



AI, Big Data, IoT Orchestration Workbench

D6.4

Authors:

Ferdinando Bosco (ENG)

Gaetano Sciabica (ENG)

Angelo Triveri (ENG)

Vassilis Sakas (ED)

Konstantinos Kotsalos (ED)

Apostolos Kapetanios (ED)

Eleni Panagou (UBE)

Responsible Partner	ENG
Verified by the appointed Reviewers	Dusan Jakovljevic, Juergen Ritzek, Rod Janssen (EEIP) Bartosz Kalinowski (TTSA), 26.07.2023
Approved by Project Coordinator	Padraic McKeever (Fraunhofer), 31.07.2023

Dissemination Level	Public
----------------------------	--------



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957739

Issue Record

Planned delivery date	31.07.2023	
Actual date of delivery	31.07.2023	
Status and version	V1.0	

Version	Date	Author(s)	Notes
0.1	26/05/2023	Ferdinando Bosco (ENG)	Table of Contents
0.2	12/07/2023	Ferdinando Bosco (ENG) Eleni Panagou (UBE)	Chapter 7 - One Net Monitoring and Analytics Dashboard
0.3	18/07/2023	Vassilis Sakas (ED) Konstantinos Kotsalos (ED) Apostolos Kapetanios (ED) Ferdinando Bosco (ENG) Gaetano Sciabica (ENG) Angelo Triveri (ENG)	Ch 2 contributions. Ch. 4 and 5 contributions. Ch. 1, 3, 6 contributions. Consolidation of the document. Ready for internal review.
0.4	27/07/2023	Ferdinando Bosco (ENG) Dusan Jakovljevic, Juergen Ritzek, Rod Janssen (EEIP) Bartosz Kalinowski (TTSA)	Document reviewed by internal reviewers. Feedback and comments addressed. Document ready for the final quality check.

Disclaimer:

All information provided reflects the status of the OneNet project at the time of writing and may be subject to change. All information reflects only the author's view and the European Climate, Infrastructure and Environment Executive Agency (CINEA) is not responsible for any use that may be made of the information contained in this deliverable.

About OneNet

The project OneNet (One Network for Europe) will provide a seamless integration of all the actors in the electricity network across Europe to create the conditions for a synergistic operation that optimizes the overall energy system while creating an open and fair market structure.

OneNet is funded through the EU's eighth Framework Programme Horizon 2020, "TSO – DSO Consumer: Large-scale demonstrations of innovative grid services through demand response, storage and small-scale (RES) generation" and responds to the call "Building a low-carbon, climate resilient future (LC)".

As the electrical grid moves from being a fully centralized to a highly decentralized system, grid operators have to adapt to this changing environment and adjust their current business model to accommodate faster reactions and adaptive flexibility. This is an unprecedented challenge requiring an unprecedented solution. The project brings together a consortium of over seventy partners, including key IT players, leading research institutions and the two most relevant associations for grid operators.

The key elements of the project are:

1. Definition of a common market design for Europe: this means standardized products and key parameters for grid services which aim at the coordination of all actors, from grid operators to customers;
2. Definition of a Common IT Architecture and Common IT Interfaces: this means not trying to create a single IT platform for all the products but enabling an open architecture of interactions among several platforms so that anybody can join any market across Europe; and
3. Large-scale demonstrators to implement and showcase the scalable solutions developed throughout the project. These demonstrators are organized in four clusters coming to include countries in every region of Europe and testing innovative use cases never validated before.

Table of Contents

1 Introduction	8
1.1 Scope	8
1.2 Task 6.4	9
1.3 Outline of the deliverable	10
2 Data-driven Services Orchestration	11
3 OneNet Architecture, Use Cases and Requirements	16
3.1 OneNet Architecture	16
3.2 OneNet Use Cases and Requirements	18
3.2.1 Requirements refinement	20
4 OneNet Cross-Platform services	25
5 OneNet Workbench: Integrating Third Party Services	27
5.1 Presify - Data Fetcher and Anomaly Detection Models	27
5.1.1 Data Fetcher Service	27
5.1.2 Anomaly Detection Service	29
5.2 BeeData – Outliers Detection and Imputation	32
5.2.1 Outlier Detection	32
5.2.2 Imputation method	32
6 OneNet Orchestration Workbench	34
6.1 Architecture	34
6.2 Functionalities	36
6.3 Interfaces and communication mechanisms	37
6.3.1 Graphical User Interface (GUI)	39
6.4 Languages and Software	45
6.5 Packaging and deployment	45
7 OneNet Monitoring and Analytics Dashboard	46
7.1 Architecture	46
7.2 Services	47
7.3 Interfaces and communication mechanisms	50
7.4 Languages and Software	50
7.5 Packaging and deployment	51
8 Conclusions	53
References	54

Figure 1: WP6 interactions	8
Figure 2: 5V's of Big Data Model	13
Figure 3: Brief overview of the Big Data workflow	14
Figure 4: OneNet Reference Architecture [3]	16
Figure 5: OneNet Interfaces	20
Figure 6: Cross-platform service categories enable the OneNet System Services identified in WP2	25
Figure 7: OneNet Middleware, Cross-Platform Services orchestration.....	26
Figure 8: Data Fetcher Service Example Request	29
Figure 9: Anomaly detection service example request	31
Figure 10: OneNet Orchestration Workbench Architecture.....	35
Figure 11: SOGNO architecture [8]	36
Figure 12: OneNet Orchestration Workbench Workflow Example	37
Figure 13: OneNet Orchestration Workbench Login section.....	39
Figure 14: OneNet Orchestration Workbench Data Offerings section	40
Figure 15: OneNet Orchestration Workbench Subscriptions and Data Consumption section.....	40
Figure 16: Microservice Orchestration Layer - Rancher Login.....	41
Figure 17: Microservice Orchestration Layer - Deployed services	41
Figure 18: Microservice Orchestration Layer - Services monitoring.....	42
Figure 19: OneNet Orchestration Workbench - Service Catalogue	42
Figure 20: OneNet Orchestration Workbench - Service execution and data integration	43
Figure 21: OneNet Orchestration Workbench - Overall system evaluation	44
Figure 22: OneNet Orchestration Workbench - Service monitoring	44
Figure 23: OneNet Monitoring and Analytics Dashboard Service Architecture	46
Figure 24: Dashboard Login and Registration Sections	48
Figure 25: Dashboard map chart displaying number of requests per country for the last 30 days	49
Figure 26: Dashboard bar chart displaying number of requests per day for the last 14 days.....	49
Figure 27: Dashboard Account Settings Component.....	49
Figure 28: Account Management Page: Device Activity and Logged in Applications.....	50
Figure 29: OneNet Monitoring and Analytics Dashboard Docker-based Deployment.....	52



Table 1: Functional Requirements relevant for OneNet Orchestration Workbench and Dashboard Implementation	18
Table 2: Refined Functional Requirements for OneNet Orchestration Workbench and Monitoring Dashboard	20
Table 3: OneNet Orchestration Workbench external interfaces.....	38
Table 4: OneNet Orchestration Workbench - Languages and software	45



List of Abbreviations and Acronyms

Acronym	Meaning
AI	Artificial Intelligence
API	Application Programming Interface
ETL	Extract Transform Load
GUI	Graphical User Interface
IoT	Internet of Things
IT	Information Technology
JSON	JavaScript Object Notation
LSTM	Long short-term memory
MQTT	Message Queuing Telemetry Transport
REST	REpresentational State Transfer
SCADA	Supervisory Control And Data Acquisition
SUCs	System Use Cases
UI	User Interface
WP	Work Package

Executive Summary

The OneNet Solution relies on the concept of a pan-European ecosystem, which, through interfaces and standardized data models adoption, enables a seamless integration and cooperation among energy stakeholders for cross-platform market and network operation services.

In line with the OneNet Reference Architecture, the OneNet Solution should utilize a fully decentralized approach to enable a secure exchange of data. Simultaneously, it should provide extra components and tools to make it easier for energy stakeholders to cooperate and collaborate with each other.

The OneNet Solution consists of three main components: the OneNet Decentralized Middleware (which includes the OneNet Connector) that is at the base of the data exchange and the OneNet Data Space ecosystem and two additional components that enrich the OneNet offering for the integration: monitoring and evaluation of data and services.

This document is focused on the detailed description of these two components: the OneNet Orchestration Workbench, and the OneNet Monitoring and Analytics Dashboard

These components were designed, implemented and released following the requirements and specifications collected in one of the project work packages (WP5) and integrated in the overall OneNet system in order to be compliance with the OneNet Decentralized Middleware and Connector.

The Orchestration Workbench allows to interconnect the OneNet Network of Platforms (using the OneNet Connector) with any kind of data-driven services. These services can be deployed, tested and evaluated in a dedicated environment offered by the Workbench. The Orchestration Workbench offers some basic service to be tested (e.g. data visualization) and can be extended with cross-platform services implemented within the demo platforms or external services such as those made available as a result of the OneNet open calls. The latter approach could, for example, give the possibility to test an external service, in a controlled environment and with full control over the use of data, given by the integration with the OneNet Connector.

Furthermore, the Monitoring and Analytics Dashboard offers a GUI for facilitating the OneNet Participants in the management, monitoring and analytics of the data exchanges. This tool offers in fact, many services, both backend and front end for logging the data exchange, monitoring the usage of the data and alerting in case of any possible security issue.

The Orchestration Workbench and the Monitoring and Analytics dashboard are closely related to each other and allow additional functionalities to be added to the overall OneNet System, not limiting to the simple integration of the OneNet Connector into existing systems and enabling data exchange. This allowed us to have a more extensive and different solution than any other implementation of data-space based systems.

1 Introduction

1.1 Scope

One of the primary objectives of OneNet is to introduce an open and adaptable architecture that will revolutionize the current European electricity system. Currently, the system is often managed on a fragmented country- or area-level basis. OneNet aims to transform it into a pan-European system that is smarter, more efficient, and highly interconnected. This transformation will empower consumers to actively participate in an open market structure.

WP6 “Reference IT Implementation for OneNet” contributes to fulfilling the OneNet vision by striving to attain four objectives:

- Develop the OneNet Decentralized Middleware (and OneNet Connector) that will enable the interconnection of energy actors and the seamless data management and exchange.
- Incorporate the data interoperability and cyber security aspects into the OneNet System.
- Incorporate the Big Data and the FIWARE layers into the OneNet System.
- Develop and test the OneNet Interoperable Network of Platforms.

WPs Interactions

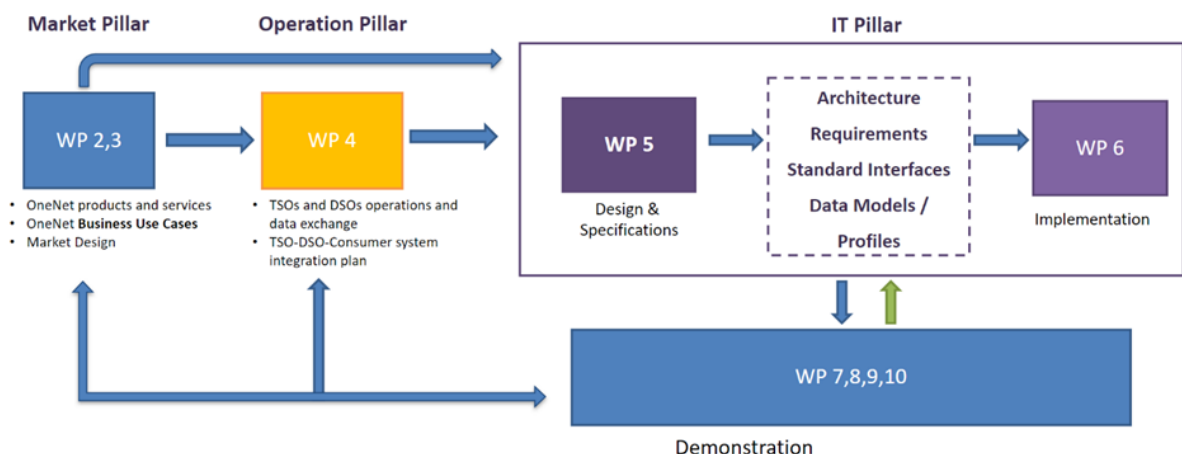


Figure 1: WP6 interactions

The WP6, following the design phase of the WP5 “Open IT Architecture for OneNet”, acts as the IT main pillar of the overall OneNet project, covering the implementation phase.

The IT pillar is closely linked to the other pillars of the project, as shown in Figure 1. It takes into consideration all the results provided in the Market Pillar (WP2 and WP3) as well as the Operation Pillar (WP4). In addition, the OneNet Solution, implemented in WP6 will be tested and evaluated in four Demonstration Clusters and the results of the evaluation will be used for adapting, improving, and enhancing the OneNet Solution.

1.2 Task 6.4

The main goal of the Task T6.4 “AI, Big Data, IoT Workbench to support scalable near real time big data-enabled services” is to design, implement and make available the OneNet Orchestration Workbench and Monitoring and Analytics Dashboard, allowing the data and service orchestration of AI, Big Data, IoT data-driven services, as well as the data visualization and monitoring.

Task 6.4, started in the seventh month of the OneNet project. It collects all the necessary information provided from the WP5 tasks and in particular: the requirements and use cases (from Task T5.1), the OneNet Reference Architecture (Task T5.2) the OneNet Orchestration Workbench and Monitoring Dashboard specifications and interfaces (Task T5.4).

All these inputs were used as reference for the implementation of the OneNet Orchestration Workbench and the OneNet Monitoring and Analytics Dashboard.

In addition, Task T6.4 strictly collaborated with Task T6.1 for the refinement of the requirements and with Task T6.5 for the integration of the two components in the overall OneNet system.

Finally, the following results were produced by Task T6.4:

- in January 2023, the initial release of the OneNet Orchestration Workbench and OneNet Monitoring and Analytics Dashboard was reported in Milestone 16.
- as reported in this document, the OneNet Orchestration Workbench and OneNet Monitoring and Analytics Dashboard reached their ultimate version and were officially released in July 2023.

It is important to emphasize, that relevant insights have been made downstream of the expected requirements for the OneNet Orchestration Workbench and the Monitoring and Analytics Dashboard, with respect to their use in relation to services based on AI, Big Data and IoT. In fact, in this analysis, starting from the assumption that specific AI, Big Data and IoT services were not available within the OneNet system or of significant interest for the Orchestration and Monitoring tools, led to the decision to implement more generic components, oriented towards integration with data coming from the OneNet Connector (in a data-based ecosystem) and providing a real orchestration, deployment and monitoring of data-driven services. For this

reason, the two components Orchestration Workbench and Monitoring and Analytics Dashboard replaced the expected (in the OneNet Description of Action) AI, Big Data, IoT Orchestration Workbench.

1.3 Outline of the deliverable

This deliverable is structured in 8 different chapters.

The Introduction, Chapter 1, puts the deliverable into context within the WP6 and task T6.4.

Chapter 2 analyses the concept of data and service orchestration, with a particular focus on AI and 'Big Data' data-driven service.

Chapter 3 report the results of WP5 in terms of architecture, requirements and use cases, used as a starting point for the design, implementation and the integration of the OneNet Orchestration Workbench and OneNet Monitoring and Analytics Dashboard.

Chapter 4 describes the concept of cross-platform services (defined in WP5) and how these services can be orchestrated, tested and evaluated in the Orchestration Workbench.

Chapter 5 reports a parallel activity conducted in collaboration with Task T12.2 "Management of external call and cascading fundings" for the analysis and integration in the Orchestration Workbench of two services came out during the open call process.

Chapter 6 describes in detail the OneNet Orchestration Workbench in terms of architecture, functionalities, interfaces, and deployment.

Chapter 7 describes in detail the OneNet Monitoring and Analytics Dashboard in terms of architecture, functionalities, interfaces, and deployment.

Finally, Chapter 8 provides the final conclusions.

To facilitate the readability of this report, it is useful to refer to previous deliverables: D5.2 [3] and D5.4 [2] for information about the design phase; D6.5 [5] for further references on the implementation and integration phase; and D6.6 [6] for additional information about the monitoring and analytics services related to cybersecurity aspects.

2 Data-driven Services Orchestration

Orchestration services are a set of technologies and tools used to coordinate and manage the execution of complex processes that involve multiple systems and applications. The main goal of service orchestration is to automate the management of activities, resource allocation, and communication between various components of a distributed system.

In practice, orchestration services enable the definition and management of the workflow between different applications and services, integrating the various elements to create a continuous and coherent process. This means that orchestration services can be used to automate a wide range of activities, such as data processing, network service management, workload distribution, computing resource management, and much more.

Orchestration for services and data offers a range of benefits, particularly in complex environments such as microservices architectures or contexts where many interconnected systems and services are present. Here are some of the main reasons why orchestration can be advantageous:

- **Simplified management:** orchestration allows managing a set of services as a single system, simplifying the configuration, deployment, and management of the services.
- **Increased efficiency:** orchestration enables optimizing resource utilization, thus reducing the total cost of ownership. For example, an orchestrator can automatically decide where and when to run a service based on resource availability.
- **Improved resilience:** a good orchestration system can detect failures and automatically restore services, thereby increasing the resilience of the system.
- **Scalability:** with orchestration, services can be easily scaled up and down based on demand, allowing for a flexible response to changes in workload.
- **Versioning and updates:** orchestration facilitates the rollout of new versions of services and the updating of existing services, making it easier to keep services up to date with the latest security and functionality.
- **Automated deployment process:** orchestration enables automating deployment processes, reducing human errors and increasing the speed of distributing new services.
- **Monitoring:** most orchestration systems provide monitoring tools that allow visualizing the status of the system and identifying issues proactively.

Data orchestration offers similar benefits, allowing for more efficient and effective management of data between different services and systems. Data orchestration is the organization and management of data between different systems and services distributed over a network. The objective of data orchestration is to

ensure that data is readily available and easily accessible to the systems and services that require it. Additionally, it aims to maintain the consistency and integrity of the data.

In a microservices architecture, for example, data orchestration can be used to ensure that data is available to all services that need it. Additionally, data orchestration can be used for efficient distribution of data, while optimizing the use of resources.

Moreover, data orchestration can be used to ensure data security by enabling the definition of data access policies and monitoring data access. Thus, data orchestration can help protect sensitive data and ensure compliance with data privacy regulations.

In summary, data orchestration is a fundamental tool for managing data distributed over a network, simplifying data management and improving efficiency of business processes.

Big Data

The term "Big Data" has gained significant traction in recent years and is increasingly being used within the scientific community for a wide range of applications. However, there is no universally accepted definition of what exactly constitutes "Big Data". It refers to datasets with a high volume, typically measured in exabytes (10^{18} Bytes) or larger. As analog technologies are gradually being replaced by their digital equivalents, the data logs associated with these new solutions often generate substantial amounts of data on a daily basis, particularly within large infrastructures and corporations.

Given that data mining, machine learning, and data sciences in general are currently among the most researched topics in computer science, the demand for data is continuously growing. However, when dealing with massive amounts of data, modern-day IT infrastructure and establishments often find that their processing power and storage capabilities are insufficient for the task at hand. As a result, innovative solutions are emerging in the domain of Big Data storage, management, processing, analytics, and visualization [1].

Currently, there is no standardized and uniform definition of Big Data. However, various descriptions agree that it refers to an emerging technical challenge presented by datasets with massive volumes, diverse categories, and complex structures. Dealing with such data requires the use of innovative frameworks and techniques to effectively extract useful information.

The characteristics of Big Data are universally identified using the "5 V" Big data model:

1. **Volume:** This refers to the size of the Big Data, and whether data can be considered Big Data or not is based on its volume. The rapidly increasing volume of data is due to cloud computing traffic, IoT, mobile traffic, etc.
2. **Variety:** This refers to the structured, semi-structured, and unstructured data resulting from different sources of data generated either by humans or machines. Structured data is organized and

conforms to the formal structure of data and can be stored in a relational database. Semi-structured data is semi-organized and doesn't conform to the formal structure of data, while unstructured data is not organized and doesn't fit into the rows and columns structure of a relational database. Examples include text files, emails, images, videos, voicemails, audio files, etc.

3. **Velocity:** This refers to the speed at which the data is accumulating, mainly due to IoT, mobile data, social media, etc.
4. **Veracity:** This refers to the assurance of quality, integrity, credibility, and accuracy of the data. Since data is collected from multiple sources, the data needs to be checked for accuracy before using it for business insights.
5. **Value:** This refers to how useful the data is in decision-making. We need to extract the value of the Big Data using proper analytics.

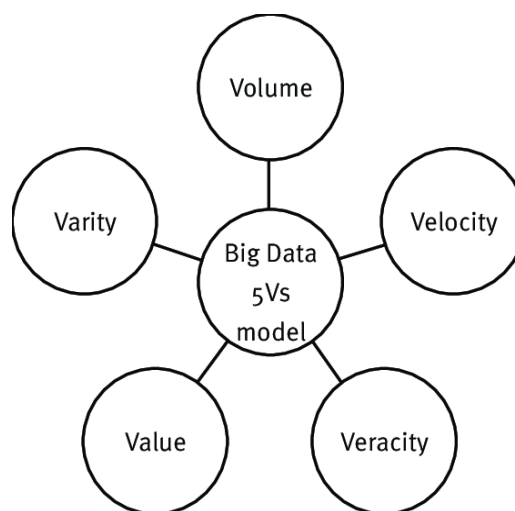


Figure 2: 5V's of Big Data Model

As modern technologies increasingly rely on data, working with Big Data has become a critical feature of these solutions. However, the acquisition, storing, processing, and information derivation of Big Data present various challenges. This section outlines the key challenges that relate to workflows with high amounts of data, with a focus on the previously explained V's and in accordance with [2]:

1. **Heterogeneity:** Differences in data sources produce significant inconsistencies with the acquired data structure. Workflows with both semi-structured and unstructured data must include a pre-processing module.
2. **Uncertainty of data:** Reliability of collected data inevitably fluctuates with missed, partial, and faulty measurements. Correct data management actions, such as data cleaning, filtering, and transforming, are mandatory at the beginning of the process to avoid incorrect outputs of analytics engines.

3. **Scalability:** As the volume of data generated dramatically increases over time, efficient Big Data processing algorithms must be able to work with chunks of data that are getting larger every day. Only incremental algorithms are resistant to this kind of data expansion. Storage management is another challenge as it requires a colossal memory capacity, rendering traditional solutions obsolete. Contemporary cloud-based solutions also face feasibility issues, especially in real-time applications where upload speeds are a bottleneck.
4. **Timeliness:** Real-time applications require quick responses immediately following data acquisition. Delayed responses render the supplied information useless. For example, in a money fraud detection scenario, information about an ongoing scam is only considered important at that specific instance in time to potentially stop the transaction from proceeding.
5. **Fault tolerance:** The correctness of Big Data processing is a key aspect, but high volumes, unstructured form, distributed nature of data in NoSQL data management systems, and the necessity of near-to-real-time responses often lead to corrupt results with no method able to guarantee complete validity of the given results.
6. **Data security:** Personal or sensitive enterprise data requires special attention to data protection. Loss or theft of personal information, such as names, addresses, location history, social security information, or credit card PIN codes, must be prevented at the highest possible standard.
7. **Visualization:** Adequate representation of results is crucial in Big Data processing. Various ways in which the data can be displayed affect the usefulness of derived knowledge.

	Storing	Processing	Analytics	Visualization
Heterogeneity	+	+		
Uncertainty of data		+	+	
Scalability	+	+	+	
Timeliness	+	+	+	
Fault tolerance		+	+	
Data security	+	+		
Visualization				+

Figure 3: Brief overview of the Big Data workflow

To overcome these challenges, a diverse range of solutions has been developed, which will be presented and discussed in the remainder of this report. The Big Data workflow can be disaggregated into four main steps: storing, processing, analytics, and visualization. Each solution presented is tasked with a specific part of the workflow. Storing and processing are key when addressing challenges related to heterogeneity and data security. Scalability and timeliness also influence the analytics stage. The proper choice of processing and analytics tools is necessary to manage data uncertainty and fault tolerance [2].

The most common data orchestration architectures are:

1. Apache Hadoop¹: an open-source framework for distributed processing of large amounts of data on computer clusters.
2. Apache Spark²: an open-source framework for distributed in-memory data processing on computer clusters.
3. Apache Kafka³: a distributed and scalable data streaming platform.
4. Apache NiFi⁴: an open-source data flow management system for automating the flow of data between systems.
5. Apache Airflow⁵: an open-source data workflow platform for creating, scheduling, and monitoring complex data workflows.
6. Kubernetes⁶: an open-source container orchestration platform for managing containerized applications.
7. AWS Glue⁷: a fully managed ETL service that simplifies and automates data management.
8. Google Cloud Dataflow⁸: a fully managed data processing service for real-time and batch processing of large amounts of data.
9. Microsoft Azure Data Factory⁹: a fully managed ETL service for processing large amounts of data.
10. Talend¹⁰: an open-source data integration architecture for processing and integrating data on on-premises and cloud platforms.

¹ <https://hadoop.apache.org/>

² <https://spark.apache.org>

³ <https://kafka.apache.org/30/documentation.html>

⁴ <https://nifi.apache.org/>

⁵ <https://airflow.apache.org/>

⁶ <https://kubernetes.io/>

⁷ https://docs.aws.amazon.com/it_it/glue/latest/dg/what-is-glue.html

⁸ <https://cloud.google.com/dataflow?hl=it>

⁹ <https://azure.microsoft.com/it-it/products/data-factory>

¹⁰ <https://www.talend.com/it/>

3 OneNet Architecture, Use Cases and Requirements

3.1 OneNet Architecture

One of the main outputs of the D5.2 [3] is a very detailed analysis of the most relevant initiatives, reference architectures and projects concerning to the OneNet main objectives and expected characteristics. The results of this analysis, together with the analysis of the OneNet main concept, Use Cases, and requirements, brought to the design and definition of OneNet Reference Architecture.

The OneNet Reference Architecture (shown in the Figure 4) includes the two components relevant for this deliverable:

- OneNet Orchestration Workbench.
- OneNet Monitoring and Analytics Dashboard.

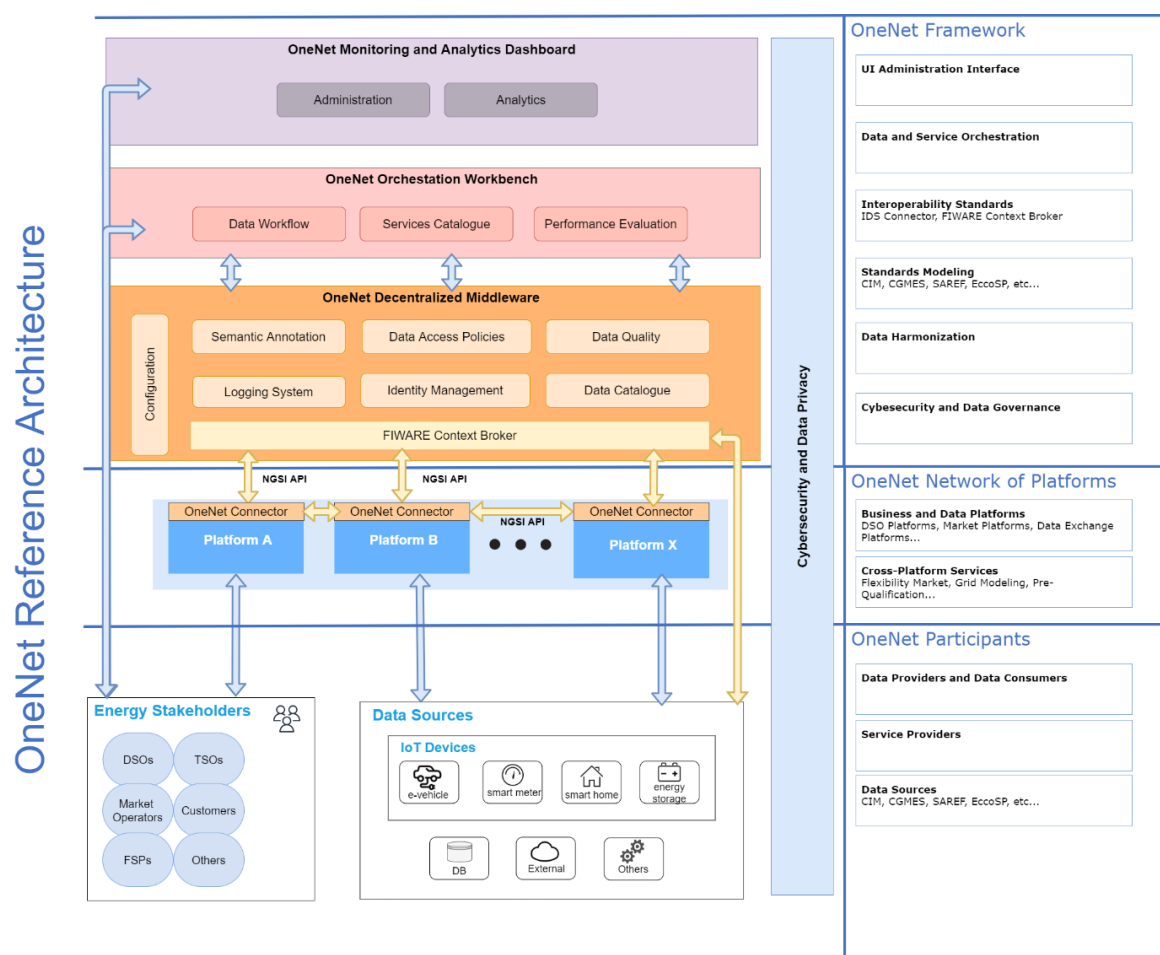


Figure 4: OneNet Reference Architecture [3]

OneNet Orchestration Workbench

The OneNet Orchestration Workbench is described in the D5.2 [3]. It is a component to orchestrate and evaluate the performance and scalability of the cross-platform services integrated and implemented in the OneNet System.

Any OneNet Participant and in particular the Service Providers can test and evaluate its own service exploiting the OneNet Orchestration Workbench, that allows to integrate data coming from the OneNet Middleware and to implement a data pipeline orchestration.

From a functional perspective, following the functional requirements of D5.1 [4], the OneNet Orchestration Workbench supports the integration and the evaluation of the performance and scalability of the AI, IoT and Big Data cross-platform services for market and grid operations.

The OneNet Orchestration Workbench allows to integrate data coming from the OneNet middleware and allow the interaction with the service providers which can register and evaluate their own services, implementing services orchestration, data pipeline and evaluation schema.

Some expected features collected during the requirements phase for the OneNet Orchestration Workbench which were not implemented are:

- Job Scheduling;
- App/Service registry and discovery;
- Error/Retries management; and
- SLAs tracking, alerting and notification.

The detailed implementation of the OneNet Orchestration Workbench is described in Ch. 6.

OneNet Monitoring and Analytics Dashboard

The OneNet Monitoring and Analytics Dashboard is the component that offers a GUI for facilitating the OneNet Participants in the management, monitoring and analytics of the data exchanges. It is the main User Interface of OneNet Participants for monitoring and data analytics features, as well as for the OneNet Administrators for configuration and administration tools.

Most of the functionalities expected from OneNet that could require a GUI will be implemented in this component.

The main features of the OneNet Monitoring and Analytics Dashboard are:

- administrative and configuration tools;
- easy integration with the OneNet Orchestration Workbench and OneNet Middleware;
- data-analytics dashboard;
- monitoring and alerting dashboard for data processes and platform integrations; and
- user-friendly selection of data sources and services from the catalogues.

The detailed implementation of the OneNet Monitoring and Analytics Dashboard is described in Ch. 7.

3.2 OneNet Use Cases and Requirements

OneNet Deliverable 5.1 [4] provides the overall concept of the OneNet System based on the collection of the Demo System Use Cases and General OneNet System Use Cases.

Starting from these SUCs, D5.1 also provides the list of Functional and Non-Functional Requirements for the overall OneNet System. In this chapter, we analyse the information related to the OneNet Orchestration Workbench and OneNet Analytics and Monitoring Dashboard implementation.

Starting from the analysis of the System Use Cases, it is evident that one of these is of fundamental importance for the purpose of this deliverable: “GSUC_02: AI, Big Data, IoT Data Orchestration for cross-platform services”.

As described in the GSUC_02, two important objectives of the OneNet System and of the OneNet Orchestration Workbench, are: to enable AI, Big Data and IoT data orchestration for cross-platform services; and tracking the performance of the cross-platform services.

The OneNet Orchestration Workbench allows to integrate data coming from the OneNet middleware and implement a data pipeline orchestration.

The results of this analysis are also reflected in the Functional Requirements, and in which are reported in the Table 1.

Table 1: Functional Requirements relevant for OneNet Orchestration Workbench and Dashboard Implementation

Requirement ID	Requirement Name	Description	Reference
OneNet_FUR_27	The OneNet Orchestration Workbench must be able to manage data and service orchestration	The OneNet Orchestration Workbench supports the data orchestration for the evaluation of the performance and scalability of the AI, IoT and Big Data cross-platform services for market and grid operations.	GSUC_02
OneNet_FUR_28	The OneNet Orchestration Workbench must be able to integrate data using the OneNet Middleware	The OneNet Orchestration Workbench allows to integrate data coming from the OneNet middleware and implement a data pipeline orchestration. It also shall include:	
OneNet_FUR_29	The Service Provider must be able to register its service in the OneNet Orchestration Workbench	Job Scheduling App/Service registry and discovery Error/Retries management SLAs tracking, alerting and notification	
OneNet_FUR_30	The Service Provider must be able to create a data		

	workflow using the Orchestration Workbench		
OneNet_FUR_31	The Service Provider must be able to evaluate the performance of its own service		
OneNet_FUR_32	The OneNet Orchestration Workbench shall provide a service catalogue to the OneNet Participants		
OneNet_FUR_33	The OneNet system shall offer a UI dashboard to OneNet Participants for monitoring and analytics		

In addition to the functional requirements, the additional requirements to be considered for the implementation of the OneNet Orchestration Workbench and Monitoring and Analytics Dashboard are described in D5.4 [2].

It was deemed necessary that the integration of the data and services be orchestrated, monitored and evaluated to comply with the specifications and requirements of the OneNet System and in particular that the two components are integrated with the OneNet Decentralized Middleware.

The integration of these two components with the OneNet Decentralized Middleware facilitates the management of the integration of data and services and allows to take advantage of all the features that the OneNet System provides, both in terms of data management and security.

The Figure 5 shows that for the integration of the OneNet Orchestration Workbench and OneNet Monitoring and Analytics Dashboard, two interface layers are required. From one side the integration and communication between the components and the OneNet Decentralized Middleware for the data and services integration and the other side the interfaces for the interaction of the OneNet Participants with the components, exploiting both the User Interfaces and Services available.

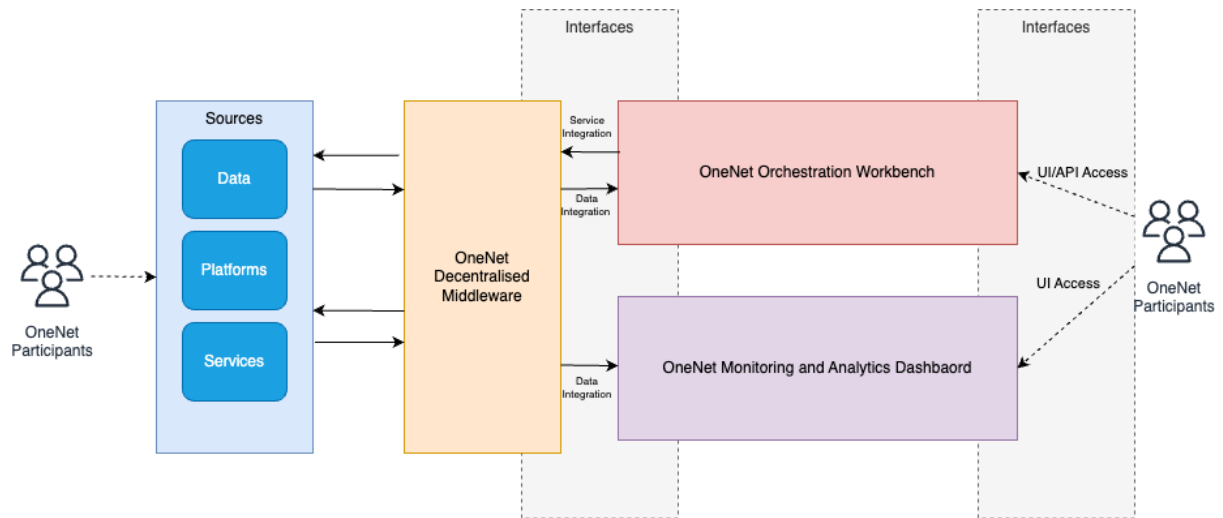


Figure 5: OneNet Interfaces

Additional information about the implemented interfaces is described in Ch. 6.3 and Ch. 7.3.

3.2.1 Requirements refinement

Starting from the requirements obtained in WP5 during the design phase, a new list of more detailed requirements was defined in the context of WP6 activities.

In fact, a preliminary activity conducted in T6.4 in collaboration with T6.1, conducted to a refinement of all the functional requirements to obtain a new detailed list of requirements ready to be used for the implementation phase.

Table 2 reports the refined list of requirements for the OneNet Orchestration Workbench and Monitoring Dashboard.

Table 2: Refined Functional Requirements for OneNet Orchestration Workbench and Monitoring Dashboard

Req Code	Req Subject	Description	Architectural component
FRUC06a	Access OneNet Framework: Browse potential data sources	OneNet Participants shall be able to browse potential data sources, services, and cooperation partners (=other OneNet Participants) and get contact information through the OneNet Framework Dashboard to establish a connection or contractual agreement on data access consent	Orchestration Workbench: Service Catalogue

FRUC06b	Access OneNet Framework: Register or modify account	OneNet Participants shall be able to register themselves with a username and email in the OneNet System with a new account or modify their existing accounts (i.e. change username, email or password) through accessing the OneNet Framework dashboard	Monitoring & Analytics: Administration
FRUC06c	Access OneNet Framework: Monitor overall performance	OneNet Participants shall be able to monitor performance KPIs (to be defined) and results from analytics algorithms (to be defined) in the OneNet Framework dashboard	Monitoring & Analytics: Analytics
FRUC06e	Access OneNet Framework: Login to Framework Dashboard	OneNet Participants shall be able to login to the OneNet Framework dashboard after successful authentication; access to the dashboard shall not be possible without authentication	Monitoring & Analytics: Administration
FRUC06f	Access OneNet Framework: Logout from Framework Dashboard	OneNet Participants shall be able to logout from the OneNet Framework dashboard after successful login	Monitoring & Analytics: Administration
FRRA01.1	OneNet Participant Access: Direct Access to OneNet Monitoring and Analytics Dashboard	Each OneNet Participant must be able to access with unique credentials to the OneNet System	Monitoring & Analytics: Administration
FFRA01.2	OneNet Participant Access: Direct Access to OneNet Orchestration Workbench GUI		Monitoring & Analytics: Administration
FFRA03.7	Middleware Features: Import/Export for analytics	The middleware shall allow the possibility to import /export analytics result	Monitoring & Analytics: Analytics
FFRA04.1	Monitoring and Analytics Tools: Administrative and configuration tools	The Monitoring and Analytics dashboard must include administrative and configuration tools for the administrator and OneNet Participants (see FFRA01)	Monitoring & Analytics: Administration
FFRA04.2	Monitoring and Analytics Tools: Data Analytics Dashboard	The Monitoring and Analytics dashboard must include a dashboard with analytics	Monitoring & Analytics: Analytics

FFRA04.3	Monitoring and Analytics Tools: Monitoring and Alerting Dashboard	The Monitoring and Analytics dashboard must include a dashboard for monitoring data exchanges and setup alert notifications	Monitoring & Analytics: Analytics
FFRA04.4	Monitoring and Analytics Tools: Data Sources Catalogue (UI)	The Monitoring and Analytics dashboard shall provide Data Source Catalogue UI	Monitoring & Analytics: Administration
FFRA04.5	Monitoring and Analytics Tools: Service Catalogue (UI)	The Monitoring and Analytics dashboard shall provide Service Catalogue UI	Monitoring & Analytics: Administration
FFRA06.1	Orchestration Workbench: OneNet Orchestration Workbench GUI	The Orchestration Workbench must offer a GUI to the OneNet Participant through the OneNet Dashboard (see FFRA01)	Monitoring & Analytics: Administration
FFRA06.2	Orchestration Workbench: Evaluate Service Performance		Orchestration Workbench: Performance Evaluation
FRIDS01a.1	ONBOARDING_Acquire identity: Acquire identity for new OneNet participant	Interested party willing to become IDS member makes request form Evaluation Facility	Monitoring & Analytics: Administration
FRIDS01a.2	ONBOARDING_Acquire identity: Acquiring of evaluation for a Service Provider's component	Service provider requests the evaluation of new service component from the Evaluation Facility	Monitoring & Analytics: Analytics
FRIDS01a.3	ONBOARDING_Acquire identity: Certification Body notifies certification authority for successful certification	Validity certificates are provided to certification authority	Monitoring & Analytics: Analytics
FRIDS01a.4	ONBOARDING_Acquire identity: Generating IDS-ID	The Certification Authority generates a unique IDS ID	Monitoring & Analytics: Analytics

FRIDS01a.5	ONBOARDING_Acquire identity: Provisioning of Digital certificate	The Certification Authority issues a digital certificate (X.509) to the participant and notifies the DAPS	Monitoring & Analytics: Analytics
FRIDS01a.6	ONBOARDING_Acquire identity: Register of component at DAPS	Digital certificate is deployed at the side of the component(connector) and the component registers at DAPS	Monitoring & Analytics: Analytics
FRIDS01a.7	ONBOARDING_Acquire identity: DTM Interaction	Dynamic Trust Monitoring (DTM) implements a monitoring function for every IDS Component. The DTM shares information with the DAPS to notify each of the two participants in a data exchange transaction of the current level of trustworthiness of the other participant.	Monitoring & Analytics: Analytics
FRIDS01c.1	ONBOARDING_Security Setup: Issue certificate for IDS participant	Connector interface to enable secure communication contacts Certification Authority to issue certificate to the Data Provider or Data Consumer.	Monitoring & Analytics: Administration
FRIDS01c.2	ONBOARDING_Security Setup: Connector deploys locally IDS certificate	Connector deploys locally IDS' participant certificate and identification of IDS and self-description as received from DAPS	Monitoring & Analytics: Administration
FRIDS03a.1	PUBLISHING AND USING DATA APPS_Data App Certification: App Provider assesses request for a data App	App Provider assesses request for a data App	Orchestration Workbench: Service Catalogue
FRIDS03a.2	PUBLISHING AND USING DATA APPS_Data App Certification: App Provider sends certification request result to Certification Body	App Provider sends certification request result to Certification Body	Orchestration Workbench: Service Catalogue
FRIDS03a.3	PUBLISHING AND USING DATA APPS_Data App Certification: Certification Body performs certification process	Certification Body performs certification process for Data App	Orchestration Workbench: Service Catalogue

FRIDS03a.4	PUBLISHING AND USING DATA APPS_Data App Certification: Certification body issues certificate	Certification body issues certificate for Data App	Orchestration Workbench: Service Catalogue
FRIDS03a.5	PUBLISHING AND USING DATA APPS_Data App Certification: App provider receives certificate for data App	App provider receives and deploys certificate for data App	Orchestration Workbench: Service Catalogue
FRIDS03a.6	PUBLISHING AND USING DATA APPS_Data App Certification: App provider publishes data App at App Store - Provider-	Data App that was successfully certified, the corresponding metadata is stored in the App Store for being retrieved by users (e.g., Data Consumers or Data Providers) via a search interface	Orchestration Workbench: Service Catalogue

4 OneNet Cross-Platform services

Towards the development of appropriate infrastructure for the seamless interconnection of all energy stakeholders – and their IT systems – the OneNet project and its System has provided thorough functional specifications of a set of services, the so called OneNet Cross-Platform Services (see D5.3 [17]). These are defined to assume the service and semantic interoperability among interacting actors. To ensure that system requirements are technically implementable and widely adopted, internationally standardized file formats, metadata, vocabularies and identifiers – are required. Essentially, behind the list of Cross-Platform services there is a standard process rendered which allows for the evolution of this catalogue providing a set of attributes/classes that define a service.

Based on the defined concept for cross-platform services in D5.3 [17], the process starts by analysing inputs from WP4 regarding data exchange patterns and roles involved for SUCs from other H2020 projects and the OneNet demo clusters (WP7 – WP11). As a result, a first list of cross-platform services was gathered. Within the categories, it has been detected redundancy in the cross-platform services within and across projects and demo clusters. To minimize repetition and duplication, all categories were carefully examined and combined. The outcome was a consolidated list of general cross-platform services for each category. This list includes the associated actors (their roles), data senders (producers), and receivers (consumers) as a basis for the following steps in this task. Figure 6 illustrates the identified service categories (bottom of the figure) empowered by the OneNet System Services and products identified in WP2 (top part of the Figure 6).

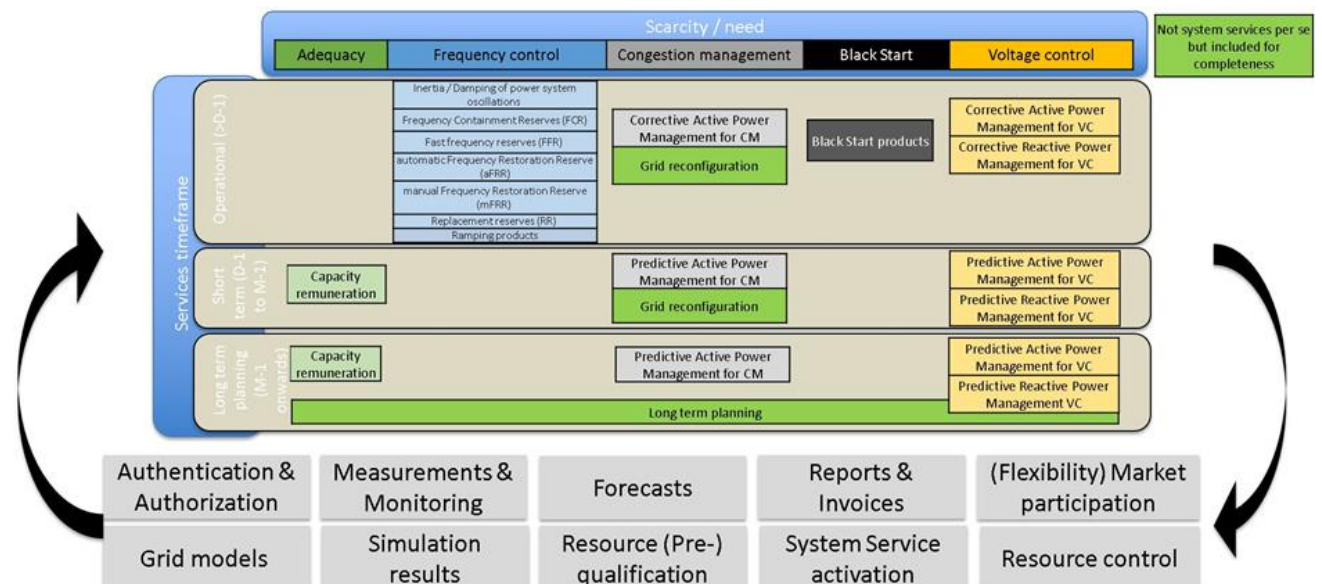


Figure 6: Cross-platform service categories enable the OneNet System Services identified in WP2

The OneNet Cross-Platform services are part of the OneNet Middleware, for all actors involved in the data ecosystem. It is a precondition for any data exchange among two actors (producer and consumer) that there be

an existing service offering from the data provider and an active/accepted subscription for the data consumer. Hence, each actor willing to register a new service offering (under which data will be shared, thereafter), chooses among the Cross-Platform service, which has preloaded semantic (OneNet harmonized- based on IEC profiles, and project base enhancements and extensions) and functional specification.

As depicted in Figure 7, the actors can see all the necessary information on the OneNet cross-platform services along with all meta-data regarding the available service offerings. Based on the available meta-data stored in the OneNet Middleware, actors can find recently introduced services through which they can request the retrieval of their data in decentralized manner, making use of the OneNet connector.

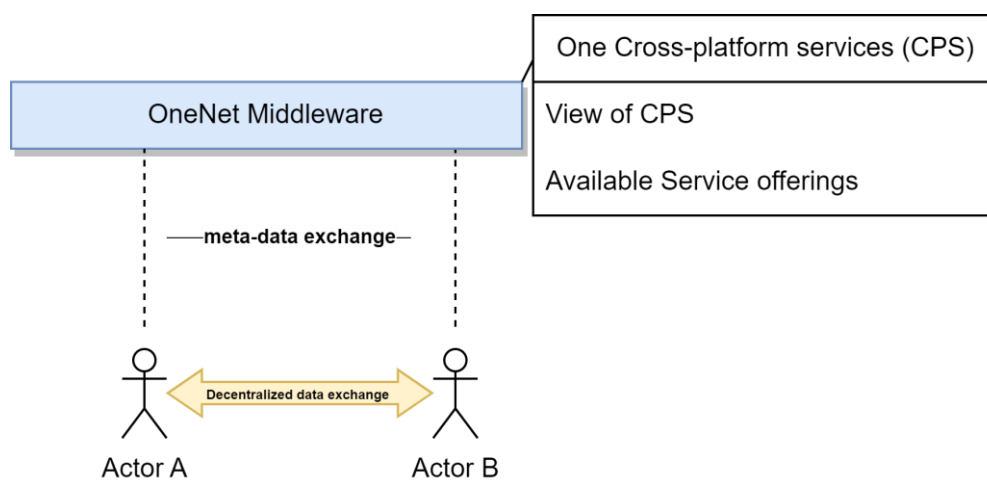


Figure 7: OneNet Middleware, Cross-Platform Services orchestration.

Cross-Platform-Services are accompanied by a set of the corresponding Business Objects that are also described as data-schemata, thus the OneNet Middleware providing a way for uniform description processes (Cross-Platform-Services) and Data (Business Objects).

Cross-Platform-Services are not a static set of services. Users can define and introduce their own customized services. These services can be entirely new or adaptations of existing ones. Similarly, users can also create their own data schemata for the services they provide. This would allow, through extensive use of the Middleware, to result to more “standardized” services and data, without impeding innovation and adaptation to new needs.

5 OneNet Workbench: Integrating Third Party Services

This chapter focusses on the feature of the OneNet Workbench as an App/Service registry and discovery module that allows the streamlined features to integrate third party services which are essential for a data space ecosystem.

Any OneNet Participant, and in particular the Service Providers, can test and evaluate its own service exploiting the OneNet Orchestration Workbench, that allows to integrate data coming from the OneNet Middleware and to implement a data pipeline orchestration.

The Workbench essentially provides the appropriate infrastructure to package and deploy services centrally using Docker containers and Docker images. The Workbench assumes the operational and security challenges of managing multiple Kubernetes and Rancher clusters across any infrastructure, while providing integrated tools for running containerized workloads. This, along with a streamlined UI, allows the streamlined discoverability of newly developed services. The OneNet users can utilize such services by providing their own/consumed data (if this is foreseen by their persisting usage contract). Data-driven services, available at the Workbench level can be used typically for the calculation of analytics or data transformation of the service user data. The users may upload to the Workbench the source data by utilizing their own connector NGSI-LD API.

The following sections describe two third party services (developed on the OneNet's cascaded funding mechanism) which were integrated at the Workbench.

5.1 Presify - Data Fetcher and Anomaly Detection Models

Presify proposed an algorithm as a novel method based on deep recurrent autoencoder ensembles. Deep ensembles are proven to be robust predictors and approximate predictive uncertainty effectively using deep networks as base learners. The solution includes two different services, one to fetch data from the ENTSO-E database and the other one that implements the above-mentioned algorithm for the anomalies detection.

5.1.1 Data Fetcher Service

The Data Fetcher module retrieves XML-formatted data from ENTSO-E database and converts it to JavaScript Object Notation format. The project utilizes Python FastAPI, Pydantic, Uvicorn, and xlmtodict. There are four domains of data that this project retrieves data from, which are load, generation, forecast, and transmission. Each domain has a corresponding route that takes parameters such as periodStart, periodEnd, outBiddingZone_Domain, in_Domain, loadType, generationType, inDomain, and securityToken. The data is returned to the client in a list of JSON format.

5.1.1.1 Service Detail

The Data Fetcher Service includes the following components:

- **FastAPI:** This is a modern, fast, web framework for building APIs using Python 3.6+ with asyncio¹¹ support. FastAPI is used for its speed, simplicity, and ease of use, allowing for rapid development of APIs.
- **Pydantic:** This is a data validation and settings management library that uses Python type annotations. Pydantic is used for its powerful validation capabilities, which ensure that data is properly formatted and structured.
- **Uvicorn:** This is a lightning fast ASGI server that supports HTTP/1.1 and HTTP/2 protocols. Uvicorn is used for its speed and reliability, allowing the service to handle large volumes of data with minimal latency.
- **Xlmtodict:** This is a Python library for converting XML documents to Python dictionaries. Xlmtodict is used to parse the ENTSO-E XML-formatted data into a format that can be easily converted to JSON.

5.1.1.2 Service Workflow

The service follows the following workflow:

1. A client sends a request to one of the four routes, specifying the data domain and type, start and end periods, and any additional parameters required for the specific domain and data type.
2. FastAPI receives the request and validates the input parameters using Pydantic. If the input parameters are invalid, an appropriate error message is returned to the client.
3. FastAPI sends a request to the ENTSO-E database to retrieve the requested data.
4. The ENTSO-E database returns the requested data in XML format.
5. Xlmtodict parses the XML-formatted data into a Python dictionary.
6. The Python dictionary is converted to JSON format and returned to the client in a list.

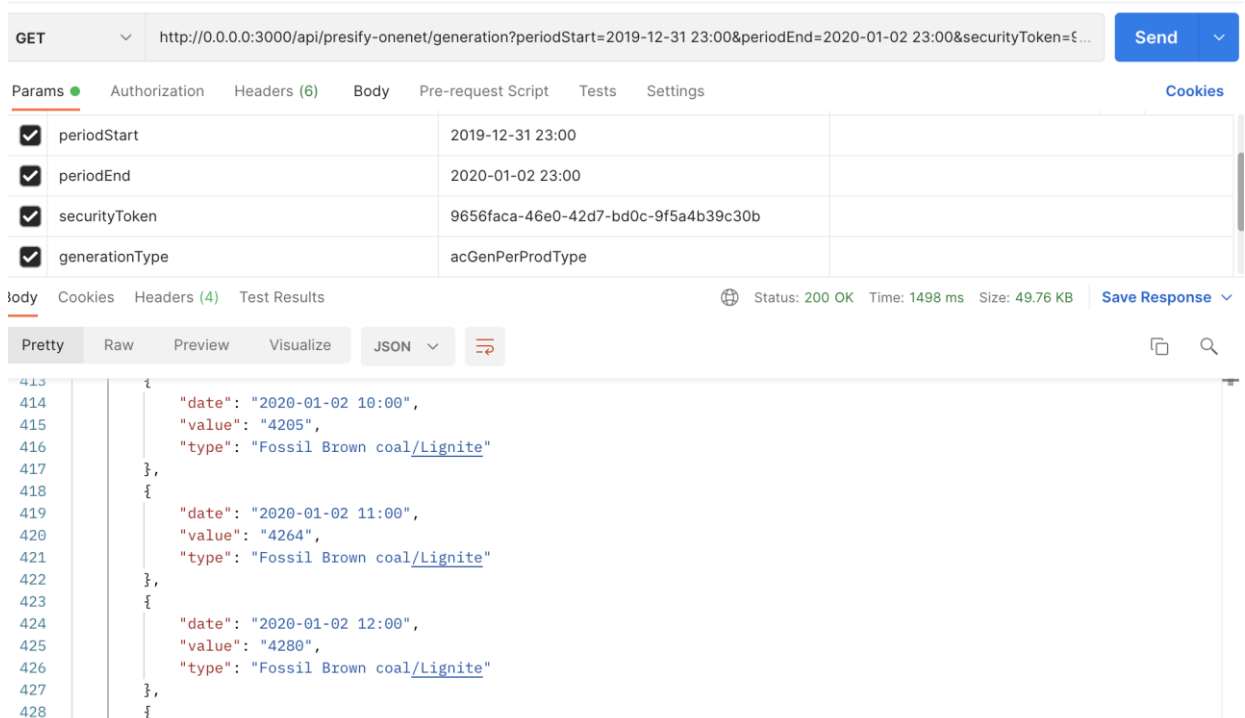
5.1.1.3 Example Request

The following link includes an example request to the data fetcher service.

https://api.postman.com/collections/12291318-4f70e279-6950-4d72-ba68-6fb6e191bb37?access_key=PMAT-01GWHVE5SVMKMGPTF6B4Y6HJW0

¹¹ asyncio is a python library that support an asynchronous mechanism (<https://docs.python.org/3/library/asyncio.html>),

The screenshot below shows the request and response to this request.



GET <http://0.0.0.0:3000/api/presify-onenet/generation?periodStart=2019-12-31 23:00&periodEnd=2020-01-02 23:00&securityToken=9656faca-46e0-42d7-bd0c-9f5a4b39c30b> Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Param	Value
periodStart	2019-12-31 23:00
periodEnd	2020-01-02 23:00
securityToken	9656faca-46e0-42d7-bd0c-9f5a4b39c30b
generationType	acGenPerProdType

Status: 200 OK Time: 1498 ms Size: 49.76 KB Save Response

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```

413 {
414   "date": "2020-01-02 10:00",
415   "value": "4205",
416   "type": "Fossil Brown coal/Lignite"
417 },
418 {
419   "date": "2020-01-02 11:00",
420   "value": "4264",
421   "type": "Fossil Brown coal/Lignite"
422 },
423 {
424   "date": "2020-01-02 12:00",
425   "value": "4280",
426   "type": "Fossil Brown coal/Lignite"
427 },
428 }
```

Figure 8: Data Fetcher Service Example Request

The service provides a simple and efficient way to retrieve data from the ENTSO-E database in JSON format. The use of FastAPI, Pydantic, Uvicorn, and Xlmtodict ensures that the service is fast, reliable, and scalable. The four routes for each data domain allow for flexible data retrieval, while the validation capabilities of Pydantic ensure that the data is properly formatted and structured. Overall, this service is an effective solution for anyone needing to retrieve data from the ENTSO-E database.

The data type for each domain can be specified by the parameters which are loadType, generationType, balancingType, and transmissionType. Here are some examples corresponding to these parameters. actLoad is for Actual Load data, dayAhLoadFor is for day ahead load forecast data, monthAhLoadFor is for month ahead load forecast data, dayAhAggGenFor is for day ahead aggregated generation forecast data, dayAhSolGenFor is for day ahead solar generation forecast data, acGenPerProdType is for actual generation per production type data. There are more values that correspond to some of data types in the four domains.

5.1.2 Anomaly Detection Service

The Anomaly Detection service is designed to perform anomaly detection in the load, generation and other domains in ENTSO-E database. The service provides two routes for anomaly detection on load and generation data, respectively. The service waits for specific data parameters and hyperparameters for anomaly detection models to return the results in the form of a list of JSON schema. According to the parameters it gets, this module

retrieves the specific dataset domain from our Data Fetcher Service. The following describes the details of the service architecture, data and hyperparameters, and the two routes.

5.1.2.1 Service Details

The Python FastAPI service for anomaly detection in the load and generation domain in ENTSO-E database is designed based on the microservice architecture. The service is built using Python FastAPI framework that enables us to build APIs quickly and easily.

The service architecture includes the following components:

- **Anomaly Detection Models:** It includes various anomaly detection models such as Histogram Based Outlier Score, Local Outlier Factor, Robust Covariance, Three Sigma Rule of Thumb and The Proposed Model: Deep Autoencoder.
- When the service is requested, the dataset is retrieved, all the models are trained and anomaly results are returned. The training phase does not last long since the algorithms have relatively basic training procedures.

Parameter List:

- **dataType:** It represents the type of data for anomaly detection. For instance, the Load_Forecast value for the datatype parameter indicates the forecasted load values for specific zone in specific date period. Fossil Coal-derived gas Actual indicates actual generation values from fossil and coal derived gas.
- **startDate:** It represents the start date for data collection. The recommended date range is one year for each request.
- **endDate:** It represents the end date for data collection. The recommended date range is 1 year for each request.
- **outBiddingZoneDomain:** It represents the specific bidding zone for data collection.
- **contamination:** It represents the proportion of outliers in the data. It is a hyperparameter of the anomaly detection models.

Routes:

- **/api/presify-onenet-anomaly/load:** It represents the route for anomaly detection on load data.
- **/api/presify-onenet-anomaly/generation:** It represents the route for anomaly detection on generation data.

Both routes require specific data parameters and hyperparameters for the anomaly detection models, as explained earlier. Once the data is processed, and the anomaly detection models are applied, the service delivers the results in the form of a list of JSON schema.

5.1.2.2 Example Request

The below postman link includes example request to anomaly detection model service.

https://api.postman.com/collections/12291318-4f70e279-6950-4d72-ba68-6fb6e191bb37?access_key=PMAT-01GWHVE5SVMKMGPTF6B4Y6HJW0

The screenshot below shows the request and response to this request.

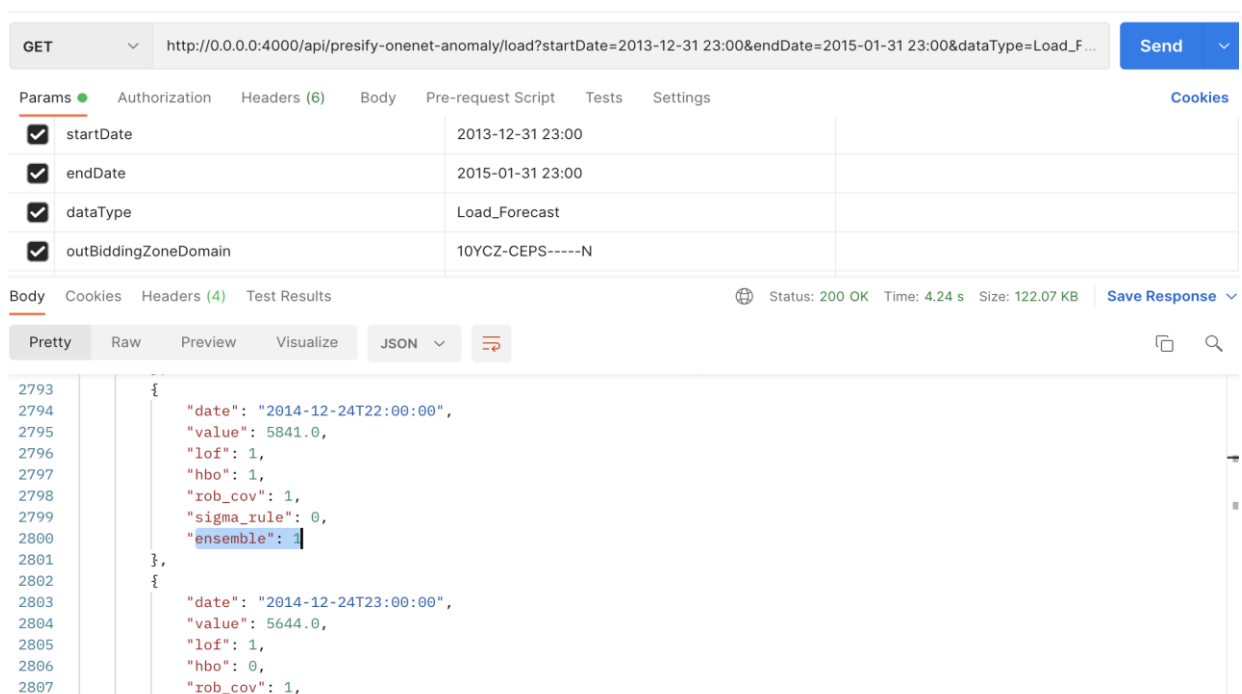


Figure 9: Anomaly detection service example request

Available Datasets for these models:

- datatype for load domain: Load_Actual, Load_Forecast, Week_Load_Forecast.
- datatype for generation domain: Biomass Actual, Fossil Brown coal/Lignite Actual, Fossil Coal-derived gas Actual, Fossil Gas Actual, Fossil Hard coal Actual, Hydro Run-of-river and poundage Actual, Hydro Water Reservoir Actual, Nuclear Actual, Other Actual, Other renewable Actual, Waste Actual, Wind Solar_Generation_Forecast Onshore Actual, Aggregated_Generation_Forecast.

The corresponding country and zone for the outBiddingZoneDomain and inDomain parameters can be seen in the transparency ENTSO-E platform.

5.2 BeeData – Outliers Detection and Imputation

The solution presented by BeeData tackles "Scenario No. 6: Advanced Data Quality Analysis of Data Exchange Platforms." It accomplishes this by actively monitoring and conducting outlier detection on time series data. This process is employed to identify errors in the databases of the Transparency Platform, the power system analysis, the cooperation among actors, and the implementation of flexibility services. The detection of outliers serves as an early warning system to flag errors.

5.2.1 Outlier Detection

Two different data scenarios are considered for the outlier detection methods and different approach is proposed per each scenario:

A) Big Data scenario: Big number (>5K) of single high variable time series. That would be the case of the load or generation from distribution grid through SCADA [7] or energy management systems, and smart meters measurements of consumers and prosumers. The method is based on using Daily Pattern based method to detect the abnormal patterns in data that are considered outliers or anomalies or errors or noise or faults or defects. In this case the method is based on using a daily pattern-based approach to identify abnormal dates in the time series. Clustering is the core of the pattern-based approach. The core of the clustering approach is based on the Gaussian Mixture Model (GMM). The result is the ability to work in a context where the size of training sample grows as time went on, leading to more training time, more computation resources, failing to detect outliers on time.

B) Small data scenario: small number of time series data. That would be the case of national load, generation, and the other time series data from the Transparency Platform, as well as or small business cases from the other scenarios (<5K). The method is based on obtaining a baseline model based on LSTM autoencoders, which is a self-supervised method based on neural networks. Our neural network anomaly analysis can flag the upcoming bearing malfunction well in advance of the actual physical bearing failure by detecting when the data readings begin to diverge from normal operational value.

5.2.2 Imputation method

Each of the time series have specific domain properties and outliers. Although all the time series are energy domain related, each of them has different dynamics depending on different factors. The dynamics of time series can depend on economics, weather, logistics, etc. The used imputation is based on a variation of the KNN regression method over subsampled data. The subsampling criteria is mainly related to recent similar days in terms of calendar and load profile. Calendar similarity is based on labour/non-labour property or weekdays. Daily load profile similarity is calculated using Euclidean distance between partial available load and partial load of all the other days. The ones with low Euclidean distance between daily profiles are picked as similar profile

days. The closest labour or non-labour days with a similar load profile are used by KNN method to evaluate neighbours' similarity. Similarity is used as a weight of the contributions of the neighbours. The imputation result is the weighted average of the neighbour's value. The weight is $1/d$ where d is the distance to the neighbour.

Each of the stages have different customization settings. Detection method customization is used to properly fit the method to the different kinds of time series provided by ENTSO-E.

6 OneNet Orchestration Workbench

The OneNet Orchestration Workbench is the component able to orchestrate and test any kind of data-driven services (e.g. OneNet cross-platform services), completely integrated within the OneNet System.

Through the Orchestration Workbench, any OneNet Participant can deploy, test and evaluate a specific service, integrating data coming from the OneNet Connector and to implement a data pipeline orchestration, supported by analytics and data visualisation features.

6.1 Architecture

The OneNet Orchestration Workbench consists of a four-layer architecture:

- **UI Layer:** access for configuration and visualization of services and data.
- **Microservice Orchestration Layer:** based on Rancher 2.0 and Kubernetes, allows the deployment and orchestration of services.
- **Service Platform Layer:** based the Linux Foundation Energy (LFE) SOGNO [8] platform, allows to integrate data (Data Management Services) and run the deployed services; and
- **Data Integration Layer:** integrates the OneNet Connector data sources via APIs and create a bridge with the Service Platform

The OneNet Orchestration Workbench architecture is shown in Figure 10.

UI Layer

The UI Layer includes the User Interfaces available for all the OneNet Participants. It includes a Web Dashboard with the following sections:

- Login
- Service Catalogue
- Service Deployment
- Running and Testing
- Data Visualization.

Microservice Orchestration Layer

- The Microservice Orchestration Layer is based on based on Rancher 2.0 [9] and allows the deployment and orchestration of data-driven services.
- The deployment of the services is based on Kubernetes [10] and Helm framework [11], supporting the automating creation, packaging, configuration, and deployment of Kubernetes applications by combining configuration files into single reusable packages.

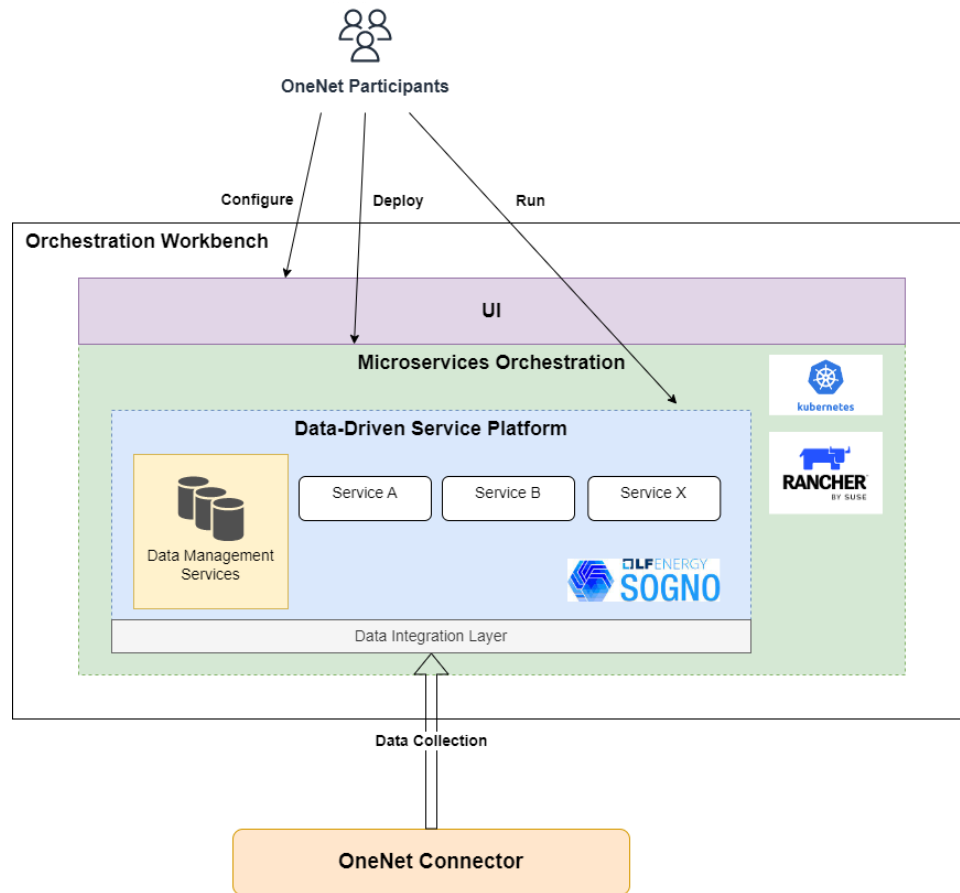


Figure 10: OneNet Orchestration Workbench Architecture

Service Platform Layer

The Service Platform Layer is based on the open source SOGNO Platform, released as a Linux Foundation Energy project [8]. The SOGNO (Service-based Open-source Grid automation platform for Network Operation of the future) Platform is a plug-and-play solution for microservices based architecture that supports data-driven service execution, integration and monitoring. Figure 11 below shows the SOGNO platform architecture.

The SOGNO platform was adapted and configured for implementing the two main goals of the Service Platform Layer:

- Service Running
- Data Management

The Service Platform Layer allows running the data-driven microservices using standardized tools or the GUI. **The Data Management** component supports the service deployment and integrated data, which are stored into specific databases. At the moment two kind of database are supported: InfluxDB [12] for timeseries (e.g. measurements) and MongoDB [13] for any other kind of data. Databases and storage mechanisms can be extended.

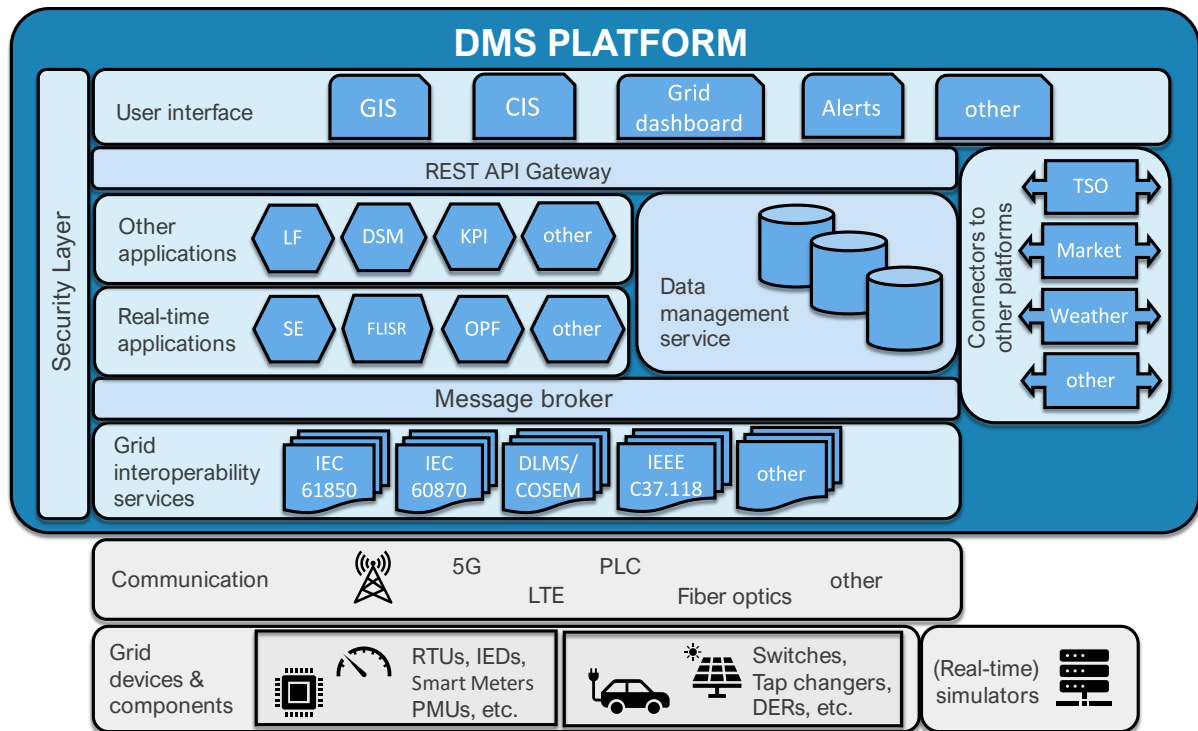


Figure 11: SOGNO architecture [8]

Data Integration Layer

This Data Integration Layer is integrated with the OneNet Connector Local API, allowing the integration of data sources. It uses a common authorization and authentication mechanisms based on the OneNet Identity Manager.

In the upper layer, the Data Integration Layer supports two different communication mechanisms: synchronous and asynchronous.

Synchronous communication is implemented in the via REST APIs. The asynchronous communication is implemented in the MQTT Broker and can be extended with other Message Brokers (e.g., Apache Kafka).

6.2 Functionalities

The main purpose of the OneNet Orchestration Workbench is to allow the deployment, testing and evaluation of specific services, exploiting the data integration of the OneNet Connector.

For this reason, the roles of Data Provider and Consumer, that are key roles for the data exchange in the OneNet Decentralized approach, are joined by another important one: the Service Provider.

In OneNet project, as described in D5.1 [4], Participants can take on several data-driven roles. Among these, the role of the Service Provider assumes a relevant importance in the Orchestration Workbench, since its main objective is precisely linked to the execution and evaluation of services.

The Figure 12 shows a basic workflow example with the main steps for a data-driven service evaluation:

- OneNet Participant (Data Provider) **creates new data** using its own connector.
- OneNet Participant (Data Consumer) **subscribes/consumes data** using its own connector.
- OneNet Participant (Service Provider) **deploys a data-driven service**.
- OneNet Participant (Data Consumer) can **run services** using consumed data.

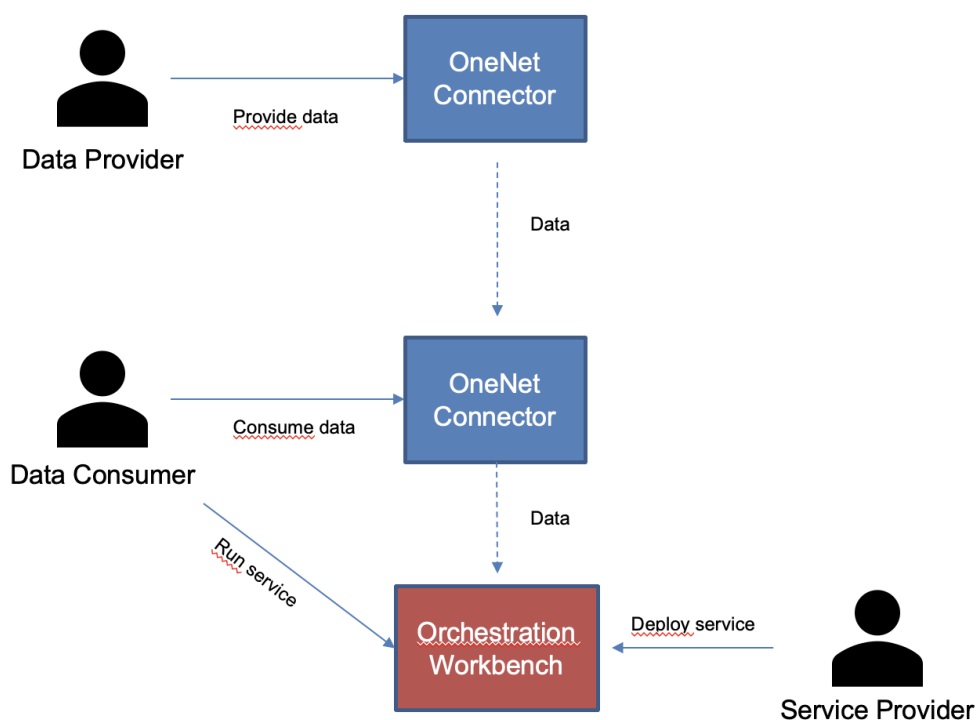


Figure 12: OneNet Orchestration Workbench Workflow Example

Service deployment and data integration are the main functionalities offered by the OneNet Orchestration Workbench. In addition, it offers some pre-installed service for providing additional features:

- SLAs tracking, analytics, alerting and notification.
- Data visualization services (e.g., measurements, grid topology).

6.3 Interfaces and communication mechanisms

The OneNet Orchestration Workbench has several external interfaces for interacting with the other actors of the OneNet ecosystem.

As described in D5.4 [2], one of the main non-functional requirements for the OneNet Orchestration workbench is to provide specific interfaces for interacting with the overall OneNet System.

Figure 5 in Ch. 3.2 shows how the OneNet Orchestration Workbench can interface itself with the OneNet Middleware (more precisely with the OneNet Connector) and offer a GUI and REST APIs to any OneNet Participants.

In Table 3 are reported the implemented external interfaces for accessing the Orchestration Workbench and for integrating the OneNet Decentralized Middleware.

Table 3: OneNet Orchestration Workbench external interfaces

Interface ID	From	To	Type	Description
int0-OW	OneNet Orchestration Workbench	OneNet Middleware (Connector)	API	Interface between OneNet Middleware and the OneNet Orchestration Workbench. The OneNet Orchestration Workbench should exploit FIWARE Context Broker implementation for integrating data sources and services.
int1-OW	OneNet Middleware	OneNet Orchestration Workbench	API	Interface offered by OneNet Orchestration Workbench for including additional services in the OneNet Decentralized Middleware (e.g., register news services or retrieving Service Catalogue information).
Int2-OW	OneNet Participant (User)	OneNet Workbench	GUI	Graphical Interfaces for accessing, configure and utilize the OneNet Orchestration Workbench
Int3-OW	OneNet Participant (User)	OneNet Orchestration Workbench	Open API	Standard OpenAPI interfaces for exploiting the services offered by the OneNet Orchestration Workbench in automated way

In addition, the Orchestration Workbench Data Integration Layer offers the possibility to provide data to the Service Platform Layer, then to all deployed services, through several communication mechanisms. In particular,

the data provided through the OneNet Connector can be made available to the deployed services using **MQTT**, **REST APIs** or **Messaging Broker with publish/subscribe** mechanism.

6.3.1 Graphical User Interface (GUI)

As introduced in the previous paragraphs, the OneNet Orchestration Workbench provides a GUI to any OneNet Participants, offering the possibility to:

- Login through OneNet credentials (OneNet Identity Manager)
- Integrate data coming from OneNet Connector.
- Deploy, Orchestrate and Monitoring services.
- Explore Service Catalogue.
- Run services.
- Evaluate performance.

Login

The login section allows any OneNet Participant to access the OneNet Orchestration Workbench using its own OneNet credentials. The OneNet Orchestration Workbench authentication mechanisms are synchronized with the OneNet Connector and Decentralized Middleware, ensuring a unique identification mechanism.

The Figure 13 below shows the login section in the Orchestration Workbench GUI.

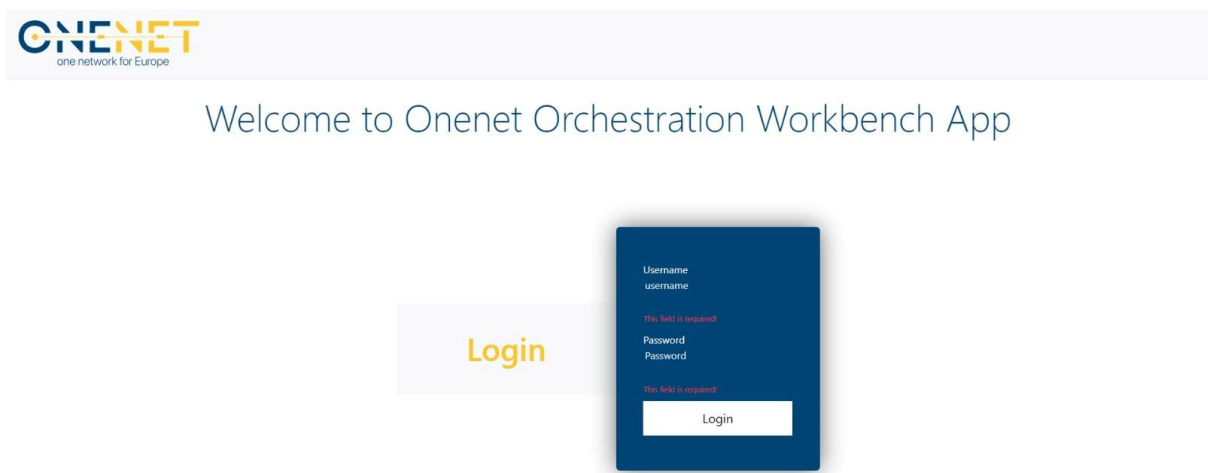


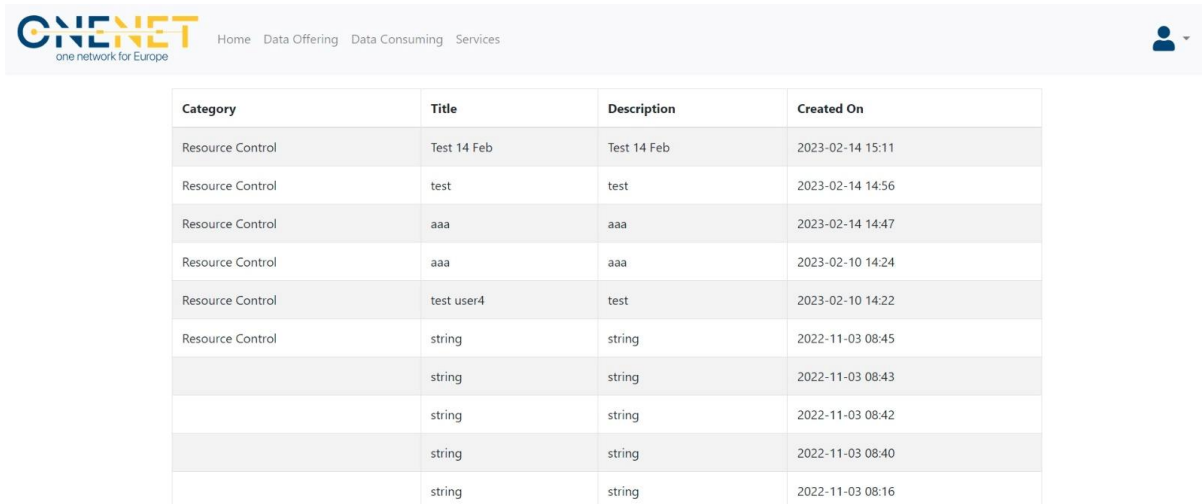
Figure 13: OneNet Orchestration Workbench Login section

Data Integration

The OneNet Orchestration Workbench is completely integrated with the OneNet Connector, exploiting the OneNet Local-APIs (see D6.1 [15] for additional details about the APIs).

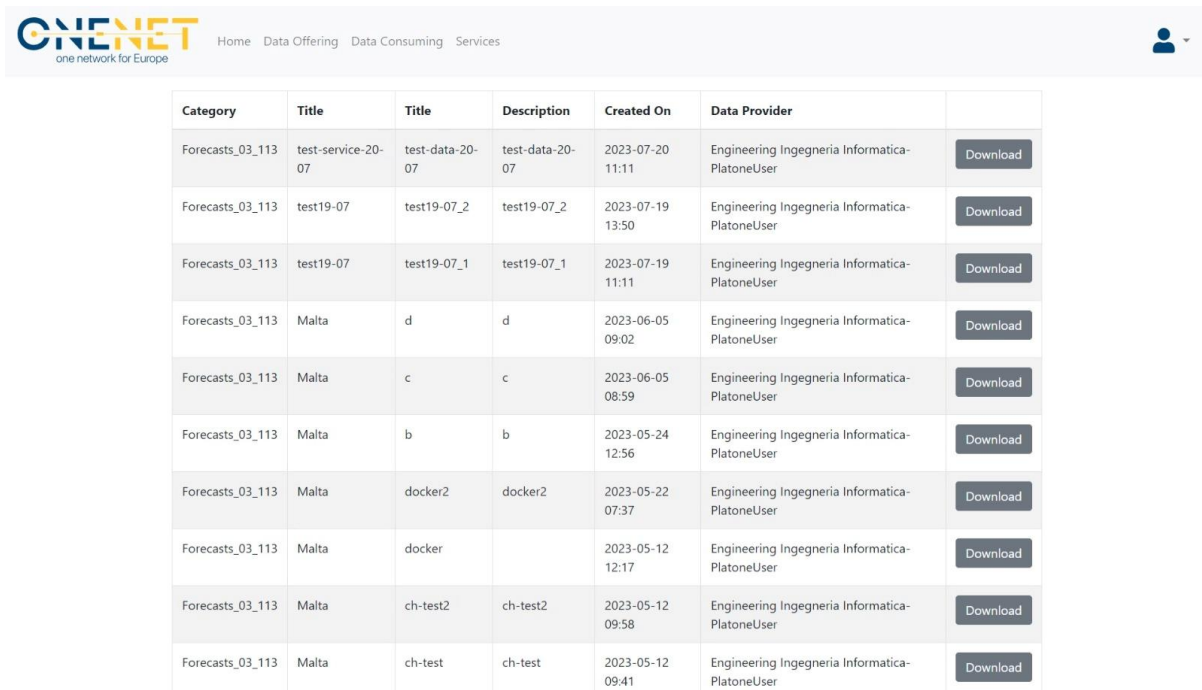
The Workbench can show all the data offerings, subscriptions and data consumed using the OneNet Connector. No data are stored in the OneNet Orchestration Workbench, unless a user explicitly wants to transfer data to a specific OneNet service.

The two figures below show the data offerings (Figure 14) and data consumed (Figure 15) retrieved from the OneNet Connector APIs.



Category	Title	Description	Created On
Resource Control	Test 14 Feb	Test 14 Feb	2023-02-14 15:11
Resource Control	test	test	2023-02-14 14:56
Resource Control	aaa	aaa	2023-02-14 14:47
Resource Control	aaa	aaa	2023-02-10 14:24
Resource Control	test user4	test	2023-02-10 14:22
Resource Control	string	string	2022-11-03 08:45
	string	string	2022-11-03 08:43
	string	string	2022-11-03 08:42
	string	string	2022-11-03 08:40
	string	string	2022-11-03 08:16

Figure 14: OneNet Orchestration Workbench Data Offerings section



Category	Title	Title	Description	Created On	Data Provider	
Forecasts_03_113	test-service-20-07	test-data-20-07	test-data-20-07	2023-07-20 11:11	Engineering Ingegneria Informatica-PlatoneUser	Download
Forecasts_03_113	test19-07	test19-07_2	test19-07_2	2023-07-19 13:50	Engineering Ingegneria Informatica-PlatoneUser	Download
Forecasts_03_113	test19-07	test19-07_1	test19-07_1	2023-07-19 11:11	Engineering Ingegneria Informatica-PlatoneUser	Download
Forecasts_03_113	Malta	d	d	2023-06-05 09:02	Engineering Ingegneria Informatica-PlatoneUser	Download
Forecasts_03_113	Malta	c	c	2023-06-05 08:59	Engineering Ingegneria Informatica-PlatoneUser	Download
Forecasts_03_113	Malta	b	b	2023-05-24 12:56	Engineering Ingegneria Informatica-PlatoneUser	Download
Forecasts_03_113	Malta	docker2	docker2	2023-05-22 07:37	Engineering Ingegneria Informatica-PlatoneUser	Download
Forecasts_03_113	Malta	docker		2023-05-12 12:17	Engineering Ingegneria Informatica-PlatoneUser	Download
Forecasts_03_113	Malta	ch-test2	ch-test2	2023-05-12 09:58	Engineering Ingegneria Informatica-PlatoneUser	Download
Forecasts_03_113	Malta	ch-test	ch-test	2023-05-12 09:41	Engineering Ingegneria Informatica-PlatoneUser	Download

Figure 15: OneNet Orchestration Workbench Subscriptions and Data Consumption section

Service Deployment, Orchestration and Monitoring

As described in Ch.6.1, the Microservice Orchestration Layer is based on Rancher 2.0 and Kubernetes. This platform allows to deploy, monitor and orchestrate service in automatic way.

The Rancher GUI is available only for the administrator of the Orchestration Workbench, while all the provided functionalities (deploy, monitoring, etc.) are available via APIs through the Orchestration Workbench GUI. The figures below show some sections of Rancher GUI: login (Figure 16), deployed services (Figure 17), services status and monitoring (Figure 18).

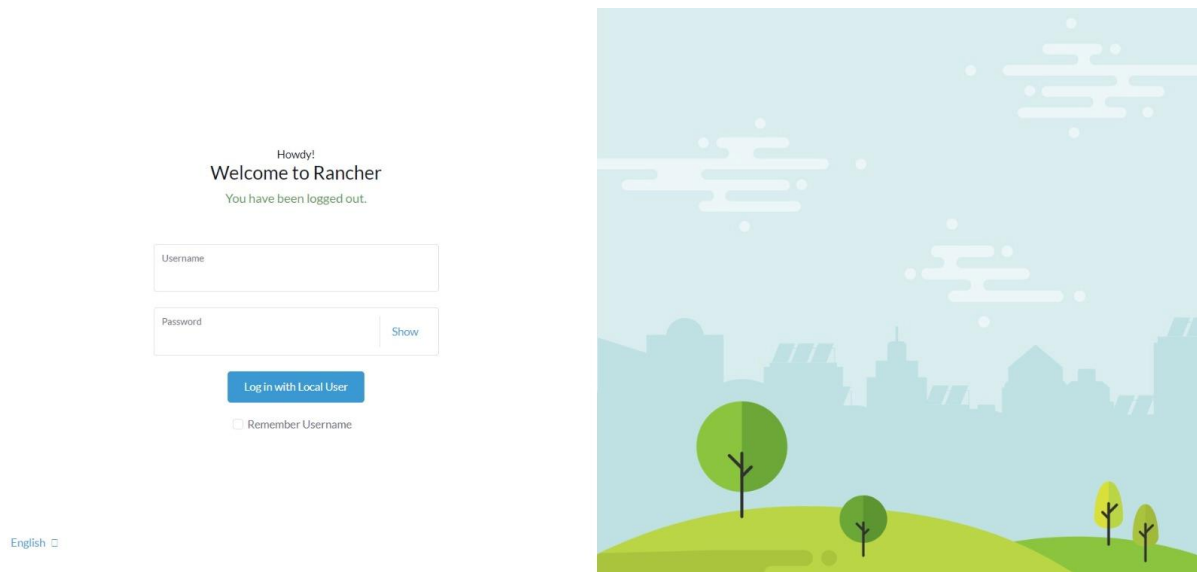


Figure 16: Microservice Orchestration Layer - Rancher Login

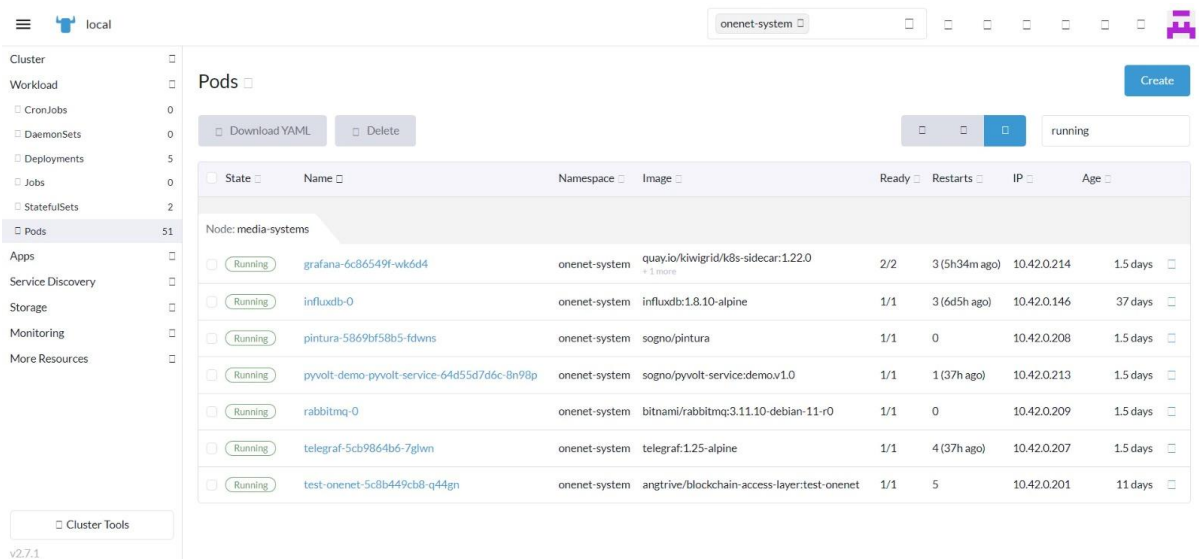


Figure 17: Microservice Orchestration Layer - Deployed services

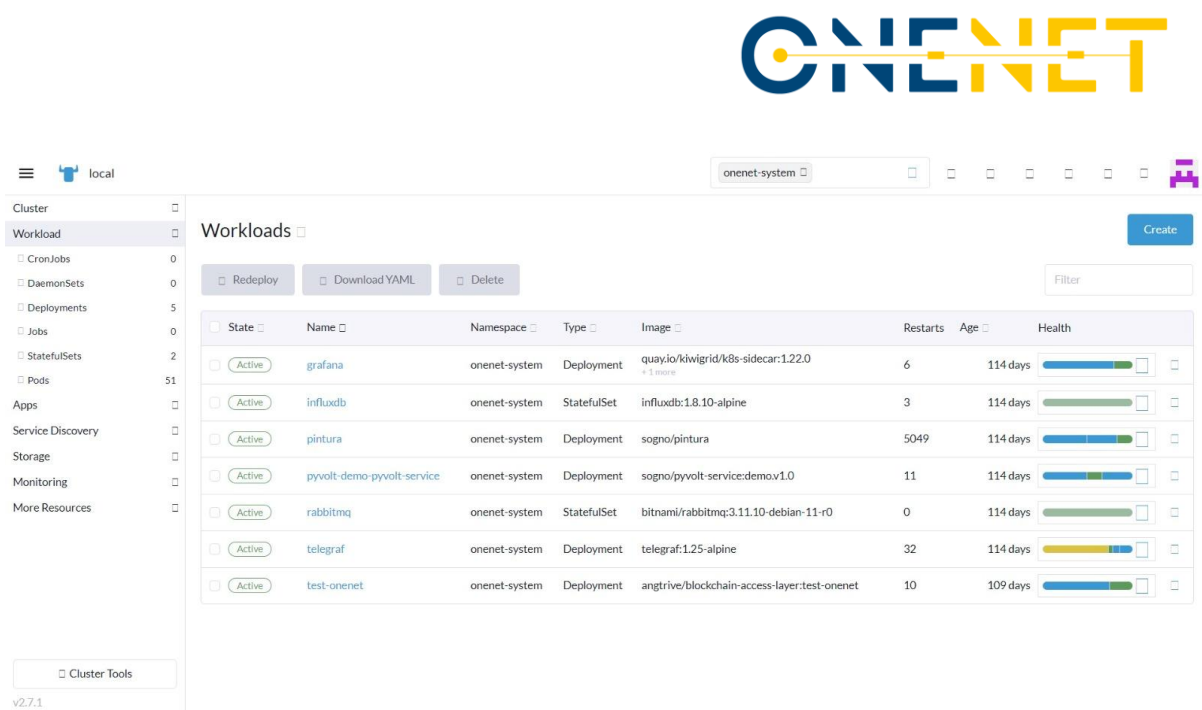


Figure 18: Microservice Orchestration Layer - Services monitoring

Service Catalogue

All the services deployed are integrated in the Service Layer with the Data coming from the OneNet Connector. The OneNet Orchestration Workbench provides an interface for exploring the Service Catalogue and testing any available service with integrated data.

The Figure 19 below shows an example of Service Catalogue with two services ready to be tested.

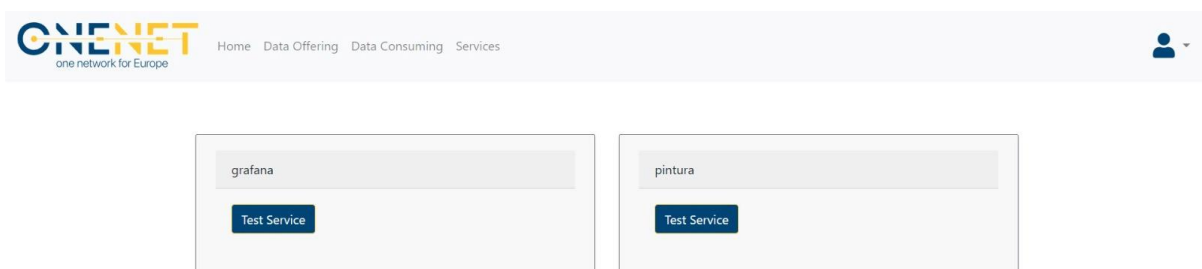


Figure 19: OneNet Orchestration Workbench - Service Catalogue

Service Execution

As described in the previous paragraph, any service available in the Service Catalogue can be tested using OneNet Connector integrated data. The Figure 20 shows an example of service execution, in which the OneNet Participant can select the data to be provided to Grafana for data visualization.

Sending data to Grafana						
Category	Title	Title	Description	Created On	Data Provider	
Forecasts_03_113	test-service-20-07	test-data-20-07	test-data-20-07	2023-07-20 11:11	Engineering Ingegneria Informatica-PlatoneUser	<button>Send</button>
Forecasts_03_113	test19-07	test19-07_2	test19-07_2	2023-07-19 13:50	Engineering Ingegneria Informatica-PlatoneUser	<button>Send</button>
Forecasts_03_113	test19-07	test19-07_1	test19-07_1	2023-07-19 11:11	Engineering Ingegneria Informatica-PlatoneUser	<button>Send</button>
Forecasts_03_113	Malta	d	d	2023-06-05 09:02	Engineering Ingegneria Informatica-PlatoneUser	<button>Send</button>
Forecasts_03_113	Malta	c	c	2023-06-05 08:59	Engineering Ingegneria Informatica-PlatoneUser	<button>Send</button>
Forecasts_03_113	Malta	b	b	2023-05-24 12:56	Engineering Ingegneria Informatica-PlatoneUser	<button>Send</button>
Forecasts_03_113	Malta	docker2	docker2	2023-05-22 07:37	Engineering Ingegneria Informatica-PlatoneUser	<button>Send</button>
Forecasts_03_113	Malta	docker		2023-05-12 12:17	Engineering Ingegneria Informatica-PlatoneUser	<button>Send</button>
Forecasts_03_113	Malta	ch-test2	ch-test2	2023-05-12 09:58	Engineering Ingegneria Informatica-PlatoneUser	<button>Send</button>
Forecasts_03_113	Malta	ch-test	ch-test	2023-05-12 09:41	Engineering Ingegneria Informatica-PlatoneUser	<button>Send</button>
						<button>Close</button>
Forecasts_03_113	Malta	docker		2023-05-12 12:17	Engineering Ingegneria Informatica-PlatoneUser	<button>Send</button>

Figure 20: OneNet Orchestration Workbench - Service execution and data integration

Performance evaluation

The orchestration layer includes an open-source monitoring solution based on Prometheus.io. It allows to monitor any service deployed in the Orchestration Workbench and to provide insights and for alerting.

The figures below show how the monitoring service, integrated with Grafana can monitor the overall system (Figure 21) and a specific service (Figure 22).

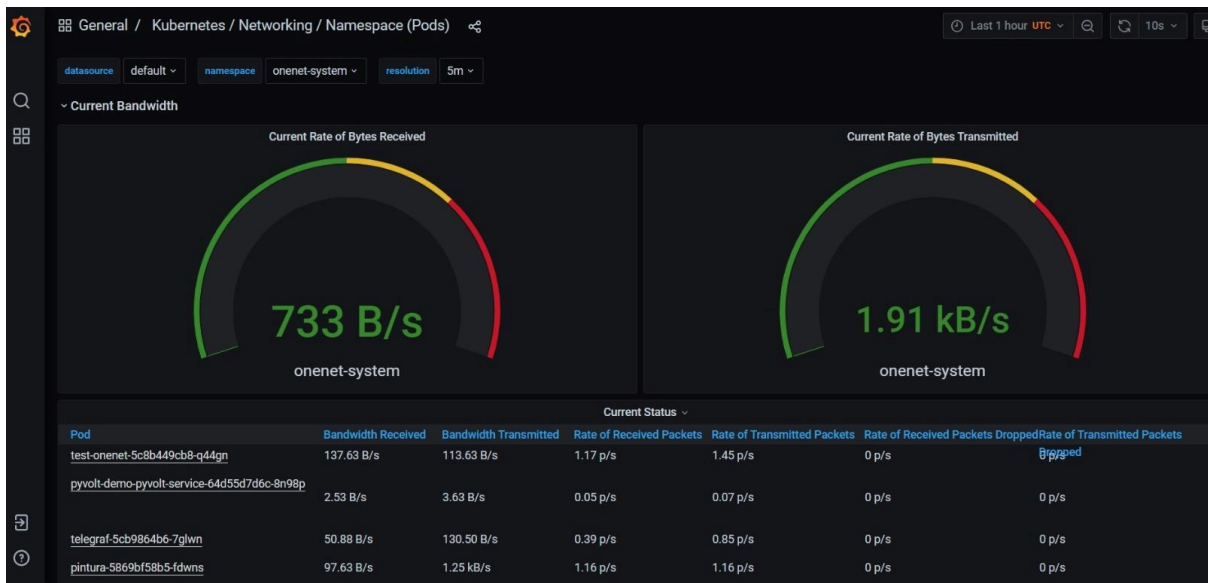


Figure 21: OneNet Orchestration Workbench - Overall system evaluation

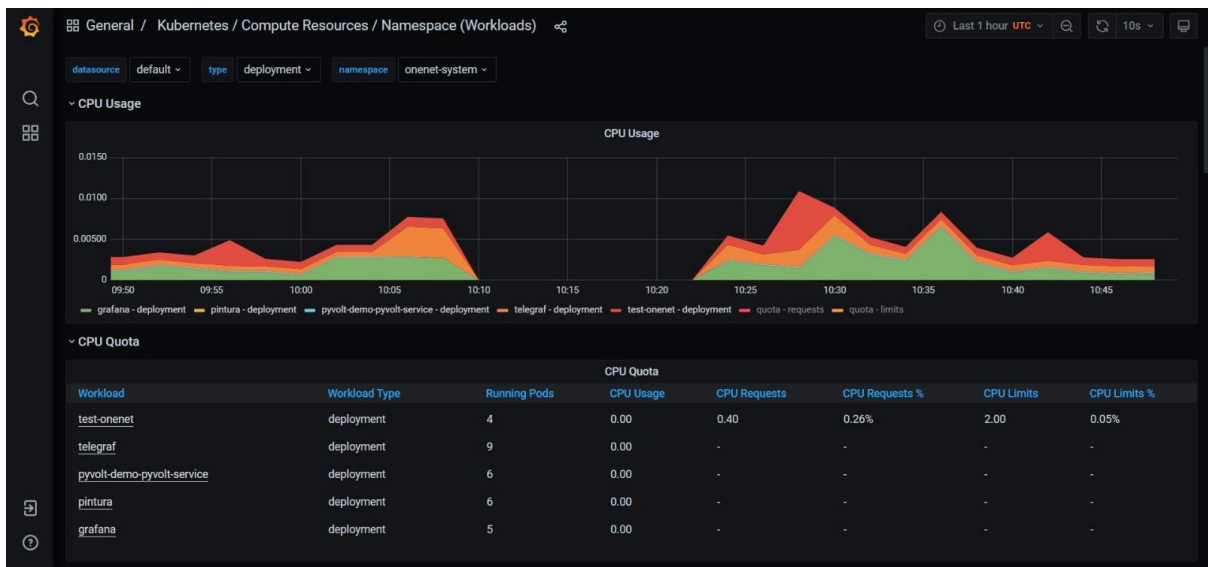


Figure 22: OneNet Orchestration Workbench - Service monitoring

6.4 Languages and Software

Table 4: OneNet Orchestration Workbench - Languages and software

Layer/Component	Languages	Technologies/Framework	External Tools
UI Layer	JavaScript HTML5 CSS/SCSS	Docker React	Nginx Grafana
Microservice Orchestration		Docker Kubernetes	
Service Platform	Python JavaScript C++	Docker Nodejs REST APIs	RabbitMQ Apache Kafka Prometheus
Data Integration Layer	JavaScript	Docker Nodejs REST APIs	MongoDB InfluxDB Telegraf

6.5 Packaging and deployment

As described in the previous paragraphs, the OneNet Orchestration Workbench is a centralized component of the overall OneNet System, integrated with the OneNet Decentralized Middleware and the OneNet Connector through a specific interface.

For the OneNet project, it was deployed in a CentOS Linux 7 virtual machine with 16 GB of RAM, 100GB of storage and 4 CPU cores allocated to it. A container-based approach was adopted for the deployment of the necessary services and Kubernetes instance for the orchestration.

The Orchestration Workbench is publicly available under authentication at the following Url: <https://smart-energy.eng.it>

7 OneNet Monitoring and Analytics Dashboard

This section provides an overview of the Monitoring and Analytics Dashboard service architecture and a description of the services comprising its implementation. Furthermore, it illustrates the interconnection of the various components it is composed of and how they are packaged and deployed to production.

7.1 Architecture

The following figure depicts the architecture of the OneNet Monitoring and Analytics Dashboard.

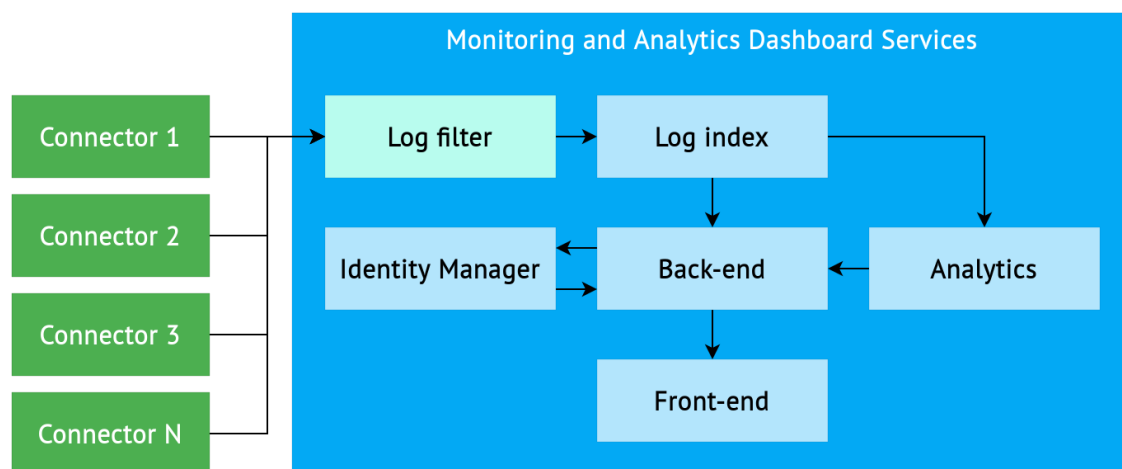


Figure 23: OneNet Monitoring and Analytics Dashboard Service Architecture

A log is generated each time a client makes an action in the connector web UI which triggers a network request. Each connector has been configured to forward these logs to our log filter using HTTP communication.

The log filter constitutes the entry point of all input of the Monitoring and Analytics Dashboard. It filters incoming connector logs by converting them to the JSON format and enriching them with metadata. The filtered logs are then forwarded to the log index service where they are permanently stored. The log index serves both as storage and as a search and analytics engine so that various queries and aggregations are performed in an efficient manner.

The logs obtained from the log index are now able to be further processed and analyzed by the analytics service. This analysis involves using machine learning algorithms to categorize client behavior as either normal or abnormal. The analytics service offers an API endpoint. The analytics service provides an API endpoint that allows users to execute the model and receive behavior classifications for recent clients.

The back-end service is responsible for the implementation of a REST API which queries the log index and uses its aggregation methods to fetch data which can be consumed by the front-end service. Moreover, in

conjunction with the identity manager, the back-end offers authentication and authorization capabilities. For this purpose, it interfaces with the identity manager where the dashboard users are created and stored.

Finally, the front-end service can use the API endpoints provided by the back-end to handle the user registration and log in procedures and fetch all the necessary data to be displayed in a diverse number of charts.

7.2 Services

The Log filter ingests incoming connector logs formatted in the Common Log Format (CLF), a standardized log format used by a number of web servers to generate access logs. It extracts various fields from each log using text patterns, in particular features needed for analysis such as bytes sent and response code. Moreover, it can define various mutations and transformations to be applied to the extracted fields and supplement them with metadata. For instance, using the extracted client IP, it can retrieve GeoIP information of the client who made the request. Finally, it compiles this data into a JSON object which is forwarded to the log index service.

The Log index receives formatted logs forwarded by the log filter service. Logs are stored as JSON documents and indexed based on the date they were received, which is particularly helpful when executing queries that perform date histogram aggregations. The log index serves not only as storage and search engine but also offers the functionality of an analytics engine that performs specialized queries in an efficient manner. It provides an interface through which the analytics and back-end services may send their queries and retrieve the results.

The Analytics service implements anomaly detection for malicious IPs visiting the connectors. Malicious IPs are defined as the IPs that perform multiple requests in a small amount of time with the intention to overwhelm the server and make the web site unresponsive. The anomaly detection model is being trained with the nginx logs using the Isolation Forest algorithm and it is being re-trained every 2 days by using the latest nginx logs. The prediction is being done automatically every 2 minutes by collecting the IPs in batches. The results are stored into a PostgreSQL database.

The Identity Manager offers user management, authentication and authorization capabilities. It is responsible for providing and refreshing access tokens to the back-end service upon user authentication and utilizes an SQL database to store and manage users, user roles and applications. Apart from its administration capabilities, it also provides methods for configuring a number of security-related settings and functions, such as signature algorithms, token lifespans, Cross-Site Scripting (XSS) protection and brute force detection.

The Back-end service provides API endpoints that cover all the functionality required by the OneNet Monitoring and Analytics dashboard. It implements user management and authentication-related endpoints that allow user registration, logging in, logging out and token refreshing by interfacing with the identity manager. It also queries the log index to retrieve aggregated data which is then inserted into a Data Transfer Object (DTO)

and returned as a response. Furthermore, it implements an endpoint that requests a classification of recent clients from the analytics service and returns the result of this classification. Endpoints that return data to be consumed by the dashboard are secured using access tokens generated by the identity manager so that they may only be used by authenticated dashboard users.

The Front-end uses frameworks and libraries for web user interface development and data visualization tools to display historic and real-time data arranged in a number of charts. It renders the required forms to allow user registration and login and calls the API endpoints implemented by the back-end service for user authentication. Finally, using the returned access tokens it can fetch data from the back end to populate the rendered charts and display the content in user-friendly manner.

The dashboard implemented by the Front-end service offers register and login forms for authentication purposes. Upon login, the users are presented with several graphs that depict statistics such as number of requests per country, number of requests per hour, anomaly detection and more. Account management functionality has also been integrated into the dashboard. Through the dashboard account settings, users may access the Account Management page provided by the Identity Manager where they are able to configure their account information and authentication settings, as well as view logged in applications and monitor device activity. It is noted that a more comprehensive description of the dashboard and the integrated account management functionality it offers is provided by deliverable D6.6 [6].

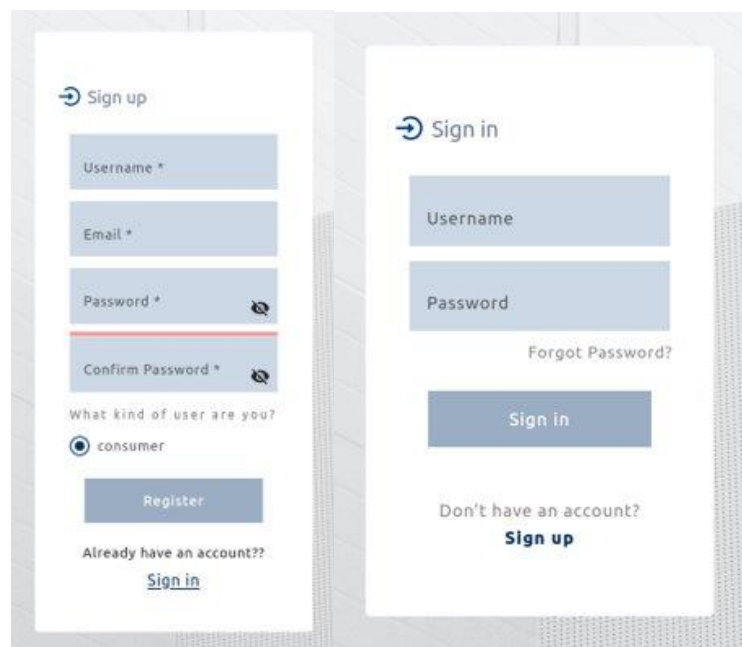


Figure 24: Dashboard Login and Registration Sections



Figure 25: Dashboard map chart displaying number of requests per country for the last 30 days

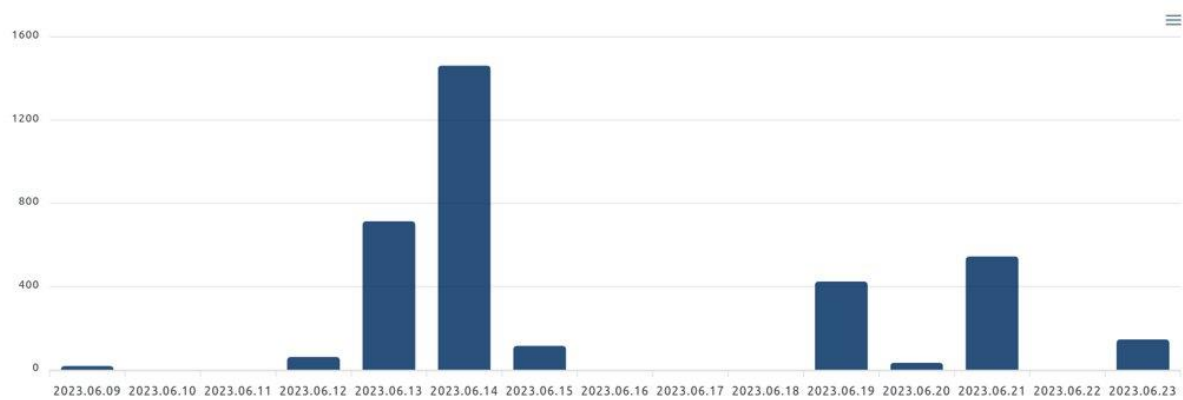


Figure 26: Dashboard bar chart displaying number of requests per day for the last 14 days

home > settings

settings

[basic-info](#)
[Avatar](#)
[time-zone](#)

Username
test-user

Email
testmail@gmail.com

Manage account information

Figure 27: Dashboard Account Settings Component

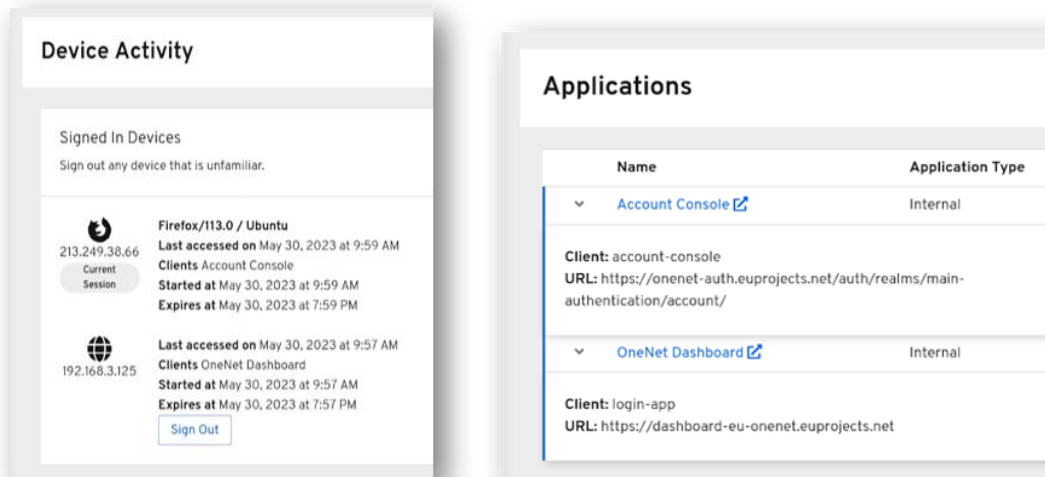


Figure 28: Account Management Page: Device Activity and Logged in Applications

7.3 Interfaces and communication mechanisms

As described in D6.3 [16], log collection is achieved by forwarding logs from each connector to a centralized destination where they may be accessed, processed, and analyzed. Connectors are configured to forward their logs to a specific domain which targets the IP and port of our log filter service. Therefore, logs are piped to the log filter using HTTP communication.

The log filter service defines the IP and port it listens to for incoming logs and specifies the destination IP and port of its JSON output. Thus, the log index receives its input from the log filter and exposes endpoints that accept highly customizable queries. The back end and analytics services can form queries and fetch data from the log index using its REST API.

The back-end service integrates with the API implemented by the Identity Manager for authentication and authorization purposes and exposes several API endpoints to be used by the front-end service. Finally, the front-end consumes data through the endpoints exposed by the back end.

7.4 Languages and Software

This section provides an overview of the languages and software used for the implementation of the OneNet Monitoring and Analytics Dashboard components.

The Elasticsearch, Logstash and Kibana (ELK) stack is used for log collection, filtering, indexing and analytics.

For the analytics service, Python has been used, and libraries related to data analysis and machine learning, such as pandas, numpy and sklearn. A PostgreSQL database is used for storing the different IPs visiting the connectors and their status.

The Keycloak Identity Manager in conjunction with a PostgreSQL database is used for user management as well as authentication and authorization purposes.

For the back-end service, Spring Boot has been selected, a Java framework that facilitates the development of web applications and microservices. It interfaces with Elasticsearch and Keycloak using their respective Java APIs.

The front-end is developed using Angular, a framework for developing dynamic web applications in the Typescript language. It is hosted by an nginx server which also serves as a reverse proxy to the Spring Boot REST API.

A comprehensive description of the implementation of the various Monitoring and Analytics Dashboard components, as well as the multitude of tools and software utilized for this purpose, is included in D6.6.

7.5 Packaging and deployment

The OneNet Monitoring and Analytics Dashboard is a centralized solution able to be deployed on bare metal or on a virtual machine. For the OneNet project, it was deployed in a CentOS Linux 7 virtual machine with 32 GB of RAM, 200GB of storage and 4 CPU cores allocated to it. At least 15 GB of HDD storage is needed for the OS and the required software and tools to be installed.

OneNet adopted a container-based approach for the deployment of the necessary services. The container deployment is orchestrated using docker and docker-compose. Each image is built from the repository which implements the respective service and then pushed to the container registry in the company's Gitlab instance. From there, it may be pulled and used for testing or deployment to production. Other images which serve as dependencies, such as postgres:10, are either pulled from official sources or a stable version pushed to the company's docker container registry.

The following containers are started upon deployment:

- logstash: A Logstash instance which accepts logs from all connectors through HTTP.
- elasticsearch: An Elasticsearch instance where filtered logs are stored.
- kibana: A Kibana dashboard for our Elasticsearch search instance
- onenet: The Spring Boot web service which implements the back-end.
- keycloak: An instance of the Keycloak v15.1.0 identity manager

- keycloak-db: A PostgreSQL image serving as the database for the keycloak service.
- frontend: Angular SPA hosted by nginx which is also used as a reverse proxy to the Spring Boot REST API
- analytics: An instance of the analytics service implementing the anomaly detection.
- analytics-db: A PostgreSQL image serving as the database for the analytics service storing the different IPs visiting the connectors and their status.
- filebeat: A filebeat instance that collects the nginx logs.
- cron: An instance of the cronjobs that are triggered through the analytics service, namely the re-training of the anomaly detection and the prediction.

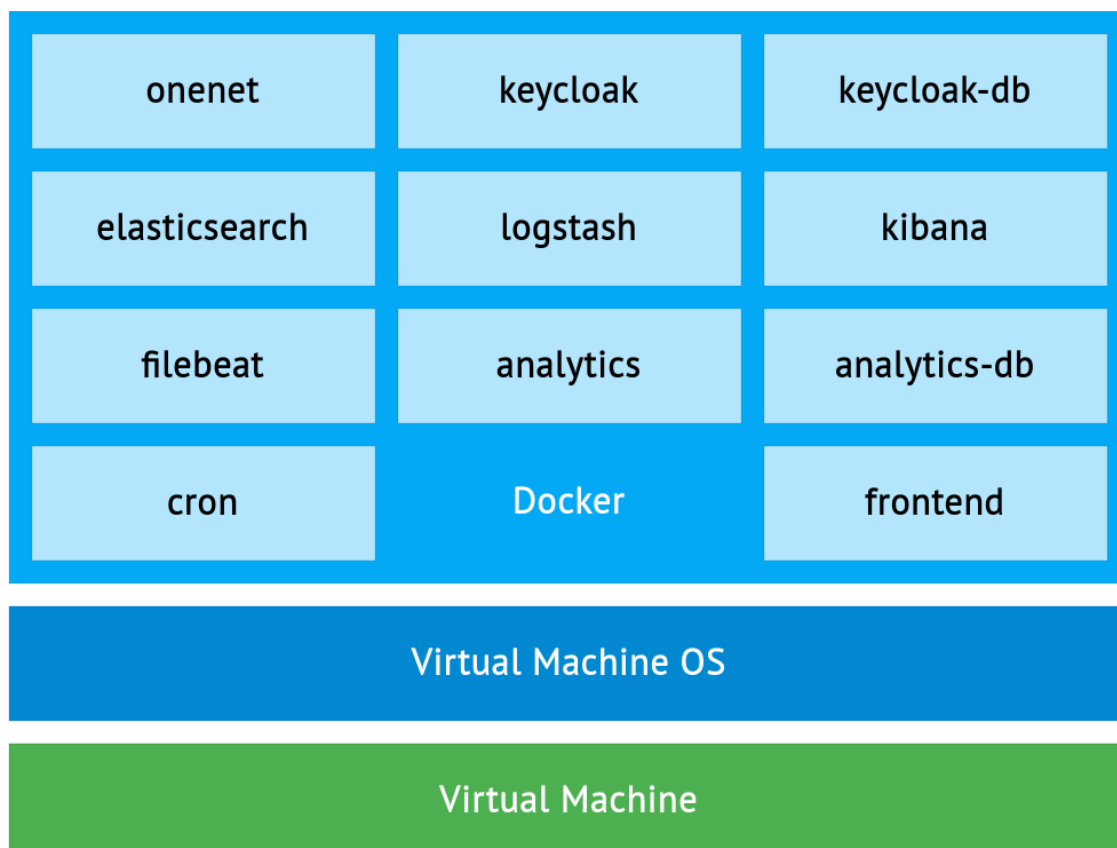


Figure 29: OneNet Monitoring and Analytics Dashboard Docker-based Deployment

8 Conclusions

The work conducted in Task T6.4 and documented in this deliverable provides a complete overview of the implementation of the OneNet Orchestration Workbench and OneNet Monitoring and Analytics Dashboard following the design and requirements defined in WP5 and then refined in WP6.

This implementation was conducted in two different stages: the first release of both the components was released in January 2023 with basic functionalities; the fully functional final release was released in July 2023.

The interconnection of the OneNet Orchestration Workbench and OneNet Monitoring and Analytics Dashboard with the OneNet Decentralized Middleware and OneNet Connector was fundamental for facilitating the access to the overall system and the interaction with the OneNet Data Space.

The two components described in this document bring extra features to enhance the OneNet offering concerning data and service orchestration, testing and evaluation, and monitoring of data exchange. Importantly, they do so without compromising the advantages provided by the OneNet decentralized approach.

These two components may be the elements that differentiate OneNet from any other data-space-based systems. In fact, they give the possibility to all OneNet participants, not only to use their data to test services in controlled environments before integrating them into their systems, but to use any data source usable in OneNet with any type of data-based service, to have a real test environment as similar as possible to the production one. Furthermore, the additional functionalities of the Monitoring and Analytics Dashboard, including the cybersecurity ones described in D6.6, facilitate any monitoring and visualization activity, favouring any decision-making action by the OneNet Participant.

From this point of view, the activity conducted with two of the companies who participated in the open call process (in collaboration with WP12) was very important. This activity gave us their point of view as a service provider and gave us the opportunity to evaluate and prepare the integration of external services to be deployed into the OneNet Orchestration workbench and made available to all the OneNet Participants.

References

- [1] Marko Jelić, Dea Pujić, Dejan Paunović - A State-of-the-Art Review on Big Data Technologies – [Online]. Available: <https://project-lambda.org/sites/default/files/2019-05/ICIST-2019-paper-4.pdf>.
- [2] OneNet Deliverable 5.4 “AI, Big Data, IoT Enablers and FIWARE-compliant interoperable interfaces for grid services” [Online]. Available: https://onenet-project.eu/wp-content/uploads/2022/12/OneNet_D5.4_v1.0.pdf.
- [3] OneNet Deliverable 5.2 “OneNet Reference Architecture” [Online]. Available: https://onenet-project.eu/wp-content/uploads/2022/12/OneNet_D5.2_v1.0.pdf.
- [4] OneNet Deliverable 5.1 “OneNet Concept and Requirements” [Online]. Available: <https://onenet-project.eu/wp-content/uploads/2022/10/D51-OneNet-Concept-and-Requirements.pdf>.
- [5] OneNet Deliverable 6.5 “OneNet Reference Platform First Release” [Online]. Available: https://onenet-project.eu/wp-content/uploads/2022/10/OneNet_D6.5_final_v1.1.pdf.
- [6] OneNet Deliverable 6.6 “Tools for Legal, Regulatory, Privacy and Cybersecurity Compliance”, <https://onenet-project.eu/public-deliverables/>.
- [7] SCADA - Supervisory Control And Data Acquisition. [Online]. Available: <https://www.techtarget.com/whatis/definition/SCADA-supervisory-control-and-data-acquisition>.
- [8] Linux Foundation Energy (LFE) SOGNO Project. [Online]. Available: <https://lfenergy.org/projects/sogno/>.
- [9] Rancher: Enterprise Kubernetes Management. [Online]. Available: <https://www.rancher.com>
- [10] Kubernetes. Production-Grade Container Orchestration. [Online]. Available: <https://kubernetes.io>.
- [11] Helm. The Package Manager for Kubernetes. [Online]. Available: <https://helm.sh>.
- [12] InfluxDB. [Online]. Available: <https://www.influxdata.com>.
- [13] MongoDB. [Online]. Available: <https://www.mongodb.com/>.
- [14] Nginx: advanced load balancer, web server & reverse proxy. [Online]. Available: <https://www.nginx.com>
- [15] OneNet Deliverable 6.1 “Report on decentralized edge-level middleware for scalable platform agnostic data management and exchange” [Online]. Available: <https://onenet-project.eu/wp-content/uploads/2023/02/D6.1-OneNet-v1.0.pdf>.
- [16] OneNet Deliverable 6.3 “Extended Interoperability and Management with FIWARE”, <https://onenet-project.eu/public-deliverables/>.
- [17] OneNet Deliverable 5.3 “Data and Platform Assets Functional Specs and Data Quality Compliance”, <https://onenet-project.eu/wp-content/uploads/2022/12/OneNet-D5.3-v1.0.pdf>