the smart data models web page, https://smartdatamodels.org, several example generation services have been created.

These services include the creation of random payloads compliant with the defined data models, in the following standardized formats.

- NGSI-LD key values.
- NGSI-LD normalized.
- Geojson features.

Additionally, in the README.md of each of the data models, direct links for the generation of examples have been included. See Figure 1 regarding dynamic examples generation.



README.md

Pipe

Version: 0.0.1

Description

This entity contains a harmonised description of a generic pipe made for the Water Network Management domain. This entity is primarily associated with the water management vertical and related IoT applications.

Specification

Link to the interactive specification

Link to the specification

Enlace a la Especificación en español

Lien vers le spécification en français

Link zur deutschen Spezifikation

Examples

Link to the example (keyvalues) for NGSI v2

Link to the example (keyvalues) for NGSI-LD

Link to the example (normalized) for NGSI-V2

Link to the example (normalized) for NGSI-LD

Dynamic Examples generation

Link to the Generator of NGSI-LD normalized payloads compliant with this data model. Refresh for new values

Link to the Generator of NGSI-LD keyvalues payloads compliant with this data model. Refresh for new values

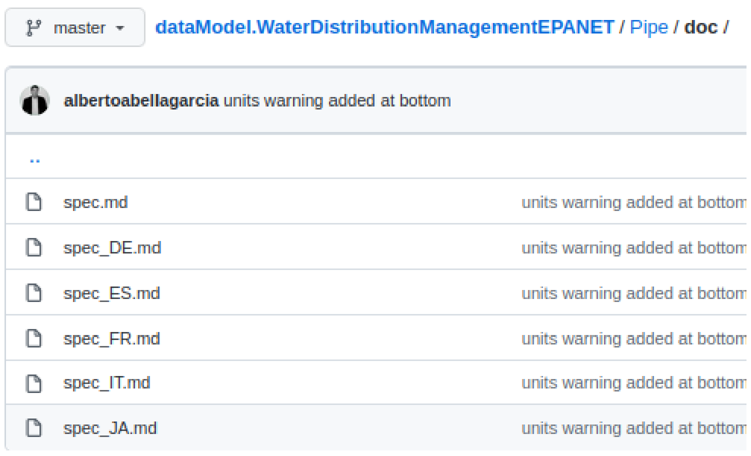
Link to the Generator of geojson feature format payloads compliant with this data model. Refresh for new values

Contribution

If you have any issue on this data model you can raise an issue or contribute with a PR

Figure 1. README.md for Pipe data model

Additionally, all the data models' specifications (Figure 2) have been translated from English into five languages German, French, Spanish, Italian, and Japanese



master ▾ dataModel.WaterDistributionManagementEPANET / Pipe / doc /

albertoabellagarcia units warning added at bottom

..

spec.md	units warning added at bottom
spec_DE.md	units warning added at bottom
spec_ES.md	units warning added at bottom
spec_FR.md	units warning added at bottom
spec_IT.md	units warning added at bottom
spec_JA.md	units warning added at bottom

Figure 2. Pipe data model specifications

II.2. Generation services

In order to be capable and simplify the process of generating the different data models, the following services were created and/or proposed.

1) **Generation from a payload (link):** it enables the drafting of a data model based on a simple JSON payload, structured in a key-values way. As a result, it generates a JSON Schema, a single source of truth of the data models, already filled with the definitions of the attributes from other data models in case they exist. It also checks if the data model could already be available, and in these cases, it provides a link to the official version.
2) **Manual creation (link):** it enables the manual creation of a data model. This service provides a text template, and relies on and redirects the user to an external site[3] to create the JSON schema file. The template includes examples for all different data types. Thus, the contributor basically only has to copy, paste and incorporate the definitions of the different attributes.
3) **Payload validator (link):** it allows the validation of the payloads and the corresponding schemas. Although there are many libraries in different programming languages[4], it is more straightforward to use an online service where the schema and the payload are pasted and validated. It also reports the errors and their causes.
4) **JSON schema validation (link)**: it enables the validation of the descriptions provided in the JSON schema. These definitions are the origin of all specifications, and therefore they need to:
   i. Be present in the doc.
   ii. Be minimally significant.
   iii. Specify what type of element is defined, either a property (containing a simple or complex value), a relationship (pointing to other elements' id), or a geoproperty (providing geolocation).

[3] http://objgen.com/json

[4] https://json-schema.org/implementations.html#validators

III. SAREF4WATR adaptation and alignment

This part of the document is mainly devoted to explaining the main elaborated procedure to link SAREF4WATR[5] with NGSI-LD and the context broker data exchange. This procedure has served to provide more meaning to the terms exposed by the context browser. Moreover, this procedure gives the possibility to import/export the resultant JSON-LD file from context-broker to enable semantic reasoning over the information due to the generation of linked data supported in the back with different ontologies and schemas (SAREF4WATR, Schema.org, etc.).

Based on these premises and objectives, this part of the document initially begins with the description of SAREF and SAREF4WATR. Later, the second part of this section is devoted to describing the Smart Data models and also the link between Smart data models and SAREF4WATR. With this information, the Smart Data models contain contextual information, terms and definitions based on different water standards.

III.1. Introduction to SAREF and SAREF4WATR

SAREF ontology was created in 2013 with the aim to share information and consensus that facilitates the matching of existing assets (standards/protocols/data models/etc.) in the smart appliances' domain and energy domain. This semantic model was created with the idea of harmonizing energy standards and having a common language to exchange information about energy systems. The common semantic model (see Figure 3) Presents an evolvement of the "Observation and Measurement" pattern to represent measures ("*Measurement*" with the corresponding properties ("*Property*") and the device or real-object ("*Feature of Interest*") that performs the corresponding measurement.

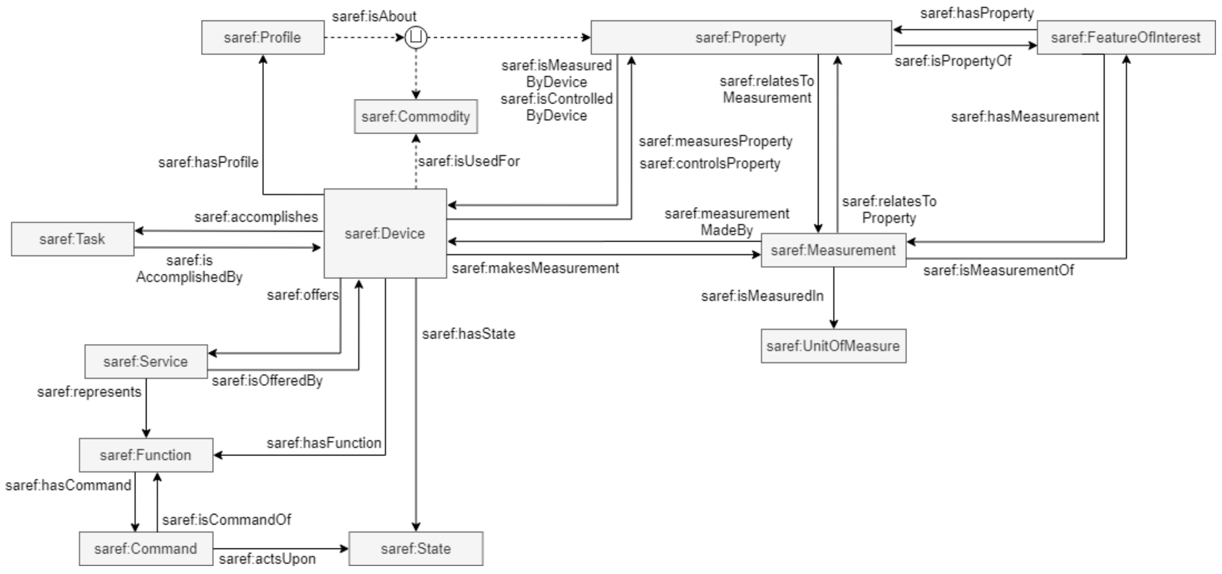


Figure 3 SAREF common semantic model

Based on this core representation, SAREF semantic model has evolved towards the interlink of cross-domain information. This interrelationship between variables and information could sustain newer decisions and holistic management. For example, water distribution decision-making actions could

[5] https://saref.etsi.org/saref4watr/v1.1.1/

take advantage of the cross-interrelation between water and energy to elaborate efficient water distribution strategies.

Considering these aspects, and combining them with the need for long-term maintenance of the ontologies, ETSI adopted the hosting and improvement of such ontologies being a standard in data modelling. Under this standardization and long-term sustainability umbrella, SAREF published an ontology to catalogue water-related observations in the water cycle. This ontology is called SAREF4WATR. The ontology has been sustained in different water-related standards (e.g., CEN/CELENEC TC 92, CEN/TC 164, CEN/TC 230, OGC, ISO, WITS, etc.) and previous semantic models (e.g., hydrOntology, InWaterSense, SWEET, SWQP, WaWO, Water Nexus Ontology, etc.).

This ontology goes deeper SAREF in the representation and determination of specific water properties and devices, as depicted in Figure 4.

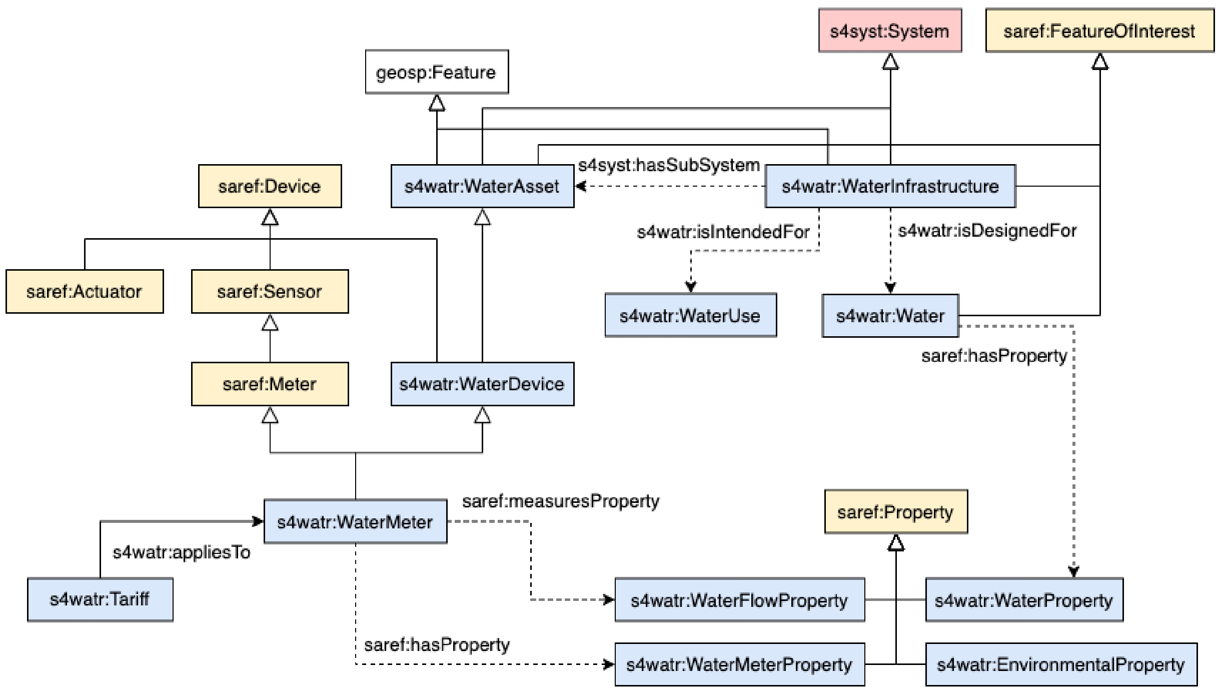


Figure 4 SAREF4WATR representation

III.2. Alignment between SAREF and Smart Data Models

The next tables describe the alignment between SAREF terms and the identified entities from the Fiware4Water project use cases. They are segmented according to the subjects defined in the Smart Data Models Program. These mappings served to link Smart Water Data models concepts with the corresponding SAREF4WATR terms and making it compatible with NGSI-LD specific concepts for water with referent semantic models and RDF knowledge graphs (in the form of context inside the JSON-LD). The following tables also includes those terms defined in SAREF4WATR extension that do not have an equivalent in the data models defined in the project (or in others in the Smart Data Models Program). In these cases, the column for Smart Data Models include a NA value.

Open Channel Management

Table 1 OpenChannelManagement model alignment

Explanations	Smart data models	SAREF4WATR
OpenChannelManagement		
A channel is a passage of water flowing in an open conduit (i.e., subject to atmospheric pressure).	Channel	s4watr:Channel
A more related class is Geometry which represents geographical region/section.	CrossSection	N/A
	HydraulicSimulationResult	N/A
	HydraulicSimulationScenario	N/A
	OpenChannelCurve	N/A
	OpenChannelJunction	N/A
	OpenChannelSystem	N/A
	RegulationStructure	N/A
	SluiceGate	N/A
	SpillWay	N/A

Wastewater

Table 2. Wastewater model alignment

Explanations	**Smart data models**	**SAREF4WATR**
WasteWater		
	Blower	N/A
	OffGasStack	N/A
	WasteWaterJunction	N/A
Where the treatment processes and operations take place	WasteWaterTank	s4watr:Tank
A treatment plant is an infrastructure to improve the quality of water to make it more acceptable for a specific end-use.	WasteWaterPlant	s4watr:TreatmentPlant

Water Consumption

Table 3. Water Consumption model alignment

Explanations	**Smart data models**	**SAREF4WATR**
WaterConsumption		
A consumption-based tariff is a tariff that is based on consumption.	WaterConsumptionObserved	s4watr:ConsumptionBasedTariff

Water Distribution Management EPANET

Table 4. Water Distribution Management EPANET model alignment

Explanations	Smart data models	SAREF4WATR
WaterDistributionManagementEPANET		
	Curve	N/A
	Junction	N/A
	Network	N/A
	Pattern	N/A
A pipe is a passage of water flowing in a closed conduit (i.e., not subject to atmospheric pressure).	Pipe	s4watr:Pipe
A pump is a device for moving water by mechanical action.	Pump	s4watr:Pump
A reservoir is an enlarged natural or artificial lake, pond or impoundment created using a dam or lock to store water.	Reservoir	s4watr:Reservoir
	SimulationResult	N/A
	SimulationScenario	N/A
A tank is a container for storing water.	Tank	s4watr:Tank
A valve is a device designed to control water flow, pressure or volume.	Valve	s4watr:Valve

Water Quality

Table 5. Water Quality model alignment

Explanations	Smart data models	SAREF4WATR
Water Quality		
Some of the waterquality parameters are defined under the classes s4watr:ChemicalProperty & s4watr:MicrobialProperty	WaterQualityObserved	N/A

Device

Table 6. Device model alignment

Explanations	Smart data models	SAREF4WATR
Device		
A tangible object designed to accomplish a particular task in households, common public buildings or offices. In order to accomplish this task, the device performs one or more functions. For example, a washing machine is designed to wash (task) and to accomplish this task it performs a start and stop function. Devices can be structured in categories (subclasses) that reflect the different domains in which a device is used, e.g., smart appliances domain (subclass FunctionRelated) vs. building domain (subclass BuildingRelated) vs. smart grid domain (subclass EnergyRelated).	Device	saref:Device

New categories can be defined, if needed, to reflect other differences, for example, different points of view, such as the point of view of the device's user vs. the point of view of the device's manufacturer. We propose a list of devices that are relevant for the purpose of SAREF, but this list can be extended.		
A device built to accurately detect and display a quantity in a form readable by a human being. Further, a device of category saref:Meter that performs a saref:MeteringFunction.	DeviceModel	saref:Meter

Key Performance Indicator

Table 7. Key Performance Indicator model alignment

Explanations	Smart data models	SAREF4WATR
KeyPerformanceIndicator		
A Key Performance Indicator (KPI) is a type of performance measurement. KPIs evaluate the success of an organization or of a particular activity in which it engages. (Definition taken from FIWARE)	KeyPerformanceIndicator	s4city:KeyPerformanceIndicator

III.3. Terms successfully mapped

The following terms have been successfully mapped between Smart Data models and SAREF4WATR.

Table 8. Terms successfully mapped

Explanations	Smart data models	SAREF4WATR
Class to group those properties related to water flow.	WaterFlow	s4watr:WaterFlowProperty
The class represents the top-level geometry type. This class is equivalent to the UML class GM_Object is defined in ISO 19107, and it is a superclass of all geometry types.	GeoProperty	geo:Geometry
	Point	sf:Point
	Polygon	sf:Polygon
A tangible object designed to accomplish a particular task in households, common public buildings, or offices. In order to accomplish this task, the device performs one or more functions. For example, a washing machine is designed to wash (task) and to accomplish this task it performs a start and stop function. Devices can be structured in categories (subclasses) that reflect the different domains in which a device is used, e.g., smart appliances domain (subclass FunctionRelated) vs. building domain (subclass BuildingRelated) vs. smart grid domain (subclass EnergyRelated). New categories can be defined, if needed, to reflect other differences, for example, different points of view, such as the point of view of the device's user vs. the	Device	saref:Device

point of view of the device's manufacturer. We propose a list of devices that are relevant for the purpose of SAREF, but this list can be extended.		
A device is responsible for moving or controlling a mechanism or system.	Actuator	saref:Actuator
A pump is a device for moving water by mechanical action.	Pump	s4watr:Pump
A valve is a device designed to control water flow, pressure or volume.	Valve	s4watr:Valve
A device that detects and responds to events or changes in the physical environment such as light, motion, or temperature changes. A device that has category saref:Sensor and performs a careful saref:SensingFunction.	Sensor	saref:Sensor
A device built to accurately detect and display a quantity in a form readable by a human being. Further, a device of category saref:Meter that performs a saref:MeteringFunction.	Meter	saref:Meter
A water meter is an instrument intended to measure continuously, memorize, and display the volume of water passing through the meter.	WaterMeter	s4watr:WaterMeter
A water device is a device that is also a water asset.	WaterDevice	s4watr:WaterDevice
A fire hydrant is a fitting in a street or other public place with a nozzle by which a fire hose may be attached to a water main.	FireHydrant	s4watr:FireHydrant
A consumption-based tariff is a tariff that is based on consumption.	WaterObservationObservation	s4watr:ConsumptionBasedTariff

III.4. Future Data Models.

The following data models have been identified and defined, but are not available yet.

Table 9. Sink Asset data model

Sink Asset	SAREF4WATR	Smart data model
An estuary is a partially enclosed coastal body of brackish water with one or more rivers or streams flowing into it, and with a free connection to the open sea.	Estuary	N/A
An ocean is a large body of saltwater.	Ocean	N/A
A river is a natural flowing watercourse, usually freshwater, flowing towards an ocean, sea, lake, or another river.	River	N/A
A sea is a body of saltwater partly or fully enclosed by land.	Sea	N/A

Table 10. Source asset data model

Source asset	SAREF4WATR	Smart data model
A glacier is a persistent body of dense ice that is constantly moving under its own weight.	Glacier	N/A
A lagoon is a shallow body of water separated from a larger body of water by barrier islands or reefs.	Lagoon	N/A
A lake is an area filled with water, localized in a basin, surrounded by land, apart from any river or other outlet that serves to feed or drain the lake.	Lake	N/A

Table 11. Storage asset data model

Storage asset	SAREF4WATR	Smart data model
An aquifer is an underground layer of water-bearing permeable rock, rock fractures, or unconsolidated materials.	Aquifer	N/A

Table 12. Transport assets data model

Transport assets	SAREF4WATR	Smart data model
An intake is an installation for obtaining water from a source of supply (river, lake, reservoir, and so on).	Intake	N/A
A main is a passage of water to flow through.	Main	N/A
A maintenance hole is an enclosure that facilitates human access to and working space for equipment.	Maintenance hole, aka Manhole	N/A
A pit is a well or hole sunk in the ground to procure, store or drain water.	Pit	N/A
A vent is the part of a system that allows air to enter a plumbing system to maintain proper air pressure and sewer gases to escape to the outside.	Vent	N/A

III.5. Terms in SAREF not available in the Smart Data Models

The following SAREF terms are not available in the Smart Data Models.

Table 13. Sink Asset SAREF term

Sink Asset	SAREF4WATR	Smart data model
A bacterial property is a property of water that is related to bacteria.	s4watr:BacterialProperty	N/A
Class to group those properties related to the environment.	s4watr:EnvironmentalProperty	N/A
A sink asset is a water asset where water sinks.	s4watr:SinkAsset	N/A

A water infrastructure is the set of facilities, services, and installations needed for water management.	s4watr:WaterInfrastructure	N/A
A water distribution system is an infrastructure to carry potable water from a centralized treatment plant or wells to water consumers in order to adequately deliver water to satisfy residential, commercial, industrial and fire fighting requirements.	s4watr:DistributionSystem	N/A
A hydroelectric power plant is an infrastructure to generate electricity by conversion of the energy of running water.	s4watr:HydroelectricPowerPlant	N/A
A monitoring infrastructure is an infrastructure to monitor water.	s4watr:MonitoringInfrastructure	N/A
A gauging station is an infrastructure to monitor and test terrestrial bodies of water.	s4watr:GaugingStation	N/A
An storage infrastructure is an infrastructure to storage both potable water for consumption, and non-potable water for use in agriculture.	s4watr:StorageInfrastructure	N/A
	saref:FeatureOfInterest	N/A
This class is used to define a particular quantity or body of water.	s4watr:Water	N/A
A water asset is a physical entity used in the process of transporting, treating, storing and distributing water.	s4watr:WaterAsset	N/A
A source asset is a water asset that is a natural source of water.	s4watr:SourceAsset	N/A
A storage asset is a water asset used to store water.	s4watr:StorageAsset	N/A
A transport asset is a water asset used to enable and support the transport and distribution of water.	s4watr:TransportAsset	N/A
Class to group those properties related to water meters.	s4watr:WaterMeterProperty	N/A
Class to group those properties related to the water.	s4watr:WaterProperty	N/A
An acceptability property is a property of water that is related to its acceptability.	s4watr:AcceptabilityProperty	N/A
A chemical property is a property of water that is related to chemical components.	s4watr:ChemicalProperty	N/A
A microbial property is a property of water that is related to microbes.	s4watr:MicrobialProperty	N/A
A bacterial property is a property of water that is related to bacteria.	s4watr:BacterialProperty	N/A
The unit of measure is a standard for measurement of a quantity, such as a Property. For example, Power is a property and Watt is a unit of power that represents a definite predetermined power: when we say 10 Watt, we actually mean 10 times the definite predetermined power called \"watt\". Our definition is based on the definition of unit of measure in the Ontology of units of Measure (OM). We propose here a list of some units of measure that are relevant for the purpose of the Smart Appliances ontology, but this list can be extended.	saref:UnitOfMeasure	N/A
An agent making an action in the context of a city. An agent could be a person, software, etc.	s4city:Agent	N/A

A Key Performance Indicator (KPI) is a type of performance measurement. KPIs evaluate the success of an organization or of a particular activity in which it engages. (Definition taken from FIWARE)	s4city:KeyPerformanceIndicator	N/A
A Key Performance Indicator assessment represents the assessment of a KPI calculated by a given agent in a given time."@en ; rdfs:label "Key performance indicator assessment	s4city:KeyPerformanceIndicatorAssessment	N/A

IV. Contributions to the ETSI ISG-CIM standardization group

This section presents the contributions made to the ETSI ISG-CIM standardization group under the scope of the F4W project. These contributions took the form of new features integrated into the NGSI-LD API specification and a description of some of the work done for the UK demo case in the Group Report focused on Digital Twins activities inside NGSI-LD enabled architectures. They are described in more detail in the following sections.

IV.1. Contributions to the NGSI-LD API specification

In the context of the project, a new feature has been presented, contributed, and integrated into version 1.4.2[6] of the NGSI-LD specification. This feature is about obtaining an aggregated representation of the temporal evolution of one or more entities. This was a feature requested by F4W demo cases, especially to online run ML models with pre-computed data instead of having to do the aggregations on their own.

In its previous version, the temporal API defined by the NGSI-LD specification only allowed to get the raw evolution of values of entities (i.e., all the timestamped values of an attribute in a time interval). This feature brings the possibility of asking for aggregated values. The resulting entity is qualified as an Aggregated Temporal Representation.

To enable this feature, two request parameters have been added to the temporal API:

- An aggregation method which specifies the function to use to aggregate the values (for instance, a sum, a mean, a max, …). More than one aggregation method can be asked for in a single temporal request.
- The duration of the period to use when applying the aggregation function (for instance, 5 minutes, 2 hours, 1 week, …). This duration is expressed using the ISO 8601 Duration Representation[7]. Put simply, it is represented as a string in the following format: *P[n]Y[n]M[n]DT[n]H[n]M[n]S* or *P[n]W*, where:
    - [n] is replaced by the value for each of the date and time elements that follow the [n],
    - P is the duration designator
    - T is the time designator.
    - For example, *P3Y6M4DT12H30M5S* represents a duration of "three years, six months, four days, twelve hours, thirty minutes, and five seconds".

[6] https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.04.02_60/gs_CIM009v010402p.pdf

[7] https://www.iso.org/standard/40874.html

Then, a list of supported aggregation methods and associated behaviour for each of them has been defined for all the types supported by the NGSI-LD API: JSON String, JSON Number, JSON Object, JSON Array, JSON Boolean, DateTime, Date, Time, URI, and Relationships. For each of these types, it has been specified which aggregation function is applicable and how it should behave. The full list of aggregation methods is the following: total count, distinct count, sum, average, minimum, maximum, standard deviation, and sum square.

To illustrate the Aggregated Temporal Representation of an entity, Figure 5 is an example of a temporal request to get the maximum and average speed of Entities of type Vehicle whose "brandName" attribute is not "Mercedes" between the 1st of August at noon and the 1st of August at 01 PM, aggregated by periods of 4 minutes:

```
GET /ngsi-ld/v1/temporal/entities/?type=Vehicle& \
    q=brandName!=Mercedes& \
    attrs=speed& \
    timerel=between& \
    timeAt=2018-08-01T12:00:00Z& \
    endTimeAt=2018-08-01T13:00:00Z& \
    aggrMethods=max,avg& \
    aggrPeriodDuration=PT4M& \
    options=aggregatedValues
Accept: application/ld+json
Link: <http://example.org/context.jsonld>; ...
```

Figure 5. Sample request to get an Aggregated Temporal Representation of an Entity

As a result of the previous request example, Figure 6 below shows the corresponding response with temporally aggregated data for the speed attribute of the Vehicle entity.

```
[
    {
        "id": "urn:ngsi-ld:Vehicle:B9211",
        "type": "Vehicle",
        "speed": {
            "type": "Property",
            "max": [
                [
                    120,
                    "2018-08-01T12:00:00Z",
                    "2018-08-01T12:04:00Z"
                ],
                [
                    100,
                    "2018-08-01T12:04:00Z",
                    "2018-08-01T12:08:00Z"
                ]
            ],
            "avg": [
                [
                    120,
                    "2018-08-01T12:00:00Z",
                    "2018-08-01T12:04:00Z"
                ],
                [
                    90,
                    "2018-08-01T12:04:00Z",
                    "2018-08-01T12:08:00Z"
                ]
            ]
        },
        "@context": [
            "http://example.org/ngsi-ld/latest/vehicle.jsonld",
            "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.3.jsonld"
        ]
    }
]
```

Figure 6 Sample data for an Aggregated Temporal Representation of an Entity

IV.2. Digital Twin implementation with NGSI-LD

Part of the work done for the UK demo case consisted in defining a digital representation of a water distribution system, based on an NGSI-LD data model, aligning it with the definition of an EPANET hydraulic simulator and finally making it available to the EPANET to run different simulations on the water distribution system. This work will be described in the upcoming Group Report of the NGSI-LD for Digital Twins working group, that will be publicly available in the coming weeks (status can be followed from the Work Item page in the ETSI portal: https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=59463).

V. Southbound interface

The concept of "Southbound Interface" is directly related to the architectural approach applied to a specific IT system and the applicative domain. In the F4W particular case, it is possible to relate this concept to the IoT – Smart Water domain in which the southbound interfaces can be considered as the connector interface to the IoT platforms at the lower layers, which are the Context Information Producers. The final purpose of this interface is to translate the context information into a commonly defined data model and exchange it through a standard open API, ETSI NGSI-LD.

Specifically, one can refer to the Southbound Interfaces in different scenarios:

- Southbound interfaces are based on the specific IoT protocols to interact with the sensors (e.g., MQTT, CoAP, XMPP, LWM2M, Sigfox, etc…).
- Southbound Interfaces can also be provided by middleware or IoT platforms API, which can have a gateway function exposing different device interfaces through a uniform one (e.g., http/rest) hiding the complexity of the underlying protocols.

In both cases, the interoperability aspect is the keystone, the wedge-shaped Stone, in a common Southbound interface definition for an open data system. In the case of F4W RA, it can be defined through the use of predefined well-known FIWARE IoT Agents or through the use of FIWARE Draco components but, in some cases, it was also needed to develop specific adapters for dedicated middleware for the harmonization of both the communication protocols and data formats.

V.1. Draco (Greek Demo Case)

NTUA and EYDAP designed and implemented a FIWARE-enabled reference architecture and deployed it to integrate seamlessly the co-existing utility's metering systems, and to enable their communication with third-party analytics and models, in a standardised way. Specifically, the FIWARE-enabled solution allowed the integration of real-time data from existing sensors of the conveyance system: 18 quality sensors in total measuring temperature, turbidity and conductivity, 20 water level meters, 9 water flow meters, 8 sluice gate opening meters, as well as daily water inflows and outflows in the 4 water treatment plants of EYDAP. Furthermore, 5 new water level meters and 1 new flowmeter have been installed in the framework of F4W, which have also been integrated with the existing data sources. To deploy the above solutions, fully operational and functional connectors have been developed to integrate FIWARE components (e.g., Orion-LD Context Broker) with the legacy system of EYDAP and the Nessie-based web platform, consisting of the two systems FIWARE-compliant.

The architecture of the FIWARE-enabled solution, developed and deployed for EYDAP, is presented in Annex I. The figure provides the schematic overview of the underlying mechanisms that enable data exchange between the components of the system, using FIWARE technology (indicated with blue borders in Figure 26). Starting from the right of the figure (yellow area, representing the legacy system of data sources at EYDAP), the Data Warehouse of EYDAP retrieves data sources from the sensors, which were hosted, before F4W, into proprietary (from different vendors) sub-systems, co-existing within EYDAP. A scheduler has been developed and configured into the different subsystems to feed continuously the Data Warehouse with new data. Data coming from the Data Warehouse is scheduled to be exported to the Main File Storage System where they become available to **the CSV IoT Agent**. The CSV IoT Agent retrieves the files from the FTP Server in a timely fashion and converts them into NGSI-LD requests for the ORION-LD Context Broker. There are two types of data the IoT Agent pushes to the Context Broker: Static and Dynamic.

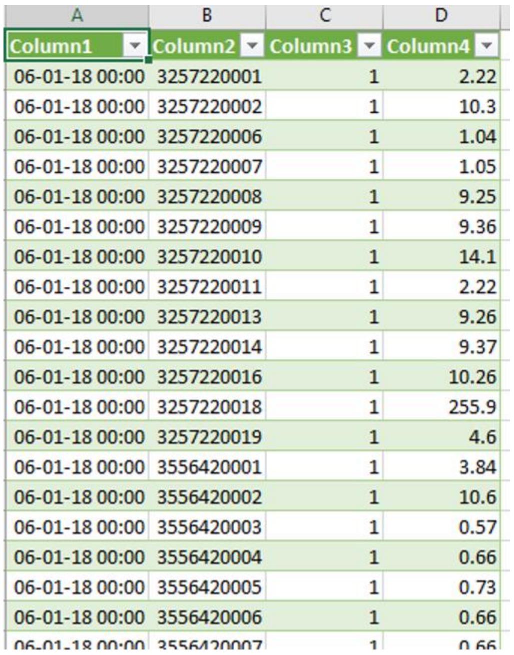
Static data refers to the characteristics of the components that comprise the EYDAP's infrastructure. For example, static data can be the sampling rate of a sensor, the location of the sensor, the geographical coordinates of a water tank or the size of it. These do not change at all or only change in very special cases. This set of data and their interlinking ontology is pushed, using NGSI-LD protocol, once and recreated automatically only in the case the Context Broker's database is lost or becomes corrupt after a system failure.

On the other hand, **dynamic data** is related to temporal information. Data that can change over time. For example, the water level at a position of the raw-water channel. Figure 7, shows an example of the file containing the dynamic data.

The IoT agent is an application designed and developed from the ground up to be future-proof and customisable to be reused in other similar projects. It is written in Python, has very few dependencies and resides in the server's memory wrapped as an Operative System (OS) service. Therefore, it runs indefinitely in the OS memory, looking for changes in the file storage. When new data arrives, it processes it very fast and notifies the Context Broker, accordingly.

The File Storage System is a server that allows for tunnelled SSH connections coming from the Data Warehouse and receives the most recent data in one big text file. Finally, the CSV IoT Agent reads this file and notifies the Orion-LD Context Broker. The Context Broker is notified on a per line-of-data basis. This means that the IoT Agent does not need to read the whole file first before starting the communication with Orion-LD context broker. This is to ensure that the Orion-LD Context Broker is getting notified as soon as new data arrives and the IoT agent retrieves it.

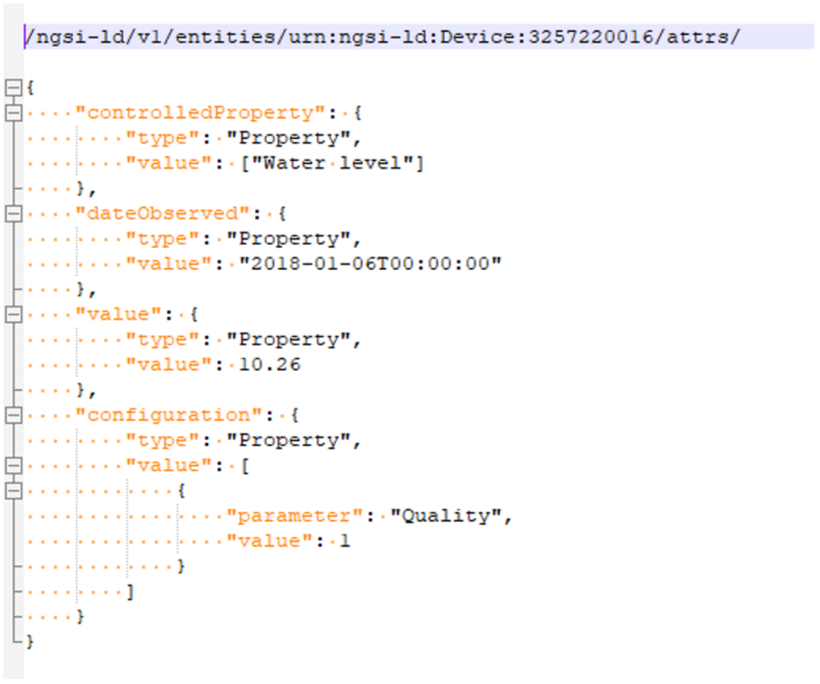
Figure 7 shows an extract of the input file coming from the Data Warehouse of the EYADAP water company.



Column1	Column2	Column3	Column4
06-01-18 00:00	3257220001	1	2.22
06-01-18 00:00	3257220002	1	10.3
06-01-18 00:00	3257220006	1	1.04
06-01-18 00:00	3257220007	1	1.05
06-01-18 00:00	3257220008	1	9.25
06-01-18 00:00	3257220009	1	9.36
06-01-18 00:00	3257220010	1	14.1
06-01-18 00:00	3257220011	1	2.22
06-01-18 00:00	3257220013	1	9.26
06-01-18 00:00	3257220014	1	9.37
06-01-18 00:00	3257220016	1	10.26
06-01-18 00:00	3257220018	1	255.9
06-01-18 00:00	3257220019	1	4.6
06-01-18 00:00	3556420001	1	3.84
06-01-18 00:00	3556420002	1	10.6
06-01-18 00:00	3556420003	1	0.57
06-01-18 00:00	3556420004	1	0.66
06-01-18 00:00	3556420005	1	0.73
06-01-18 00:00	3556420006	1	0.66
06-01-18 00:00	3556420007	1	0.66

Figure 7. Sensor data coming from the Data Warehouse

Column 2 shows the unique symbols of each sensor and column 3 displays a flag which describes the quality of the measured value (column 4). Next, the IoT Agent converts each line into a PATCH request as shown in Figure 8.



```
/ngsi-ld/v1/entities/urn:ngsi-ld:Device:3257220016/attrs/

{
    "controlledProperty": {
        "type": "Property",
        "value": ["Water level"]
    },
    "dateObserved": {
        "type": "Property",
        "value": "2018-01-06T00:00:00"
    },
    "value": {
        "type": "Property",
        "value": 10.26
    },
    "configuration": {
        "type": "Property",
        "value": [
            {
                "parameter": "Quality",
                "value": 1
            }
        ]
    }
}
```

Figure 8. IoT Agent PATCH request

It is important to note that the IoT agent holds most of its information in template files, regarding the payloads, so customization is very easy to do, and therefore, the source code of the application is publicly available at https://bitbucket.org/xpanta/csv2ngsi_ld/src/master/.

V.2. Legacy system connectors (French Demo Case)

In the context of the French Demo Case, a specific connector has been developed in order to set up a bidirectional communication between the 3S's legacy business applications (called AQDV hereafter) and the F4W NGSI-LD context broker deployed on the 3S premises.

AQDV exposes a REST API that gives access to two kinds of data:

- A list of entities called "time series": each time series is identified by an id, a name, an optional mnemonic, an optional unit code, and an optional last update time.
- Timestamped data associated with a given time series: this data can be in the past (historical data), in the future (prediction data) or continuously aggregated data (e.g., daily water consumption).

The main role of this connector is to synchronize a subset of the time series exposed by AQDV to the context broker. This process follows the following steps:

- Retrieve the list of time series to be synchronized: this list is configurable in the properties of the connector and can be dynamically adjusted.
- If a time series has never been synchronized, it has updates (in the past or in the future) since the last synchronization, or it has an allowed mutation period (for instance, for continuously aggregated data), call AQDV to get new (for historical data) or updated (for prediction data) data since the last synchronization.
- For all time-series that have to be updated, send all the time-series data to the historical storage of the NGSI-LD context broker, using one of the temporal API endpoints defined in the NGSI-LD specification and, finally, send the last value and timestamp of each time-series to the service of the context broker holding the information context and its current state.

The connector also has the task of initializing the entities corresponding to the time series in the context broker when it starts up. Each configured time series is mapped to a specific entity in the context broker, using the following principles:

- Entity id is based on the id of the time series, prefixing it with *urn:ngsi-ld:AqdvTimeSerie:*
- Entity type is fixed to the *AqdvTimeSerie* value.
- A temporal property (named *measure* by default, but it is configurable) is created based on the metadata found on the time series retrieved from AQDV. This property is the one receiving all the temporal data during the execution of the synchronization job.

The development of this connector opened up the opportunity to develop a Kotlin-based[8], Java-compatible client library for the NGSI-LD API. It is already publicly available on GitHub[9], and will soon be promoted to the FIWARE community, so that it can become an official FIWARE Generic Enabler. Indeed, these kinds of client libraries are very important to strengthen the adoption of FIWARE-based architectures, as it makes the integration with NGSI-LD context brokers a lot easier for developers (no need to reinvent the wheel in every application).

For this first version of the library, a strong emphasis has been put on type safety and coding convenience when manipulating NGSI-LD Entities and Attributes. For instance, Figure 9 reflects the entity definition process in the context of the connector developed for the French demo case.

[8] https://kotlinlang.org/

[9] https://github.com/stellio-hub/kngsild/

```
val ngsiLdEntity = NgsiLdEntityBuilder(
    scalarTimeSerie.ngsiLdEntityId(),
    AQDV_TS_TYPE,
    listOf(applicationProperties.contextBroker().context())
).addAttribute(
    NgsiLdPropertyBuilder(applicationProperties.aqdv().targetProperty())
        .withValue(0)
        .withObservedAt(ZonedDateTime.parse("1970-01-01T00:00:00Z"))
        .withUnitCode(scalarTimeSerie.unit)
        .withSubProperty("name", scalarTimeSerie.name)
        .withSubProperty("mnemonic", scalarTimeSerie.mnemonic)
        .build()
).build()
```

Figure 9. Entity definition process

This kind of builder-based pattern immediately brings the following advantages to the developers:

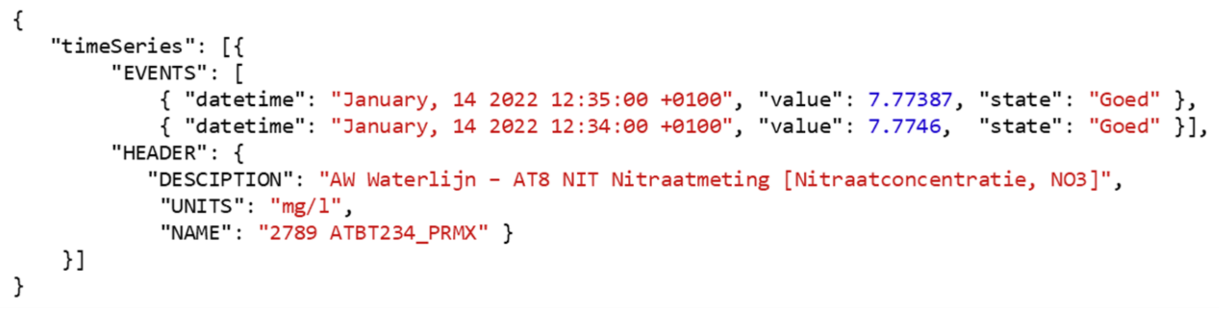
- Improved type safety: for instance, the value passed to the *observedAt* metadata cannot be anything but a date
- Ease of use: a developer no longer needs to remind of all the possible values for an attribute's metadata since any IDE will provide auto-completion on them
- Fwere errors in the construction of NGSI-LD payloads: this guarantees the correct syntax of the payload and the presence of all required attributes (for instance, it is not possible to create an entity without an id or type)

V.3. Legacy systems connectors (Amsterdam Demo Case)

In the context of the Amsterdam Demo Case, several connectors have been developed that facilitate the communication between the legacy system of the end-user Waternet (WNT), the smart applications developed and the F4W architecture components deployed. Each connector enables the integration of a data source and destination, these may be more generic or specific for a given application. The developed smart applications have been detailed in Deliverable 3.3 and the various connectors and integration to FIWARE have been discussed in Deliverable 4.4.

A connector named the FIWARE Smart Legacy Connector was developed to retrieve sensor data from WNT's legacy system and to post data to the FIWARE IoT Agent, within the F4W architecture setup for DC 3. An illustration of the overall architecture can be found in Annex III, and a detailed explanation of the architecture has been reported in Deliverable 4.4. As a result, this connector serves as the Southbound API, by providing (near) real-time data that can be fed to various smart applications via the FIWARE components. The connector is configurable to retrieve the sensor data at any schedule. To perform the retrieval, the connector also contains a mapping between the WNT legacy system sensor identifier (termed as tags within the legacy system) to their respective NGSI-LD identifies (device id and property). The connector then retrieves the sensor data via the WNT legacy system API by the system identifier, as raw data or by aggregation over a time window. The time window and the aggregation function are configurable. Different aggregation functions such as mean, interpolation, minimum, and maximum can be used. The data received by the connector is then translated from the WNT legacy system identifier to the corresponding NGSI-LD identifier. Finally, the data is then sent to the IoT Agent, at which point the FIWARE components take over the processing of the data, brokering it to the various applications linked to the specific NGSI-LD identifier. Specifically, this connector has been used to provide the necessary input data to the F4W AI-based data validation application (discussed further in Deliverable 4.4 and in Section VII of this document), but is easily configurable to retrieve more data from the WNT legacy system.

An example of retrieving specific raw data from the WNT legacy system using the FIWARE Smart Legacy Connector has been provided below. The connector initially retrieves the necessary data using the legacy system identifier in FEWS (Flood Early Warning System) format, the data protocol of the WNT legacy system, as shown in the screenshot of the payload provided in Figure 10.



```
{
   "timeSeries": [{
        "EVENTS": [
            { "datetime": "January, 14 2022 12:35:00 +0100", "value": 7.77387, "state": "Goed" },
            { "datetime": "January, 14 2022 12:34:00 +0100", "value": 7.7746,  "state": "Goed" }],
        "HEADER": {
           "DESCIPTION": "AW Waterlijn - AT8 NIT Nitraatmeting [Nitraatconcentratie, NO3]",
            "UNITS": "mg/l",
            "NAME": "2789 ATBT234_PRMX" }
    }]
}
```

Figure 10. API response containing data retrieved from the WNT Legacy System by the FIWARE Smart Legacy Connector in FEWS format.

In this example, raw sensor values of the nitrate (NO_3) concentration levels from a wastewater tank in one treatment lane of the plant are retrieved by the connector based on the legacy system identifier ("2789 ATBT234_PRMX"). This data, retrieved from the FEWS format, is then translated to NGSI-LD, as described in the Table 14 below, taking the first entry from the FEWS payload provided in Figure 10. This NGSI-LD payload is then sent to the IoT Agent JSON thereby being provided for processing within the F4W architecture.

Table 14. Translation of data into NGSI-LD by the FIWARE Smart Legacy Connector to send data to the IoT Agent JSON

IoT Agent JSON Field	Type	Value
i	Device identifier part from the NGSI-LD identifier of the device	WasteWaterTank02
t	Datetime the value was observed	20220114T123500
<Request body>	JSON of NGSI-LD properties and values	{"no3": 7.77387}

Full NGSI-LD device identifier: urn:ngsi-ld:WasteWaterTank:WasteWaterTank02. Furthermore, connectors/integrators have also been developed for individual smart applications, such as the F4W AI Soft Sensor Integrator and the F4W AI-based Data Validation Routines Manager. Primarily, these integrators play the role of Northbound APIs, and hence have been discussed further in Section VI.3. However, there are certain functionalities that these integrators perform which can be attributed to Southbound API capabilities:

- Devices are registered (if not done so before) with the FIWARE Context Broker, upon startup of the integrator.
- Subscriptions are registered (if not done so before) with the FIWARE Context Broker, to subscribe to any data point needed as input for the AI airflow soft sensor for AI-based data validation application.
- Subscriptions are registered (if not done so before) with the FIWARE Context Broker, to subscribe to any data point (input or output of the smart applications) to be received by Cygnus NGSI-LD, which will then store the data in the PostgreSQL database.

V.4. Sigfox IoT Agent (UK Demo Case)

Since the integration and customization work done with the Sigfox IoT Agent described in deliverable D2.2, the data ingestion flow is running flawlessly and the F4W platform is now providing access to the history of measures since September 2020.

In the context of the integration of the Sigfox IoT agent within the UK demo case, some contributions for fixes and improvements have been submitted and integrated. They can be found in the Sigfox IoT Agent on GitHub: https://github.com/telefonicaid/sigfox-iotagent/pulls?q=is%3Apr+author%3Abobeal+is%3Aclosed, and also in the IoT Agent Node Lib (which is the core library used by all the IoT Agents) repository in GitHub: https://github.com/telefonicaid/iotagent-node-lib/pulls?q=is%3Apr+author%3Abobeal+is%3Aclosed.

The final version of the F4W architecture is presented in Annex IV, despite it has not been modified since deliverable D2.2

VI. Northbound interface

The concept of "Northbound Interface" can be considered as the interface that provides harmonized access to higher levels of an IT system, encapsulating both functionalities and services of the lower layers. Talking specifically about Open Data APIs, this concept represents the mechanism to access the open data for further usage provided by a generic system, in the case of any F4W RA implementation. Besides, common northbound interfaces are also the pathway for communication between different systems. Therefore, it is also relevant for the standardization in the field of common security and privacy technologies.

The major issues related to northbound have been represented by the standardization of data models for both open data and metadata as well as the communication protocols (e.g., HTTP) used to exchange this information. Therefore, common "Northbound interface" amplifiers the necessity of the identification of standard ways to access and share data between open data systems.

Consequently, we have some common suggested approaches and technologies to provide an answer to the previous points through the use of REST API based on presentational state transfer technology and the Context Information Management based on a proper homogeneous representation of the data using Data Models.

Accordingly, the F4W Northbound interface provides the corresponding solutions to encapsulate the open data represented in open Data Models defined in the project and accessed through open APIs represented in ETSI NGSI-LD, and facilitate the consumption of the Context Information by both third-parties solutions and user interfaces.

VI.1. Greek Demo Case

Figure 11 illustrates the final dataflow implementation. In the Greek DC, the FIWARE Draco service is waiting for ORION-LD notifications (ListenHTTP Processor). When the notification is received by the processor it forwards it to the NGSIToPostgreSQL processor to be converted into SQL commands and inserted in the database for permanent storage. There is also the LogAttribute processor which exports various messages and data to a log file for debugging and maintenance reasons. The FIWARE Draco service receives data in the format shown below and forwards it to the NGSIToPosgreSQL processor.

The most important module in the configuration shown in Figure 11 is the NGSIToPostgreSQL processor. It is designed to persist NGSI-like context data events within a PostgreSQL server. Usually, such context data is notified by a FIWARE Context Broker (Orion-LD CB in this case) instance but could be any other system speaking the NGSI language. Independently of the data generator, NGSI context data is always transformed into internal NGSIEvent objects at Draco sources. In the end, the information within these events must be mapped into specific PostgreSQL data structures.

Each piece of "User Data" (i.e., data that the user brings into NiFi for processing and distribution) is referred to as a FlowFile. A FlowFile is made up of two parts: Attributes and Content. The Content is the User Data itself. Attributes are key-value pairs that are associated with the User Data. Processors are modules that are responsible for processing and transporting the FlowFile. The Processor is the NiFi (the underlying tool of the FIWARE Draco) component that is responsible for creating, sending, receiving, transforming, routing, splitting, merging, and processing FlowFiles. It is the most important building block available to NiFi users to build their dataflows.

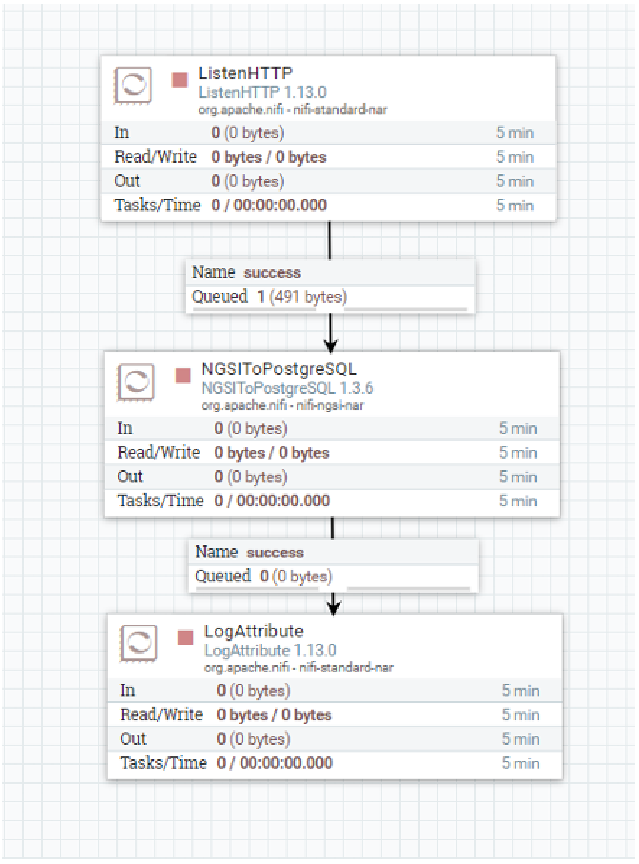
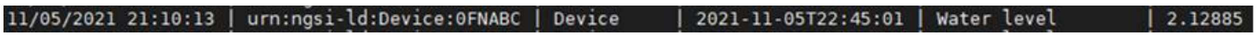

ListenHTTP
ListenHTTP 1.13.0
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
Name success
Queued 1 (491 bytes)
NGSIToPostgreSQL
NGSIToPostgreSQL 1.3.6
org.apache.nifi - nifi-ngsi-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
Name success
Queued 0 (0 bytes)
LogAttribute
LogAttribute 1.13.0
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Figure 11. FIWARE Draco implementation

FIWARE Draco is listening for subscription notifications and processes and stores them as timestamped value pairs in the PostgreSQL database. An example of a subscription notification converted into SQL data is shown in Figure 12.



```
11/05/2021 21:10:13 | urn:ngsi-ld:Device:0FNABC | Device     | 2021-11-05T22:45:01 | Water level         | 2.12885
```

Figure 12. NGSI-LD data in the PostgreSQL Database

VI.2. French Demo Case

Following the experiments discussed in deliverable D2.2, and the first conclusions drawn in this deliverable, a new component has been developed to integrate ML models in the F4W RA data pipeline. In parallel, BigData mechanisms have been also included in the architecture to demonstrate the scalability, resiliency, high performance and fault-tolerant capabilities of FIWARE components.

ML models serving

The BentoML component has been integrated and deployed inside the F4W-RA. Indeed, it has proven to be capable of easily embedding an ML model prepared with any major ML framework (e.g. scikit[10], Pytorch[11] or TensorFlow[12]). What's more, in the context of the French DC, it has even been possible to integrate a model developed with Matlab[13] (the technology used by 3S scientific team to prepare and train ML models), then later exported as a Python package.

Following the plan described in section III.2 of the deliverable D2.2, the major work has been conducted to develop a wrapper around the BentoML component in order to make it NGSI-LD compliant. The general position of this new component inside the F4W-RA is illustrated in Figure 13.

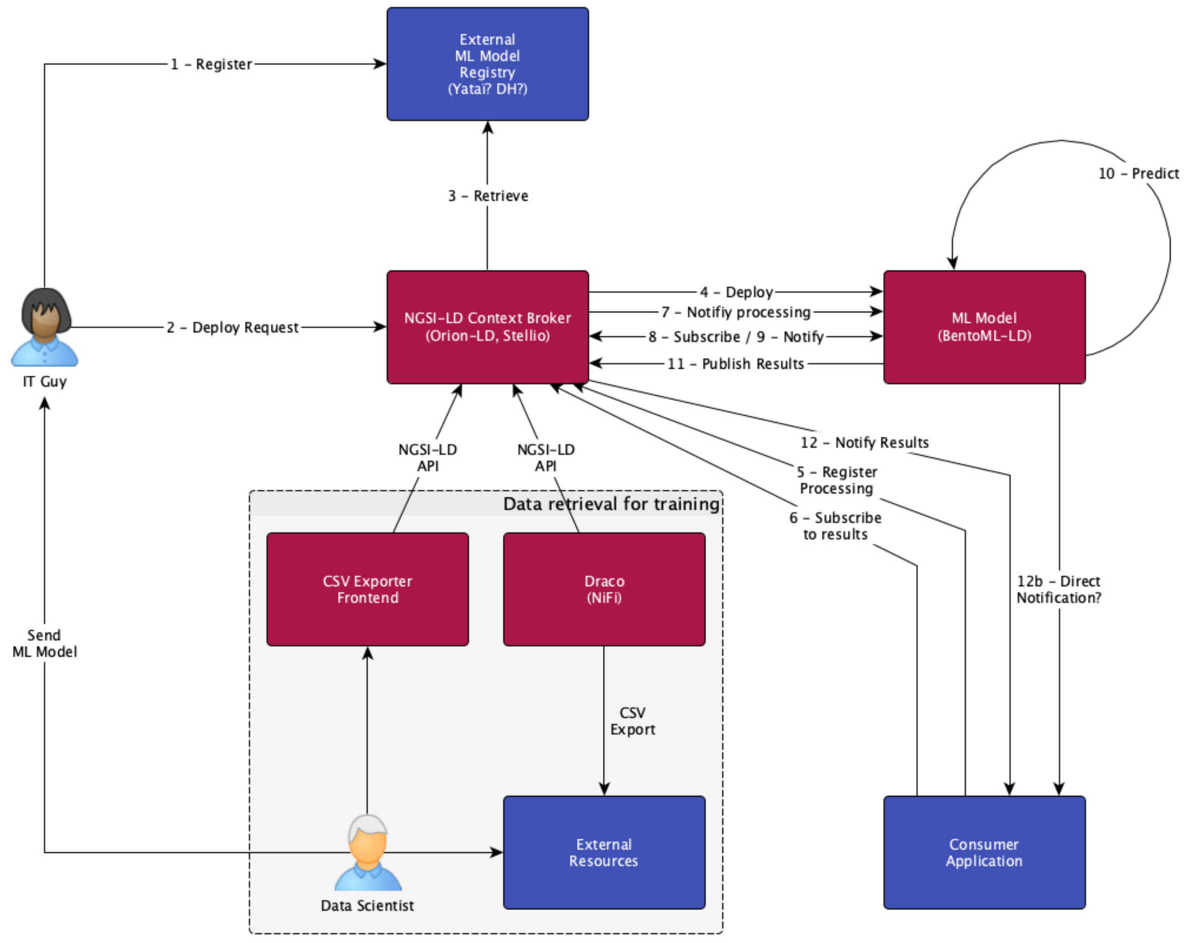


Figure 13 Global overview of a MLaaS in a FIWARE architecture

[10] https://scikit-learn.org/stable/
[11] https://pytorch.org/
[12] https://www.tensorflow.org/
[13] https://www.mathworks.com/products/matlab.html

The work done in the last months has been focused on steps 5 to 12 in Figure 13 (the runtime aspects of a deployment of an ML model):

- Pre-requisite: an ML model has been previously packaged into BentoML as shown in Figure 14, and the component is registered and deployed inside the platform. The corresponding NGSI-LD entity is created in the NGSI-LD context broker.

```
{
  "id":"urn:ngsi-ld:MLModel:01",
  "type":"MLModel",
  "name": "Water consumption prediction",
  "description": "Predicts water consumption based on …",
  "dockerImage": "Docker image containing the model",
  "algorithm": "k-means",
  "version": 12,
  "inputAttributes": ["minFlow", "maxFlow", "waterConsumption"],
  "outputAttributes": ["consumptionNextDay", "consumptionNextWeek"]
}
```

Figure 14 Example of an MLModel entity representing a deployed Machine Learning model

- Step 5 - Register a processing (represented as an MLProcessing entity, Figure 15) in the context broker: this is where a user (typically through a consumer application) asks for a processing (prediction, anomaly detection, …) on some entities with a specific ML model available inside the platform.

```
{
  "id":"urn:ngsi-ld:MLProcessing:01",
  "type":"MLProcessing",
  "refMLModel": "urn:ngsi-ld:MLModel:01",
  "subscriptionData": {
    "entities": [{
      "type": "WaterConsumption"
    }],
    "q": "refCity==urn:ngsi-ld:City:Valbonne"
  }
}
```

Figure 15 Example of an MLProcessing entity creating a link between an ML model and some input entities

- Step 6 – Subscribe to results: in parallel to the previous registration, the user subscribes (via the creation of an NGSI-LD subscription) to changes occurring in the entity where the ML model publishes results of the processing (not required but it allows for reactive notifications of new data)
- Steps 7 and 8 – Notify a processing and subscribe to input data: the BentoML component is notified of the processing declared in step 5. In return, it subscribes to the entities declared in the processing (in the previous figure, it subscribed to entities of type *WaterConsumption*)
- Steps 9, 10 and 11 – Receive data and publish results: when the subscription created in step 8 is matched, a notification is received by the BentoML component. It then uses the data contained in the entities that are part of the notification (or it can also get historical data from these same entities if it has been configured in this way) to perform its specific processing. Once a result is ready, it is published into the target entity configured to hold the results of the processing.

- Step 12 – Notify results: the consumer application, who has subscribed to the results of the processing in step 6, is notified and has direct access to the new data.

As previously described in section I.7 – Machine Learning, two data models (MLModel and MLProcessing) have been designed and implemented to enable this type of integration between a FIWARE context broker and a Machine Learning component.

Future work after the end of the project is also planned for this component to:

- Prepare it for integration into the incubator of the FIWARE ecosystem of Generic Enablers.
- Extract the NGSI-LD wrapper layer into a separate component so that it can be used in any other Machine Learning based deployments, without any dependency on the BentoML component.

BigData components

BigData capabilities have been incorporated and demonstrated through the integration of the Apache Spark[14] framework, a unified engine for large-scale data analytics, which includes functionalities for ML models training and serving.

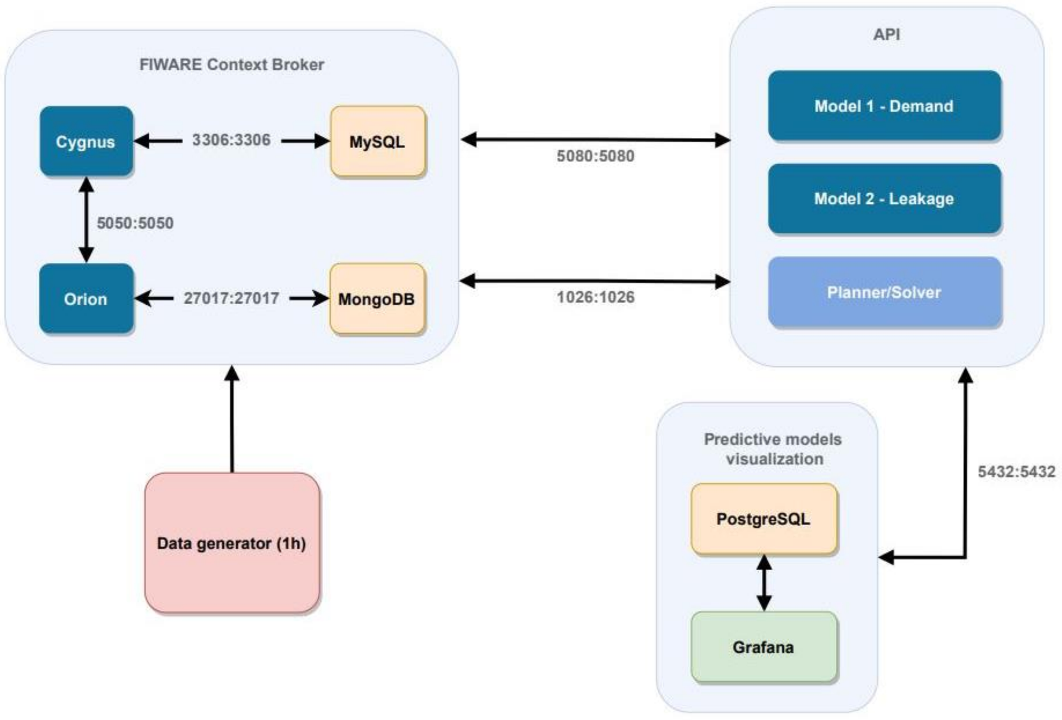


Figure 16. BigData F4W framework extension

Details on ML models implemented with Apache Spark are provided in deliverable D3.2, and the final solution (Figure 16) is documented in deliverable D4.2, which includes all the integrated BigData FIWARE components and the developed data pipelines.

[14] https://spark.apache.org/

VI.3. Amsterdam Demo Case

In the section V.3, the FIWARE Smart Legacy Connector developed as the Southbound API for the Amsterdam DC is discussed. Furthermore, for the integration and real-time implementation of the smart applications developed for the DC (discussed in Deliverable 3.3), individual Northbound APIs were developed per application. In the following section, these integrators are briefly discussed. A detailed explanation of the integrators can be found in Deliverable D4.4.

Fiware4Water AI Airflow Soft Sensor Integrator

Over and above the FIWARE Smart Legacy Connector, the WNT legacy system is also able to perform API calls to push and retrieve data to or from the F4W architecture (as illustrated in Annex III). For this given integrator, the legacy system is configured to push data specific to blowers that are present in Amsterdam West WWTP, which provides the given aeration to the bioreactor units. This data is then sent to the Fiware4Water AI Airflow Soft Sensor Integrator, which serves as an API, and is hosted along with the FIWARE components and the AI Airflow Soft Sensor application. Within the integrator, the WNT legacy system identifiers for the given sensors have been mapped to the NGSI-LD identifiers. The integrator can then send the data to the IoT Agent JSON. The data is then stored by FIWARE components (specifically in PostgreSQL using Cygnus NGSI-LD) and is then sent to the AI Airflow Soft Sensor application. The application, in return, provides the computed soft sensor values which are sent back into FIWARE through the IoT Agent JSON. Subsequently, this specific integrator also ensures that the computed soft sensor values are returned to the legacy system and are then stored in the legacy database.

Fiware4Water AI-Based Data Validation Routines Manager

The F4W AI-based data validation (DV) application has been deployed and integrated fully within the FIWARE architecture and is largely coordinated by the Southbound API integrator been developed which has been named as F4W AI-DV routines manager. Initially, the raw data signals that are stored in the legacy system are accessed by the FIWARE Smart Legacy Connector (Section V.3) which then sends the data to the FIWARE Context Broker, via the IoT Agent JSON, and finally to the PostgreSQL database. The routines manager simultaneously receives the data from the context broker through a subscription notification. Upon receiving a notification and at a given time interval, the routines manager queries the PostgreSQL database (which holds all historic data) to retrieve historical data. The routine manager then calls the relevant methods that are part of the AI-based data validation application, while also inputting the historical data, that is needed as inputs by AI models. Finally, the outputs from the data validation application available to the routines manager are then sent to the IoT Agent JSON, which eventually is also stored in the PostgreSQL database. This data is then visualised in a Grafana dashboard. In Table 15 provided below, an example of JSON payload has been shown, where the outputs from the application containing the reconciled values for a nitrate sensor signal as sent to the IoT Agent JSON.

Table 15. NGSI-LD Payload used by the AI-based Data Validation Routines Manager to send the data validation application output to the IoT Agent JSON

IoT Agent JSON Field	Type	Value
i	Device identifier part from the NGSI-LD identifier of the device	NitrateReconciled
t	Datetime the value was observed	20220114T123500
<Request body>	JSON of NGSI-LD properties and values	{"no3": 7.4580}

Full NGSI-LD device identifier: urn:ngsi-ld:WasteWaterTank:NitrateReconciled

VI.4. UK demo case

As already described in the deliverable D2.2, a real-time data visualisation solution based on the Grafana open-source project has been integrated for the UK DC. As shown in the figures below, the solution has been running flawlessly for almost two years and is providing end-users with useful data retrieved from an NGSI-LD enabled context broker.

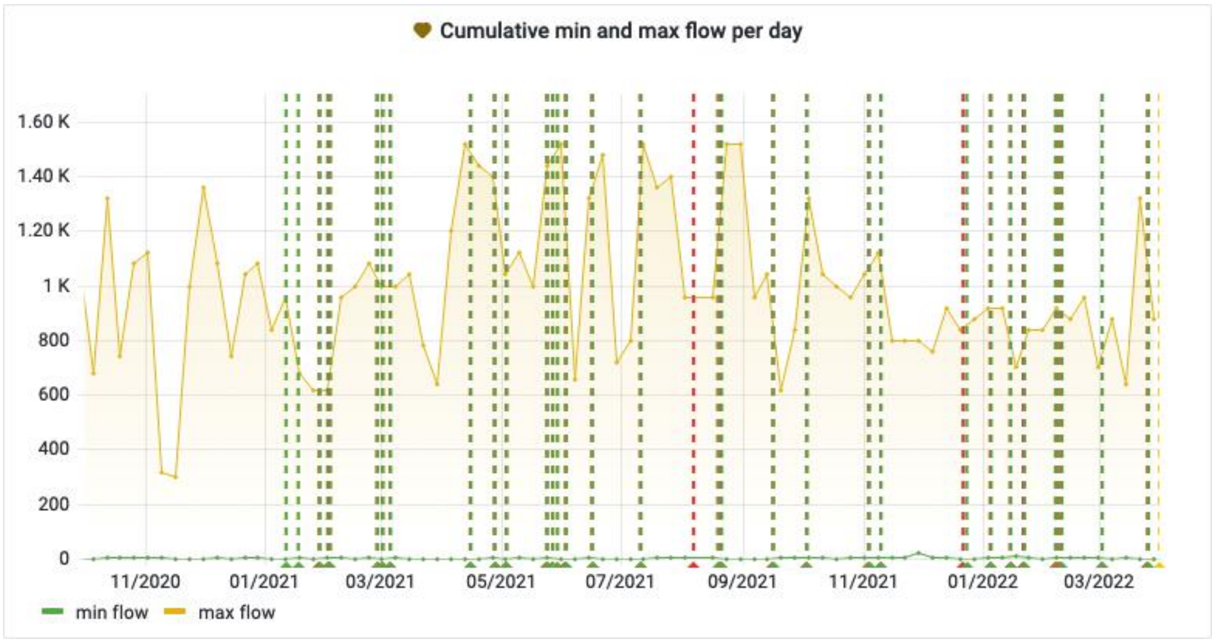


Figure 17. Visualization of cumulative min and max flows since September 2020

In the Figure 17, vertical bars represent an automatic alerting on missing data so that any failure can be known and acted upon as early as possible.

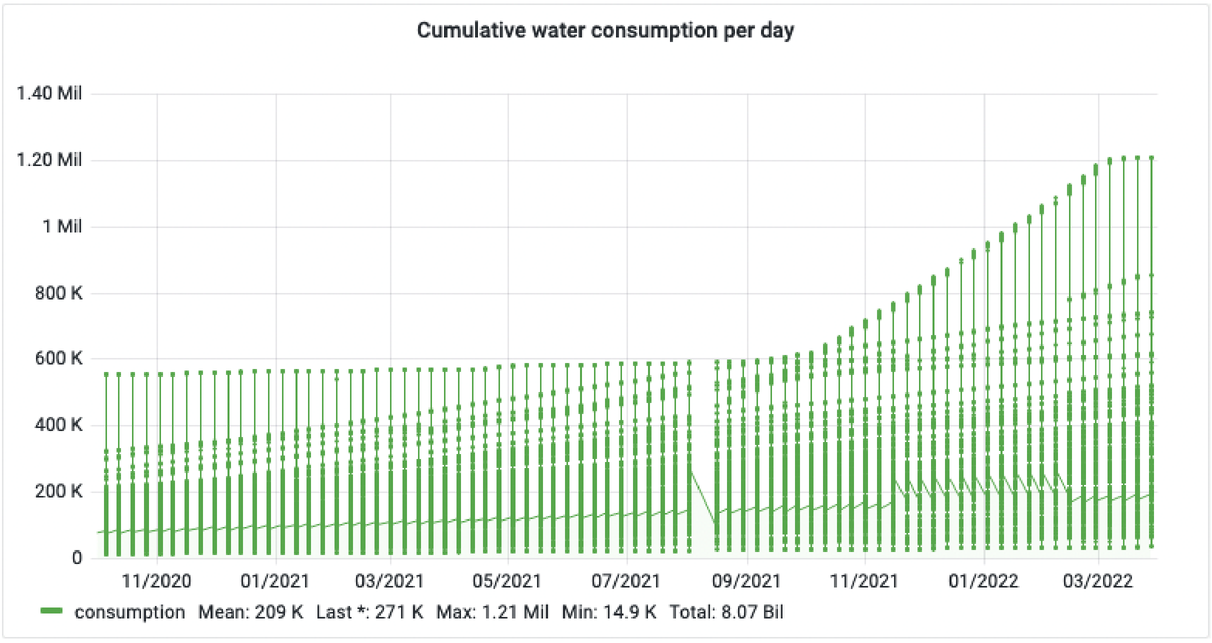


Figure 18. Visualization of cumulative daily water consumption since September 2020

VII. Cybersecurity activities

The results of the questionnaires developed in the WP1 produce a specific requirement in terms of identified lack of IT security due to "People think because data models or source code are available online it's easier to hack it". As a result, an action plan was developed to provide a response to this identified gap.

In the comments, it was mentioned from one side the data model security and the open-source code.

VII.1. Data Model security activities

Opening the specification has two effects like the two sides of the same coin. On the one hand, malicious attackers can study the potential vulnerabilities and develop tools for their exploitation. On the other hand, openness and multiple users allow collaboration on the protection and quick fix of the weak elements. Additionally, the creation of JSON schema validators allows to implement format validations of the payloads before being ingested on the platform. Finally, there is an emerging demand for the extension of the data models with signature attributes (frequently based on blockchain technology) to 'sign' the content of the different payloads and restrict the potential modifications without the right permissions.

One example of successfully implementing security measures is present in the data model of VaccinationCertificate.

VII.2. Open Source security activities

During the execution of the WP2 activities, and associated with task T2.1, there were developed a set of activities to provide a response to the previous comment. As a result, a continuous scanning was developed monthly on each of the FIWARE Generic Enablers involved in the F4W architecture (currently extended to all FIWARE Community). This activity involves the security scanning of the

Docker[15] Images to find some security issues as well as SAST (Static Application Security testing) analysis of the GitHub code of the FIWARE Generic Enablers. Additionally, another scan was developed to provide best practices in the deployment of a common application using those FIWARE Generic Enablers. The summary of the activities can be found in the GitHub repository (https://github.com/flopezag/fiware-security).

The activities were focused on several lines, summarised as:

- Static analysis of vulnerabilities in a Docker container.
- Detecting and preventing hardcoded secrets (SAST).
- Checks for common best practices around deploying Docker containers in production.
- Configuration of GitHub Actions.
- Reporting of issues.

VII.2.1. Static analysis of vulnerabilities in docker container

The main objective of this activity is to get detailed information about any vulnerability that can be found in the Docker image with the tag *latest* during the development stage of the corresponding components. For this purpose, we take care of two open-source components (Clair[16] and Anchore[17]) that inspect containers layer-by-layer for well-known security issues, based on the Common Vulnerabilities and Exposures database (CVE) and similar databases from Red Hat®, Ubuntu, and Debian OS distributions. We follow the recommendation to make the analysis using two different solutions and compare the vulnerabilities obtained with them. This, therefore, provides a better overview of the security issues on our components. Figure 19 and Figure 20 show an example of the reporting data on some components.

```
[
  {
    "image": "fiware/orion-ld:1.0.0",
    "vulnerabilities": [
      {
        "featurename": "cyrus-sasl-lib",
        "featureversion": "2.1.27-5.el8",
        "vulnerability": "RHSA-2022:0658",
        "namespace": "centos:8",
        "description": "The cyrus-sasl packages contain the Cyrus implementation of Simple
Authentication and Security Layer (SASL). SASL is a metho
d for adding authentication support to connection-based protocols. Security Fix(es): *
cyrus-sasl: failure to properly escape SQL input allows an at
tacker to execute arbitrary SQL commands (CVE-2022-24407) For more details about the
security issue(s), including the impact, a CVSS score, acknowle
dgments, and other related information, refer to the CVE page(s) listed in the References
section.",
        "link": "https://access.redhat.com/errata/RHSA-2022:0658",
        "severity": "High",
        "fixedby": "0:2.1.27-6.el8_5"
      },
…
]
```

Figure 19 Example of identified vulnerability in Orion-LD with Clair component

[15] https://www.docker.com/

[16] https://github.com/quay/clair

[17] https://docs.anchore.com/current/

```
{
    "imageDigest":
"sha256:9a388506dec69269e5e54144cc49fa48d4f756de57e683bd4c010a6f8642783d",
    "vulnerabilities": [
        {
            "feed": "vulnerabilities",
            "feed_group": "alpine:3.12",
            "fix": "6.2_p20200523-r1",
            "nvd_data": [
                {
                    "cvss_v2": {
                        "base_score": 6.8,
                        "exploitability_score": 8.6,
                        "impact_score": 6.4
                    },
                    "cvss_v3": {
                        "base_score": 8.8,
                        "exploitability_score": 2.8,
                        "impact_score": 5.9
                    },
                    "id": "CVE-2021-39537"
                }
            ],
            "package": "ncurses-libs-6.2_p20200523-r0",
            "package_cpe": "None",
            "package_cpe23": "None",
            "package_name": "ncurses-libs",
            "package_path": "pkgdb",
            "package_type": "APKG",
            "package_version": "6.2_p20200523-r0",
            "severity": "High",
            "url": "http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-39537",
            "vendor_data": [],
            "vuln": "CVE-2021-39537",
            "will_not_fix": false
        },
…
]
```

Figure 20 Example of identified vulnerability in Stellio Timescale Postgis with Anchore component

As a result of the execution, each of the FIWARE GE owners receives a summary of the identified issue with the link to the CVE ID for more details as well as a Common Vulnerability Scoring System (CVSS) v2.0 and v3.X provided by the National Vulnerability Database[18] (NVD). NVD provides a qualitative severity rating categorized by "Low", "Medium", and "High" for CVSS v2.0 and “None”, "Low", "Medium", "High", and “Critical” CVSS v3.0 score ranges how we can see in Table 16.

Table 16. NVD CVSS rating scores

CVSS v2.0 Ratings			**CVSS v3.0 Ratings**	
Severity	**Base Score Range**		**Severity**	**Base Score Range**
			None	0.0
Low	[0.0, 3.9]		Low	[0.1, 3.9]
Medium	[4.0, 6.9]		Medium	[4.0, 6.9]
High	[7.0, 10.0]		High	[7.0, 8.9]
			Critical	[9.0, 10.0]

[18] https://nvd.nist.gov/

VII.2.2. Detecting and preventing hardcoded secrets (SAST)

Another step in the management of Cybersecurity aspects of the component is the Static Application Security Testing (SAST) of the GitHub repositories. During the execution of this process, a monthly analysis of the Repository content and Git Log details are realized to find hardcoded secrets such as passwords, API Keys, and tokens. For this purpose, we have integrated into the scan process the Gitleaks[19] tool with recommendations to install pre-commit option in your repositories to allow controlling the security scanning of the code before pushing it into GitHub.

The analysis produces a report that is informed to the FIWARE Generic Enablers owners with details of the hardcoded secrets in files in which it was found, as well as the author of this commit as can be seen in the following JSON result.

```
[
 {
  "Description": "RSA private key",
  "StartLine": 1,
  "EndLine": 1,
  "StartColumn": 1,
  "EndColumn": 31,
  "Match": "-----BEGIN RSA PRIVATE KEY-----",
  "Secret": "-----BEGIN RSA PRIVATE KEY-----",
  "File": "security/localhost.key",
  "Commit": "2377a3245a7f66cae3caf434c0bdaa0fc338676d",
  "Entropy": 0,
  "Author": "Ken Zangelin",
  "Email": "kzangeli@tid.es",
  "Date": "2014-04-17T10:21:41Z",
  "Message": "the context broker now supports https",
  "Tags": [],
  "RuleID": "RSA-PK"
 }
]
```

Figure 21 Example of SAST scan over Orion-LD

It is important to mention that possible security issues are identified, but it needs the review by the owners in order to verify that they are only false positives. For example, in Figure 21, a private key in the Orion-LD repository, but this key is being used for testing purposes of the component therefore the secret was marked as a false positive by the component owner.

VII.2.3. Checks for common best practices around deploying Docker containers in production

The purpose of this analysis is to check well-known best practices for deploying Docker Containers in a production environment using a pre-defined Docker Compose file for the FIWARE Generic Enablers. These tests are based on the CIS Docker Benchmark[20] v1.3.1.

```
{
  "dockerbenchsecurity": "1.4.0",
  "start": 1650363731,
  "tests": [
    {"id": "4", "desc": "Container Images and Build File",  "results": [
      {"id":  "4.1",  "desc":  "Ensure  that  a  user  for  the  container  has  been  created
(Scored)", "result": "INFO", "details": "No containers running"},
```

[19] https://github.com/zricethezav/gitleaks
[20] https://www.cisecurity.org/benchmark/docker

```
      {"id": "4.2", "desc": "Ensure that containers use only trusted base images (Not Scored)", "result": "NOTE"},
      {"id": "4.3", "desc": "Ensure that unnecessary packages are not installed in the container (Not Scored)", "result": "NOTE"},
      {"id": "4.4", "desc": "Ensure images are scanned and rebuilt to include security patches (Not Scored)", "result": "NOTE"},
      {"id": "4.5", "desc": "Ensure Content trust for Docker is Enabled (Scored)", "result": "WARN"},
      {"id": "4.6", "desc": "Ensure that HEALTHCHECK instructions have been added to container images (Scored)", "result": "PASS"},
      {"id": "4.7", "desc": "Ensure update instructions are not use alone in the Dockerfile (Not Scored)", "result": "PASS"},
      {"id": "4.8", "desc": "Ensure setuid and setgid permissions are removed (Not Scored)", "result": "NOTE"},
      {"id": "4.9", "desc": "Ensure that COPY is used instead of ADD in Dockerfiles (Not Scored)", "result": "PASS"},
      {"id": "4.10", "desc": "Ensure secrets are not stored in Dockerfiles (Not Scored)", "result": "NOTE"},
      {"id": "4.11", "desc": "Ensure only verified packages are are installed (Not Scored)", "result": "NOTE"}
    ]},
    {"id": "5", "desc": "Container Runtime",  "results": [
    ]},
    {"id": "6", "desc": "Docker Security Operations",  "results": [
      {"id": "6.1", "desc": "Ensure that image sprawl is avoided (Not Scored)", "result": "INFO", "details": "0 active/1 in use"},
      {"id": "6.2", "desc": "Ensure that container sprawl is avoided (Not Scored)", "result": "INFO", "details": "0 total/0 running"}
    ]}
  ], "checks": 13, "score": 0, "end": 1650363732
}
```

Figure 22 Example Docker Bench Security analysis (FIWARE Sigfox)

VII.2.4. Configuration of GitHub Actions

In order to provide actual security information about released images, integration with GitHub-actions[21] was implemented. The action is active for the FIWARE-Catalog[22] and conducts static vulnerability analyses of released oci-images, using the Clair-scanner[23].

The images to be scanned are retrieved from the catalog's submodules, which have to include a JSON-structured config-file (https://fiware-requirements.readthedocs.io/en/latest/docker_templates/index.html) in a well-known location. The action retrieves the container-registry information from that file and combines it with the SemVer-compliant releases of the component, available through the github-releases API (see https://github.com/FIWARE-Ops/get-images-from-submodules). The images are tested every night.

Additionally, we have provided two sample GitHub Action Workflows in the fiware-security repository (https://github.com/flopezag/fiware-security/tree/master/sample-github-actions), in order to facilitate the configuration of the components in their own repositories. For example, to enable an Anchore Scan of a Docker image based on node-slim:

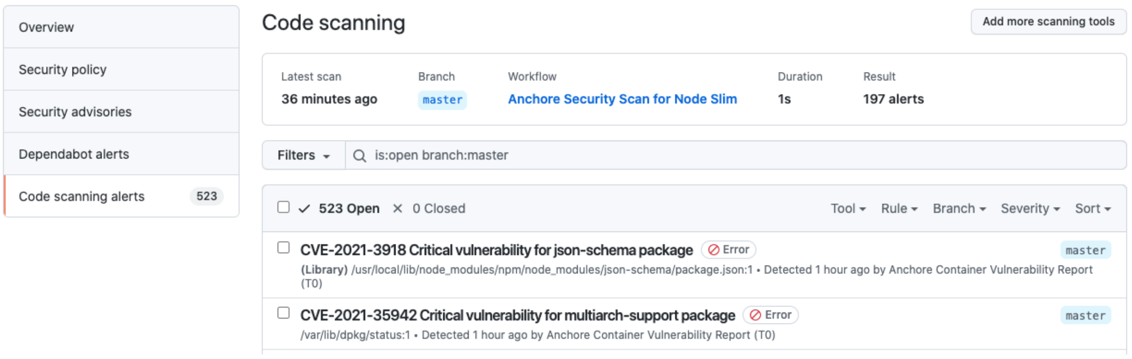
[21] https://github.com/features/actions

[22] https://github.com/FIWARE/catalogue

[23] https://github.com/quay/clair

- Copy the anchore-node-slim.yaml file (https://github.com/flopezag/fiware-security/blob/master/sample-github-actions/anchore-node-slim.yml) to .github/workflows/anchore-node-slim.yml .
- Amend the Dockerfile context location (https://github.com/flopezag/fiware-security/blob/master/.github/workflows/anchore-node-slim.yml, line 34) if necessary - the example assumes a folder called docker is used.
- After committing and pushing the file, run the new GitHub Action manually (https://docs.github.com/en/actions/managing-workflow-runs/manually-running-a-workflow).

A security report will be displayed on https://github.com/<Owner>/<Repository>/security/code-scanning?query=is%3Aopen+branch%3Amaster+severity%3Aerror



Code scanning

Latest scan: 36 minutes ago | Branch: master | Workflow: Anchore Security Scan for Node Slim | Duration: 1s | Result: 197 alerts

is:open branch:master

523 Open × 0 Closed

CVE-2021-3918 Critical vulnerability for json-schema package Error master
(Library) /usr/local/lib/node_modules/npm/node_modules/json-schema/package.json:1 • Detected 1 hour ago by Anchore Container Vulnerability Report (T0)

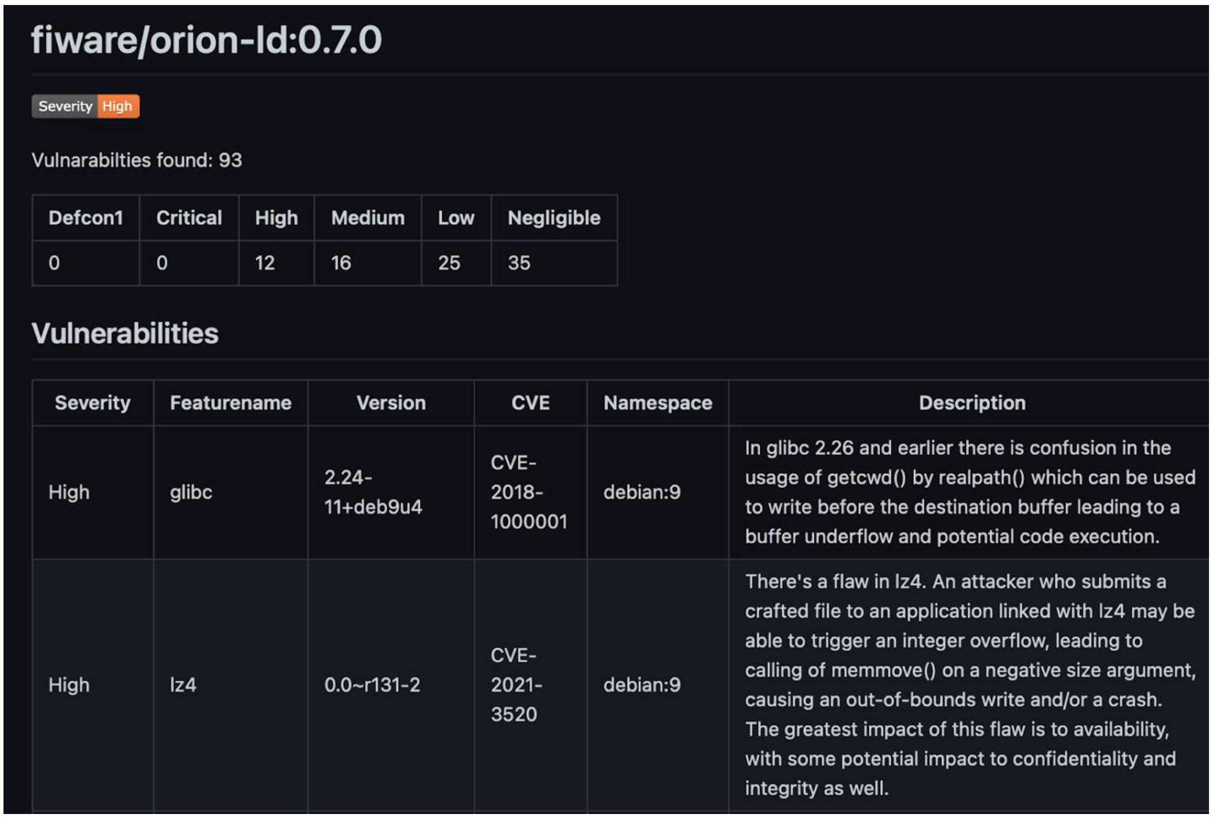
CVE-2021-35942 Critical vulnerability for multiarch-support package Error master
/var/lib/dpkg/status:1 • Detected 1 hour ago by Anchore Container Vulnerability Report (T0)

Figure 23 Example Code scanning summary generated by the Anchore GitHub Action executed

Like any GitHub Action Workflow, the creation of additional Docker images to scan can also be added to a repository and creation can be arbitrarily more complex. A second example file shows how to build an alternative base image (https://kuberty.io/blog/best-os-for-docker) using --build-arg parameters on the command line to create a container based on Red Hat UBI (Universal Base Image) 8 (https://developers.redhat.com/articles/2021/11/08/optimize-nodejs-images-ubi-8-nodejs-minimal-image). To scan this alternate image, just copy over anchore-ubi.yaml (https://github.com/flopezag/fiware-security/blob/master/sample-github-actions/anchore-ubi.yml) to .github/workflows/anchore-ubi.yml.

VII:2.5. Reporting of issues

The Clair-scanner does produce JSON-structured report files, containing the CVE information. In order to get a better human-readable overview, the results are parsed and formatted (using https://github.com/FIWARE-Ops/clair-to-markdown) in the markdown format (https://github.github.com/gfm/) to enable rendering by Github. The results are published directly on the catalog, using a dedicated branch (https://github.com/FIWARE/catalogue/tree/container-scan-results/reports).


fiware/orion-ld:0.7.0

Severity High

Vulnarabilties found: 93

Defcon1	Critical	High	Medium	Low	Negligible
0	0	12	16	25	35

Vulnerabilities

Severity	Featurename	Version	CVE	Namespace	Description
High	glibc	2.24-11+deb9u4	CVE-2018-1000001	debian:9	In glibc 2.26 and earlier there is confusion in the usage of getcwd() by realpath() which can be used to write before the destination buffer leading to a buffer underflow and potential code execution.
High	lz4	0.0~r131-2	CVE-2021-3520	debian:9	There's a flaw in lz4. An attacker who submits a crafted file to an application linked with lz4 may be able to trigger an integer overflow, leading to calling of memmove() on a negative size argument, causing an out-of-bounds write and/or a crash. The greatest impact of this flaw is to availability, with some potential impact to confidentiality and integrity as well.

Figure 24 Example of reporting security issues in FIWARE Catalogue (FIWARE Orion-LD)

Additionally, we provide an analysis of the severe issues of the different FIWARE Generic Enablers each eight weeks in the FIWARE Technical Steering Committee (TSC) based on Anchore analysis of the latest container image label. In this analysis we show the information about:

- Total number of vulnerabilities
- Total number of vulnerabilities without NVD Data
- Total number of vulnerabilities with NVD Data
- Total number of vulnerabilities with CVSS v2 Base Score > 9
- Total number of vulnerabilities with CVSS v3 Base Score > 9

The main purpose of this report is to inform FIWARE Generic Enablers owners about the number of identified vulnerabilities greater than 9 score, together with an overview of the distribution of issues, as we can see in the following example.

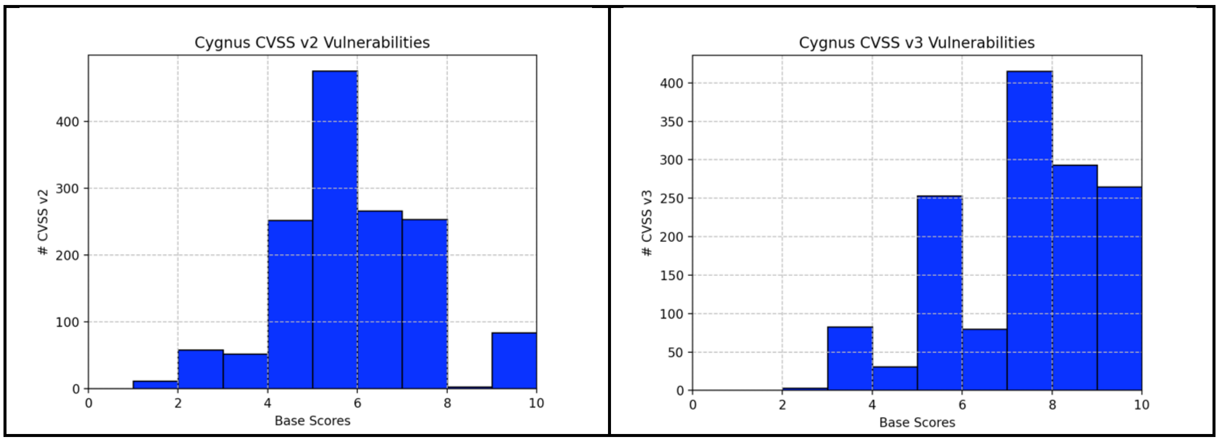


Figure 25 Example of FIWARE TSC report of FIWARE Cygnus component

VIII. Review of D1.4 recommendation for WP2

This section deals with the different actions that were developed to respond to the corresponding Gaps identified in D1.4. In the following tables, an analysis of the identified gap and the actions developed during the execution of the WP2 is presented.

Table 17. List of data models identified in D1.4

Final list of identified datamodels	Actions developed
Sluice Data model	Sluice was finally merged in the sluice gate data models while spillway was not identified initially. Spillway data model is available in the subject Open Channel Management. Developed in the Athens demo case.
Metadata attribute - Datamodels	Publication in open data portals required the population of metadata. Following the DCAT-AP 2.1.0 specification, some data models were created for the subject DCAT-AP, the most relevant Dataset data model.
Corresponding data model for water demand	Water consumption Observed data model in the subject water consumption contains some of the elements of the demand together with alarms on water use.
Sluice Gate datamodel	SluiceGate data model is available in the subject Open Channel Management. Developed in the Athens demo case.
Sensor data models	Water sensors have been identified in the project. Mainly based on the generic Device data model in the homonymous subject Device.
Data model	Other subjects were created to allocate the data from other use cases of the project, specifically • OpenChannelManagement • WasteWater • WaterConsumption • WaterQuality
EPANET Data models	The complete set of data models to allow the interoperation with EPANET modelling tool has been developed. A specific subject, WaterDistributionManagementEPANET, is available at a github repository inside the water domain.

Table 18. List of identified third parties in D1.4 to be integrated into the F4W-RA

Final list of identified Third Parties	Actions developed
EPANET	Development of the corresponding data models and gateway to automatically extract the information of a simulation and inject it into FIWARE Context Broker and vice-versa. Available at the subject Water Distribution Network Management EPANET. It includes not only the data models for the input of the network but also for the parameters of the simulation and the results outcome.

AQUADVANCED® Water Networks	Development of a connector to get time-series data from AQUADVANCED Water Networks and sends it as NGSI-LD compliant data into a FIWARE context broker.
Third-party applications	Wastewater treatment data models were developed at the Amsterdam demo case. There is a subject, WasteWater, with 6 data models in it: Blower, OffGasStack, WasteWaterJunction, WasteWaterPlant, WasteWaterSimulationResult, WasteWaterTank.
RAPID	Water consumption data from smart meters installed in Great Torrington is sent via Sigfox to Temetra; a cloud-based meter data management platform. Temetra has historically been used by SWW to capture automated meter reads (AMR). There is a pre-existing data feed from Temetra to RAPID (SWW's corporate billing system) enabling the transfer of meter reads for billing purposes. RAPID has not been configured to bill customers within the Great Torrington demo case using smart meter data however this option is available should the trial prove successful.

Table 19. Gaps from the requirements for End-Users

Requirement	Comment	Implementation Strategy	WP2 actions
Possibility of multiple implementation strategies	Currently, companies use different options for data management.	**WP2:** • FIWARE needs to provide the necessary implementation methods	F4W-RA includes several components (Cygnus, Draco, Orion-LD, Stellio) based on different technologies to facilitate the multiple data implementation strategies.
Ensure the implementation of national safety guidelines	For companies to be allowed to use F4W, the implementation of national standards concerning critical infrastructure must be ensured.	**WP2:** • Technical implementation of the requirements of national standards.	Work on the alignment between the French Sandre standard and existing NGSI-LD data models (like WaterQualityObserved).Used when ingesting data through the FIWARE Draco Generic Enabler.

Table 20. Recommendations about Open Source concerns

Requirement	Comment	Implementation Strategy	WP2 actions
Maintenance	The open source components are not properly maintained, or they are discontinued.	**WP2:** • Publish the overall daily analysed information over the FIWARE Components in order to evaluate the proper status of them.	FIWARE Foundation has translated to the FIWARE TSC the re-evaluation of the FIWARE GEs periodically in terms of support and maintenance. (WIP in the FIWARE Community).

Table 21. Recommendations about the use of Open Source platforms

Requirement	***Comment***	***Implementation Strategy***	***WP2 actions***
Integration with current open source platforms	*Companies currently using open source solutions are taking the benefit from visualisation, information searching and processing, and GIS solutions.*	***WP2:*** • Develop an architecture in which the integration with Visualisation, Information Searching and Processing are available. These functionalities are currently covered by FIWARE Components. • NGSI-LD, by default, provides geo-location information about the Entity data. The integration with GIS is a matter of translating NGSI-LD into the corresponding format using Cygnus and/or Draco FIWARE Components.	An NGSI-LD plugin has been developed for the Grafana visualisation platform. It allows displaying entities information in tables and maps. Geolocation information has been available since the first version of the ETSI NGSI-LD API (v1.1.1, Q1'2019) and fully implemented and tested in Orion-LD since Q3'2019 and Stellio since Q1'2020. PostgreSQL has been connected with the Grafana visualisation platform, showing analytics on water-related data already stored, using information aggregation features and customized plots.
Self-hosted	*The most important triggers to use open source platforms are the ability of the platform to be self-hosted*	***WP2:*** • All the FIWARE Components are deployable on-premise. • Any new component will follow the same approach.	All FIWARE GE are developed following the docker paradigm, which means that a corresponding docker image is available to be deployed on-premise. See FIWARE Requirements.
Integration with sensors and other data sources	*Existence of interfaces for integrating sensors and other data sources. There already are specialised data models per domain from different sources (Standardisation bodies, Smart Cities Initiatives, FIWARE Community). Probably such remarkable efforts shall be better advertised within the F4W community.*	***WP2:*** • Review and comment on existing data models. • Create a specific service and related page on the Project website as well as in the FIWARE portal. • Develop new interfaces (IoT Agents) to not	F4W include data models for the modelling of sensors, including the generic device data model and related to it to more specific sensors. All of them are in the Github repository. Once published a service for creating examples is automatically provided. The published data models are announced to the mailing list once published, tweets and LinkedIn posts are already automatically published. A weekly/biweekly meeting about water data models is held together with project

	Further developments in data models is advised for those areas considered not well covered by the users.	cover transport protocols and/or payload formats. • Develop new Connectors (Cygnus, Draco) to cover the integration with other Open Source solutions.	**AQUA3S** and **NAIADES**. F4W RA includes a Sigfox compatible IoT Agent that has been updated and leveraged to support the NGSI-LD specification. F4W RA includes a new component that allows to quickly deploy ML models into production and to make them natively interact with a NGSI-LD compliant context broker (in particular, to trigger new computations when receiving an NGSI-LD notification).
Dependency on third-parties	*The minimized dependency on third-parties for daily operation is expected.*	***WP2:*** • Definition of the F4W RA based on FIWARE will resolve the dependency on third-parties.	F4W RA is fully aligned with the FIWARE Community, which facilitates the integration of any third-party solution through the adoption of ETSI NGSI-LD standard and the corresponding Data Models defined in the project.

Table 22. Recommendations about sharing data

Requirement	**Comment**	**Implementation Strategy**	**WP2 actions**
Standard data models	Companies have expressed their willingness to share data with us.	**WP2:** • Definition of standardized data model/format.	Several subjects have been created to allocate all the data models required by the use cases. Extensive support for the creation and documentation. • OpenChannelManagement • WasteWater • WaterConsumption • WaterDistributionManagementEPANET • WaterQuality All the examples in the payloads are in 4 formats, json key values and normalized for NGSIv2 and the same for NGSILD. But also they are exported, when possible, into CSV, DTDL and geojson features formats.
Datasets	In order to use the corresponding datasets from interviewees, it is needed to define a mechanism to manage the data that is provided by them in the platform.	**WP2:** • Depending on the scenario, we define a simulated sensor to provide NGSI-LD data or the corresponding Cosmos connector to load the data.	An specific IoT Agent has been designed and developed from the ground up to be future-proof and customisable in order to be reused in other similar projects. When new data arrives at a given location, it processes it very fast and notifies the FIWARE Context Broker using NGSI-LD requests.

Table 23. Recommendations about data formats

Requirement	Comment	Implementation Strategy	WP2 actions
Data formats	We detect several data formats being used, most important are XML, JSON and CSV (including Excel files).	**WP2:** • Generic information exchange data format. F4W provides NGSI-LD data format, a JSON-LD representation of the Entity information. • Depending on the use case, we implement a corresponding IoT Agent to get the data or basically use a parser inside Cosmos to load the data and process them.	All the examples in the payloads are in 4 formats, JSON key values and normalized for NGSIv2 and the same for NGSILD. But also they are exported, when possible, into CSV, DTDL and geojson features formats. A @context is created containing long IRI for the terms. Additionally, specific web pages with basic information about any term of the data models are also created. Example: https://smartdatamodels.org/pressure There was no need to create new IoTAgents. We use Draco to translate the data into proper format.
Standard data models	Some of them use data models but the diversity of the information requires a large diversity of them.	**WP2:** • Through the TM-Forum and ETSI CIM as well as the ETSI SAREF we plan to integrate all of these data models.	Whenever possible ETSI SAREF4WATR and SAREF core terms were used. Mapping between these terms and the ontologies is available at another section of this document. Additionally, a service for mapping water data models with any other ontology is also available. TMforum terms do not cover water needs.
Standard data models	There is not currently a widespread standard for sensor use for (near) real-time in water monitoring.	**WP2:** • Develop the corresponding data model for water monitoring.	Extension of the generic data model for devices with new categories of properties to be measured was done. See controlledProperty attribute in the schema.

Conclusion and Perspectives

The work carried out in F4W WP2 has provided the Fiware4Water Demo Cases with the required digital platforms, by means of F4W Reference Architectures, to develop and implement the proposed smart applications for intelligent water management. Required advanced functionalities, such as real-time data analytics, systems interoperability and standardized communications, Artificial Intelligence and Machine Learning models integration, scalability, fault tolerance or high performance have been enabled thanks to the accurate architectures design and FIWARE-enabled technologies selection. Furthermore, in those cases where current technologies were limited, new pieces of FIWARE-compliant Software have been developed. The most notable example of these new developments is the implementation and FIWARE integration of BentoML, an open platform that enables MLOps functionalities and simplifies the ML model deployment and integration in data pipelines, which will be integrated into the FIWARE catalogue.

As a final point of F4W WP2 activities, the F4W RAs have been closed for all four DC, following initial designs presented in D2.1 and D2.2, by implementing and documenting the Southbound and Northbound APIs. Here, all the required developments to integrate low-level Context Information Producers and IoT devices and high-level services for data storage, manipulation and visualization have been finalized and reported. Additionally, security aspects have been analysed in order to secure the F4W deployments. On this basis, the smart applications for intelligent water management have been constructed. The flexibility of the FIWARE ecosystem has enabled the DCs to implement the corresponding smart applications with no limitations nor restrictions on the underlying technology, and integration of state of the art solutions has been possible (e.g. Matlab, Tensorflow or Apache Spark frameworks, or scikit-learn python library).

In terms of water domain standardization, several activities have been carried out, from the creation of several data models for the digital representation of water information, to the adaptation of SAREF4WATR ontology or the contributions made to the ETSI ISG-CIM standardization group. All these efforts have contributed to the open interoperability and standardized communication between the water domain agents, which includes IoT networks, legacy systems, external data sources, etc.

European added value (EAV) and upscaling

The Fiware4Water project has widely and deeply contributed to the extension and adaptation of the EU funded FIWARE platform to the digital water management domain, one of the most relevant sectors where the FIWARE technology was still not present.

In this aspect, the project outcomes have demonstrated the novel capabilities of FIWARE, nowadays required by new digital products, and how it can be extended to the water domain to deal with its peculiarities, requirements and challenges. As a result, four demonstrators (including the corresponding FIWARE Reference Architectures) have been developed in the domains of raw water supply, water distribution, wastewater treatment and water customers, which can be used as a starting point for new activities, thus acting as a successful real examples to reduce the FIWARE learning curve. Moreover, the F4W developments must be used as a base guideline of best practices for future integrations of the FIWARE ecosystem not only in the aforementioned domains, but also in any other water topic or even in any kind of FIWARE implementation.

Rellevant contributions have been made to the extension of the FIWARE ecosystem, either by means of validating the current technology, by advancing into the always critical cybersecurity aspect, or directly by developing new FIWARE components to increase its functionalities offer.

Furthermore, important inputs to the digital standardization of the water domain have been proposed and developed under the F4W project umbrella in close collaboration with other water-domain experts belonging to the ICT4Water cluster or the DigitalWater2020 synergy group, in which five H2020 projects are involved. In this aspect, the so-called **Common Information Model for Digital Water Management** proposes several new data models to digitally represent water information and assets in a standardized way, and important efforts have been devoted to linking them to the well established and accepted SAREF and SAREF4WATR ontologies, thus enabling open interoperability among the water domain agents. Also, as a result of F4W developments, important contributions have been made to the ETSI ISG-CIM standardization group.

All these efforts would not have been possible without the close collaboration of F4W European partners and external experts (e.g. ICT4Water cluster or DW2020 projects) who contributed with their expertise in very sparse areas (e.g. water domain, ICT technology or standardization mechanisms) to the Fiware4Water project outcomes. Additionally, the complex and variate challenges proposed by F4W DCs have also contributed to the wide F4W offer of solutions and implementations.

References

[1] Wegner, Peter. "Interoperability." *ACM Computing Surveys (CSUR)* 28.1 (1996): 285-287.

[2] Rossman, L.A., 2000. EPANET2 user’s manual. U.S. Environmental Protection Agency, Cincinnati.

Annex I. Greek Demo Case: the final Fiware4Water Reference Architecture

The following diagram depicts the final version of the F4W RA of the Greek Demo case. The main modifications regarding the version presented in deliverable D2.2 reside in (a) the connection of the Nessie Engine with the FIWARE Draco system's database for reading historical data and (b) the development of an API for the potential 3rd party applications to connect to Nessie engine and read historical, real-time and other analytical data. Nessie engine comes with its own database connection (PostgreSQL). But since it is the same database with the one FIWARE Draco is using, we decided that using two databases was unnecessary and decoupled our database in favour of the database used by FIWARE Draco.

Furthermore, the Restful API we developed is secure and an API key must be given to anyone who wants to access historical or real-time data. In practice, the EYDAP front-end UI developed by NTUA connects with the Nessie Engine using mostly this API in order to read real-time and historical data.

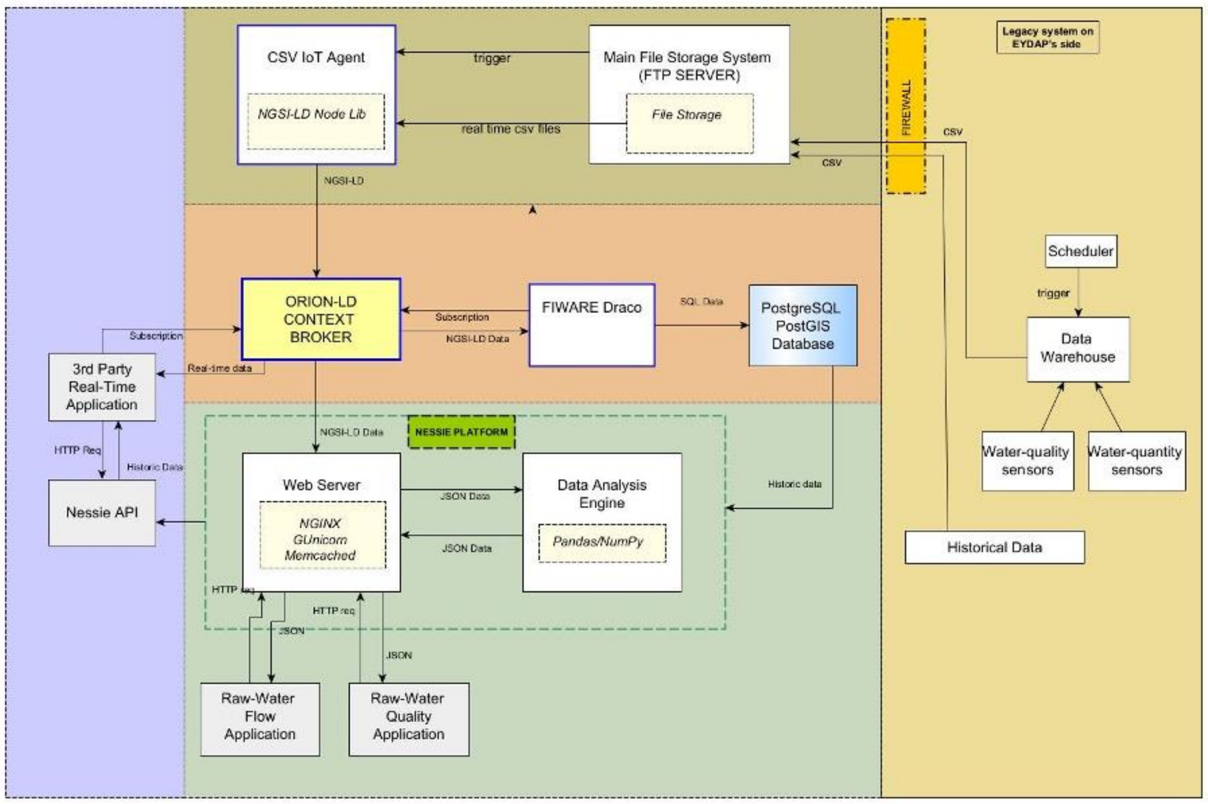


Figure 26. Greek Demo Case: the final Fiware4Water Reference Architecture

Annex II. Fench Demo Case: the final Fiware4Water Reference Architecture

The architecture deployed for the French demo case illustrates the interoperability of systems, made possible thanks to the use of the standardized NGSI-LD API and the alignment of the FIWARE ecosystem on this specification. Thus, it is based on a functional architecture common to both Fiware4Water and aqua3S projects. This specific point has already been highlighted in the deliverables of these two projects and in particular in deliverable D4.2 of the F4W project.

On the other hand, it responds to the need for synergy desired by the EC which created the DW2020 synergy group in which five H2020 projects are involved.

The general principles and flows of data are then the same for both projects (even if the data and processes applied to data are of course not the same for both):

- Business data is retrieved from the 3S legacy systems
- It is ingested in the target context broker (Stellio for the Fiware4Water project, Orion-LD for the Aqua3S project)
- Scientific models developed by 3S use the data ingested in the context broker to run some processing and obtain predictions
- The predictions are sent to the context broker and are later retrieved by the 3S legacy systems for use by operators of the system

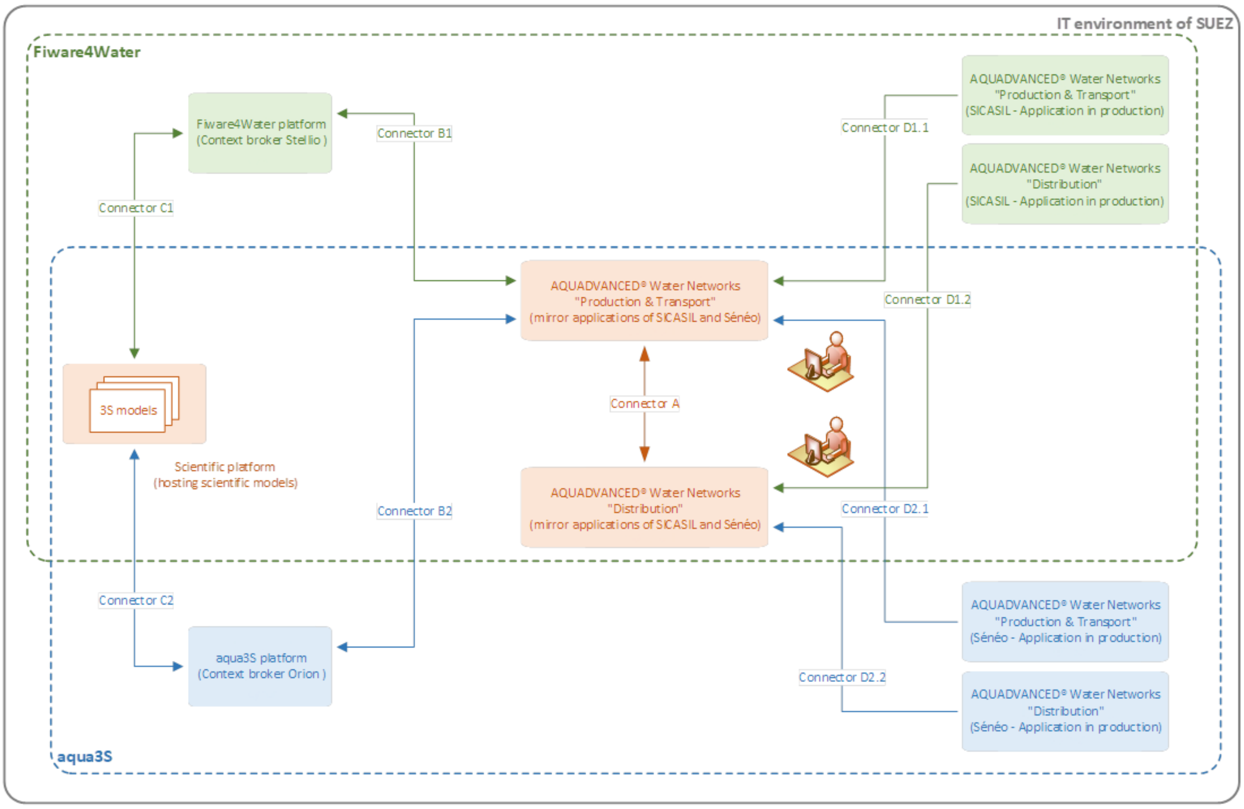

IT environment of SUEZ
Fiware4Water
Fiware4Water platform (Context broker Stellio)
Connector B1
Connector D1.1
AQUADVANCED® Water Networks "Production & Transport" (SICASIL - Application in production)
AQUADVANCED® Water Networks "Distribution" (SICASIL - Application in production)
Connector C1
AQUADVANCED® Water Networks "Production & Transport" (mirror applications of SICASIL and Sénéo)
Connector D1.2
3S models
Connector A
Scientific platform (hosting scientific models)
Connector B2
AQUADVANCED® Water Networks "Distribution" (mirror applications of SICASIL and Sénéo)
Connector D2.1
Connector C2
AQUADVANCED® Water Networks "Production & Transport" (Sénéo - Application in production)
aqua3S platform (Context broker Orion)
Connector D2.2
AQUADVANCED® Water Networks "Distribution" (Sénéo - Application in production)
aqua3S

Figure 27. Fench Demo Case: the final Fiware4Water Reference Architecture

A BigData extension has been presented through the integration of the Apache Spark framework, a large-scale data analytics engine which provides scalability, high performance and fault tolerance capabilities.

Annex III. Amsterdam Demo Case: the final Fiware4Water Reference Architecture

Within the Amsterdam demo case, the F4W architecture was developed for the Amsterdam West WWTP. The F4W setup consists of a Docker containers microservices architecture on a Windows VM in the Azure cloud, as utilised in Waternet's legacy system. Given below in Figure 28, the various FIWARE components, connectors/integrators developed and the various smart applications developed have been visualised. Additionally, in Table 24. Description on the type of architectural components, as part of the F4W setup illustration in Figure 28, an overview has been provided that describes what type the individual components are. Further details on the F4W architecture specific to this demo case can be found in Deliverable 4.4.

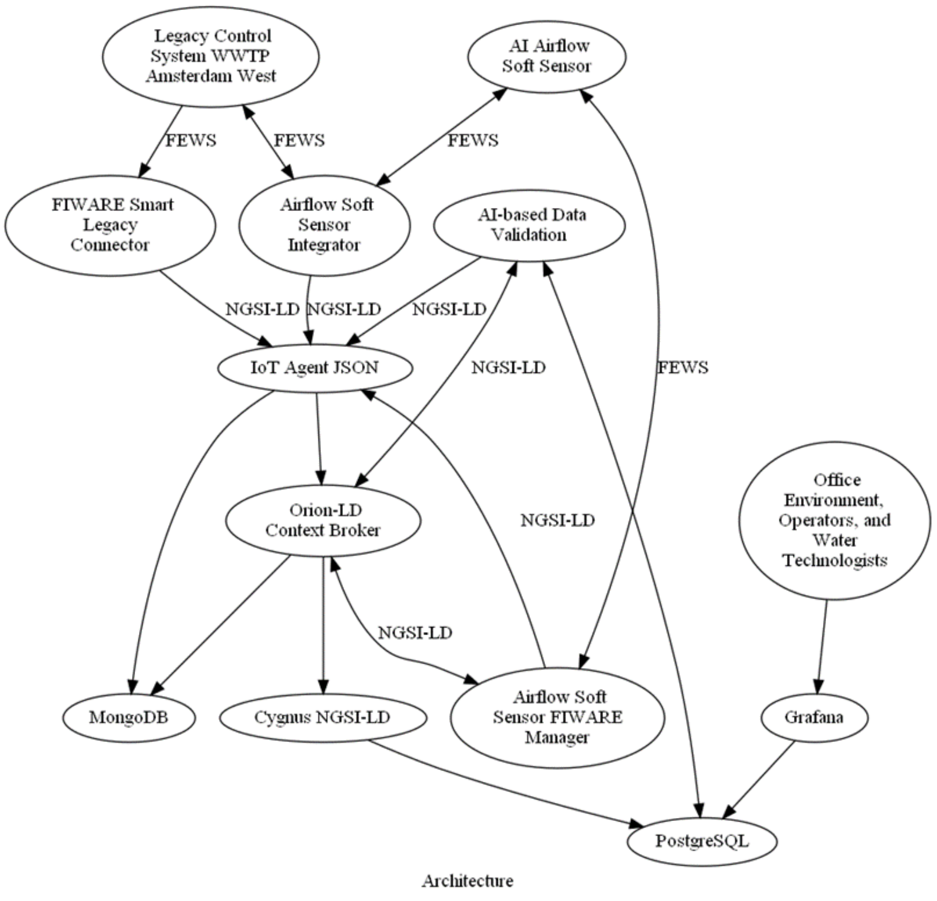


Figure 28. Amsterdam Demo Case: the final Fiware4Water Reference Architecture

Table 24. Description on the type of architectural components, as part of the F4W setup illustration in Figure 28

Architectural Component	Type
FIWARE Smart Legacy Connector	Gateway/Connector
Airflow Soft Sensor Integrator	Gateway/Connector
IoT Agent JSON	FIWARE
AI Airflow Soft Sensor	Smart Application
Orion-LD Context Broker	FIWARE
AI-based Data Validation	Smart Application, Gateway/Connector

MongoDB	Database
Cygnus NGSI-LD	FIWARE
Airflow Soft Sensor FIWARE Manager	Gateway/Connector
PostgreSQL	Database
Grafana	Dashboard

Comparing the final F4W architecture as setup for the demo case (Figure 28), with the intermediate architecture proposed in Deliverable 2.2, the primary modifications were the use of a different database (PostgreSQL) instead of the proposed CrateDB to store the inputs and output of the smart applications. Additionally, Cygnus NGSI-LD was utilised to be able to dump data from the Orion-LD context broker into PostgreSQL. Initially, Apache Spark was considered for the development of the smart applications. However, during the course of the development of the applications, it was decided upon to use Python-based packages and software (such as Tensorflow) for the development of the deep learning models. As a result, the initially proposed use of Cosmos was not necessary and was not included in the final architecture.

Annex IV. UK Demo Case: the final Fiware4Water Reference Architecture

The final version of the Fiware4Water architecture deployed in the scope of the UK DC is aligned with the intermediate version presented in the deliverable D2.2.

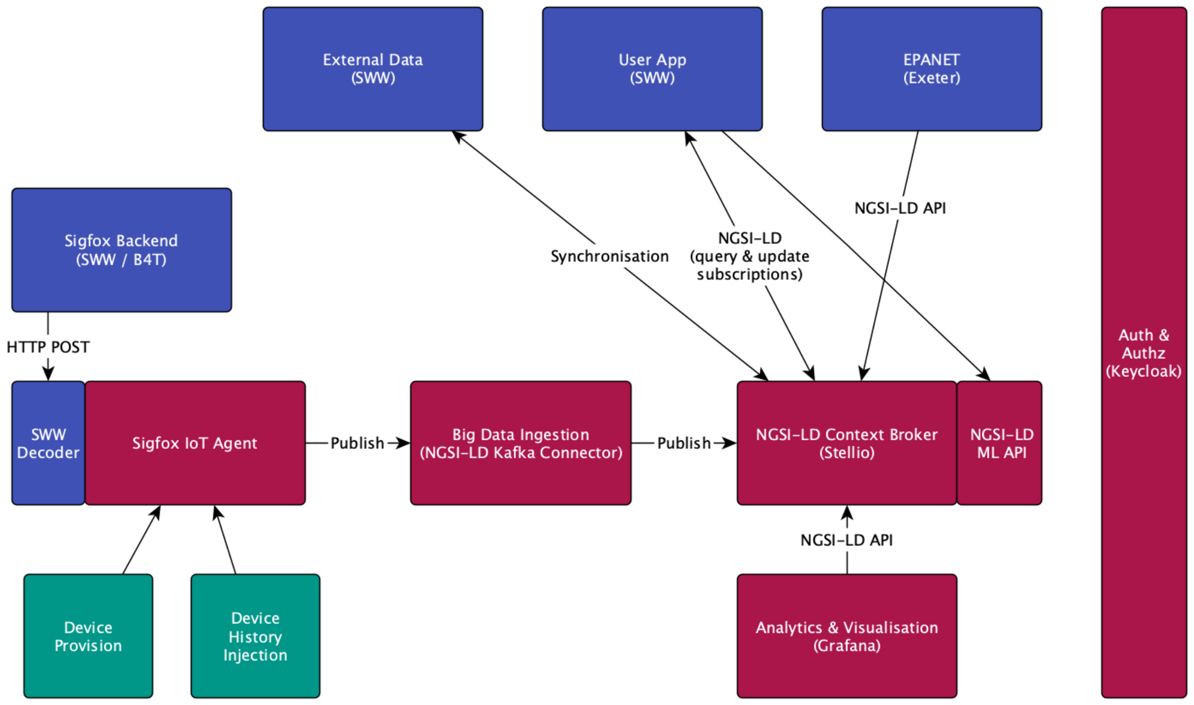


Figure 29. UK Demo Case: the final Fiware4Water Reference Architecture