PLATFORM FOR OPERATION
OF DISTRIBUTION NETWORKS

Platone

PLATform for Operation of distribution NEtworks

D3.2

Report of optimal communication solutions between customer database and market players

The project PLATform for Operation of distribution NEtworks (Platone) receives funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no 864300.

Project name:	Platone
Contractual delivery date:	30.04.2021
Actual delivery date:	30.04.2021
Main responsible:	Ercole De Luca, areti
Work package:	WP3 – Italian Demo
Security:	P = Public
Nature:	R
Version:	V1.0
Total number of pages:	27

Abstract

The main objective of the Platone project and, in particular, of the Italian Demo, is contributing to breaking down barriers to the flexibility market, allowing prosumers and consumers connected to the electricity distribution network to actively participate in the optimized management of the grid. Within a multi-layer System Architecture, an Access Layer as common interface between customers and market players has been designed specifically to give the opportunity to customers to offer flexible services to the grid and to guarantee data sharing and data access to the stakeholders involved, improving observability of flexible resources.

The *Report of optimal communication solutions between customer database and market players (D3.2)* describes the architecture of the Access Layer of the Italian Demo and its three components: 1) the Light Node, as an edge device that communicates with the smart meters, which is installed in the Points of Delivery (PoD) of the customers involved in the trial; 2) the Blockchain Access Layer, which allows the integration of the data coming from the Light Node, adding a level of security, transparency and trustworthiness thanks to the blockchain technology and smart contracts; 3) and the Shared Customer Database (SCD), which stores all the energy data (e.g., measurements, set points, etc.), providing access to the data to all the stakeholders involved in the flexibility mechanisms, implementing data security, data privacy and data access policies mechanisms.

The Access Layer of the Italian Demo is currently under implementation. The first release of the Access Layer will be provided together with the complete Italian Demo Architecture in its first version by May 2021. A supporting report describing the first release of the Italian Demo System Architecture will be made available with the release of D3.3 “Delivering of technology (v1)”.

Keyword list

Shared Customer Database; Access Layer; Light Node; Blockchain; Data Access; Data Sharing; Data Integration; Data Certification; Smart Contracts.

Executive Summary

The energy system is facing important changes. Renewable energy sources are increasing in terms of quantity and power. Furthermore, the current energy transition will result in an increase in electrical energy demand in the short and long term due to the large variability and unpredictability of generation and consumption power flows. The distribution grids will be the most impacted by the related effects. Concerning consumption, the spread of Electric Vehicle (EV) charging infrastructures together with the spread of electrical cooling and heating systems, represents a risk factor for the electricity system, causing non-predictable grid imbalances, grid congestions and voltage issues.

In this scenario, the services that flexible resources can offer to System Operators are an effective means to solve grid issues, guaranteeing safety of the system and continuity of electricity supply. Distributed Energy Resources can provide services both to DSO and the TSO. Therefore, enabling DERs to participate to the market is a priority for an efficient management of the grid.

The main objective of the Platone project and, in particular, of the Italian Demo, is to contribute to breaking down barriers to the flexibility market, allowing prosumers and consumers connected to the electricity distribution network to actively participate in the optimized management of the grid.

The proposed solution at the basis of the Italian Demo consists of a layered set of platforms designed to meet the needs of system operators, aggregators and end-users. In the Italian Demo system architecture, the Access Layer is the common interface between customers and market players that has been designed specifically to give the opportunity to customers to offer flexible services to the grid and to guarantee data sharing and data access to involved stakeholders, improving the observability of flexible resources.

The Access Layer is composed of three main components:

1) The Light-Node (LN) is an edge device installed on the customer’s side, which allows metering data coming from the smart meters installed on Distributed Energy Resources’ (DERs) premises to be gathered.
2) The Blockchain Access Layer (BAL) certifies data coming from the Light-Node device and runs Smart-Contracts.
3) The Shared Customer Database (SCD) collects, organizes and stores all the data/measurements related to the DERs and make them available to the other components of the system and to different stakeholders involved, according to authorization rules specifically defined to fulfil security and privacy requirements.

The Access Layer of the Italian Demo is currently under implementation. The first release of the Access Layer will be provided together with the complete Italian Demo Architecture in its first version by May 2021.

A supporting report describing the first release of the Italian Demo System Architecture will be made available with the release of D3.3 “Delivering of technology (v1)”.

Authors and Reviewers

Main responsible		
Partner	**Name**	**E-mail**
Areti		
	Ercole De Luca	Ercole.DeLuca@areti.it
	Gabriele Fedele	Gabriele.Fedele@areti.it
	Antonio Vito Mantineo	AntonioVito.Mantineo@areti.it
Author(s)/contributor(s)		
Partner	**Name**	**E-mail**
Areti		
	Angela Irlandese	Angela.Irlandese@areti.it
Apio		
	Alessandro Chelli	alessandrochelli@apio.cc
Reviewer(s)		
Partner	**Name**	
BAUM		
	Kilian Karg	
E.DSO		
	Kirsten Glennung	
Approver(s)		
Partner	**Name**	
RWTH		
	Padraic McKeever	

Table of Contents

1 Introduction 6
1.1 Task 3.2 6
1.2 Objectives of the Work Reported in this Deliverable 7
1.3 Outline of the Deliverable 7
1.4 How to Read this Document 7
2 Access Layer Architecture 8
3 Light Node 10
3.1 Architecture 10
3.1.1 MQTT Client 10
3.1.2 Identity Layer 10
3.1.3 EMS Interface 10
3.1.4 Meter Interface 11
3.2 Communication 11
4 Blockchain Access Layer 12
4.1 Architecture 12
4.1.1 UI Layer 12
4.1.2 Service Layer 12
4.1.3 Data Layer 12
4.1.4 MQTT Broker 12
4.1.5 Blockchain Layer 13
4.2 Communication 13
5 Shared Customer Database 14
5.1 Architecture 14
5.1.1 Communication Layer 15
5.1.2 Streaming Layer 16
5.1.3 Persistence Layer 18
5.1.4 Data Reporting 20
5.2 Communication 20
5.2.1 Data treatment and data archiving 21
6 Conclusion 23
7 List of Tables 24
8 List of Figures 25
9 List of References 26
10 List of Abbreviations 27

1 Introduction

The project "PLATform for Operation of distribution Networks – Platone - aims to develop an architecture for testing and implementing a data acquisitions system based on a two-layer approach (an access layer for customers and distribution system operator (DSO) observability layer) that will allow greater stakeholder involvement and will enable an efficient and smart network management. The tools used for this purpose will be based on platforms able to receive data from different sources, such as weather forecasting systems or distributed smart devices spread all over the urban area. These platforms, by talking to each other and exchanging data, will allow collecting and elaborating information useful for DSOs, transmission system operators (TSOs), Market, customers and aggregators. In particular, the DSOs will invest in a standard, open, non-discriminating, economic dispute settlement blockchain-based infrastructure, to give to both the customers and to the aggregator the possibility to more easily become flexibility market players. This solution will allow the DSO to acquire a new role as a market enabler for end users and a smarter observer of the distribution network. By defining this innovative two-layer architecture, Platone aims to remove technical barriers to the achievement of a carbon-free society by 2050 [1], creating the ecosystem for new market mechanisms for a rapid roll out among DSOs and for a large involvement of customers in the active management of grids and in the flexibility markets. The Platone platform will be tested in three European trials (Greece, Germany and Italy) and within the Distributed Energy Management Initiative (DEMI) in Canada. The Platone consortium aims to go for a commercial exploitation of the results after the project is finished. Within the H2020 programme "A single, smart European electricity grid" Platone addresses the topic "Flexibility and retail market options for the distribution grid".

Within this context, the Italian Demo (WP3) of the Platone project is realizing a fully functional system architecture that enables distributed resources connected in medium and low voltage to provide grid services in different flexibility market models which include all the stakeholders (TSO, DSO, aggregators and end-users). The main goals of the Italian Demo are:

- Use of Blockchain technology for an efficient, democratic and non-discriminatory market model for exploitation of local flexibility in the Rome area;
- Improving and promoting access by customers thanks to Blockchain infrastructure and to the presence of the Aggregator;
- Use of local flexibility to solve criticalities which can affect the distribution grid in terms of reliability and security of supply;
- To enable distributed resources to provide flexibility to transmission grid to contribute to guaranteeing that the whole system remains balanced and safe;
- To increase the grid observability for improving the network management.

To address these aims, specific activities have been carried out and are under implementation within WP3. The Italian Demo partners are developing the components of the System Architecture that will be released by the end of May 2021 with the delivery of D3.3 *Delivery of technology (v1)*.

Among the components composing the Italian Demo System Architecture, the SCD is the key-component specifically designed for collecting, storing and organizing all the data related to the DERs to make them available to the other actors and components of the System Architecture. Within the Platone and the Italian Demo solution, the SCD is covered by the Access Layer, which is a common interface between customers and market players that gives the opportunity to the customers to offer flexible services to the grid and to guarantee data sharing and data access to the stakeholders involved, improving observability of flexible resources. The Access Layer allows entry barriers for aggregators to the flexibility market to be lowered and, at the same time, fosters the participation of the end users in the energy market with an active role, thanks to the blockchain mechanisms at the basis of the solution proposed, that will guarantee certified transactions.

1.1 Task 3.2

Task 3.2 "*Development of a standard Blockchain based infrastructure, implementing a Common Access Interface between all the market players*" is one of the project tasks composing WP3 – Italian Demo. Task 3.2 is coordinated by Areti in cooperation with Apio, Siemens, Engineering and Acea Energia. It includes the following sub-activities:

- *HW/SW development of Blockchain technologies to include customers and Aggregator in network management*, led by Apio;

- *Definition of communication protocols, identification of the communication channel and development of the apparatus for meter's data exchange*, led by Apio;
- *Development of Shared Customer Database platform* led by Areti.

The sub-task *Development of Shared Customer Database platform* addresses the development and implementation of a key component of the Common Access Interface within a blockchain based infrastructure. Specifically, the Shared Customer Database (SCD) component aims to collect, organize and store all the data related to the Distributed Energy Resources to make them available to the other components of the System Architecture of the Italian Demo.

1.2 Objectives of the Work Reported in this Deliverable

The *D3.2 Report of optimal communication solutions between customer database and market players* aims to describe the Access Layer architecture of the Italian Demo, its specific components and the related communication mechanisms that have been designed. The work reported in this Deliverable is a building block forming the basis of the release of the complete Italian Demo System Architecture (D3.3).

1.3 Outline of the Deliverable

This first introductory chapter describes the Platone reference context and the specific project task linked to deliverable D3.2, also providing indications about the objectives and characteristics of the document. Chapter 2 "Access Layer Architecture" provides an overview on the Common Access Interface architecture characteristics and its components within the Italian Demo. Chapter 3 "Light Node" describes the layers and interfaces composing the Light Node and the communication protocols that will guarantee the interactions between the device and the other components of the system architecture. Chapter 4 "Blockchain Access Layer" focuses on the architectural sub-layers composing the Blockchain Access Layer and related communication protocols. Chapter 5 "Shared Customer Database" describes the SCD architecture and its communication protocols. Chapter 6 concludes the document by outlining results achieved as of today and next steps.

1.4 How to Read this Document

This document is self-contained, and it can be read independently from other Platone's deliverables.

Interesting links for deepening the progress and evolution of the selected approach throughout the project from its beginning to now, can be made with the following Platone deliverables:

- D1.1 *General functional requirements and specifications of joint activities in the demonstrators* [2] released by E.DSO in August 2020 as public detailed work that provides an overview of the three Platone Demo sites. Within D1.1, through an empirical analysis of context of the demos provides an overview of the use cases and some technical definition that are useful for understanding the different components composing the System Architecture;
- D3.1 *Internal operational plan and WP3 roadmap*, released by Areti in November 2019 as confidential detailed work plan and roadmap for kick starting WP3. Within D3.1, proposals of Blockchain-based System Architecture have been reported, indicating an initial time frame for technology release;
- D3.3 *Delivering of technology (v1)* to be released by the end of May 2021 as demonstrator supported by a report elaborated for the release of the first version of the System Architecture, describing all the components of the system architecture developed by the Italian Demo and their interactions;

2 Access Layer Architecture

The Access Layer allows the management and the sharing of Distributed Energy Resources' (DERs) data in a secured and certified way, enabling customers to provide flexible services to the grid and the System Operators to improve the grid observability.

This common interface includes three main components, which will be described in detail in the following chapters:

- **Light-Node (LN) device**, that gathers metering data coming from the smart meters installed at DERs' premises, receives the flexibility activation commands and makes them available to the customer Energy Management System (EMS) (e.g. Storage System, Smart Homes Devices, etc.);
- **Blockchain Access Layer (BAL)**, that certifies data coming from Light-Node device and runs Smart-Contracts that link Light Node Public Keys to Unique Meter IDs. Others Smart Contracts could be developed and deployed on the Access Layer;
- **Shared Customer Database (SCD)**, that collects, organizes and stores all the data related to the DERs and make them available to the other components of the system and to different stakeholders involved, according to authorization rules specifically defined to fulfil security and privacy requirements.

The role of the Access Layer is to authenticate and make immutable the Energy Data, through the following process:

- The Light Node allows data authentication. In fact, data are signed before being sent to the Blockchain Access Layer, through an asymmetric key signature mechanism;
- Before timestamping the data and forwarding them to the SCD, the Blockchain Access Layer verifies the signatures by trusting the data recorded in the Blockchain. Specifically, it verifies that the data have been signed by the account whose public key corresponds to a specific Point of Delivery (PoD) by using a Smart Contract that can only be modified on behalf of the DSO;
- The Blockchain Access Layer microservice aggregates and performs a transaction for a set of measures communicated in the same time frame;
- The microservice sends the data containing the time stamp and the transaction carried out to the Blockchain Access Layer to SCD.

This way, the Platone platforms and actors can verify (timestamping and Light Node identity) energy and activation data without storing them in the Blockchain.

The Blockchain, which is a permissioned blockchain based on Ethereum, is not used as a database because is not scalable, for this reason the Blockchain Access Layer is used as a verification tool of the data stored in the SCD.

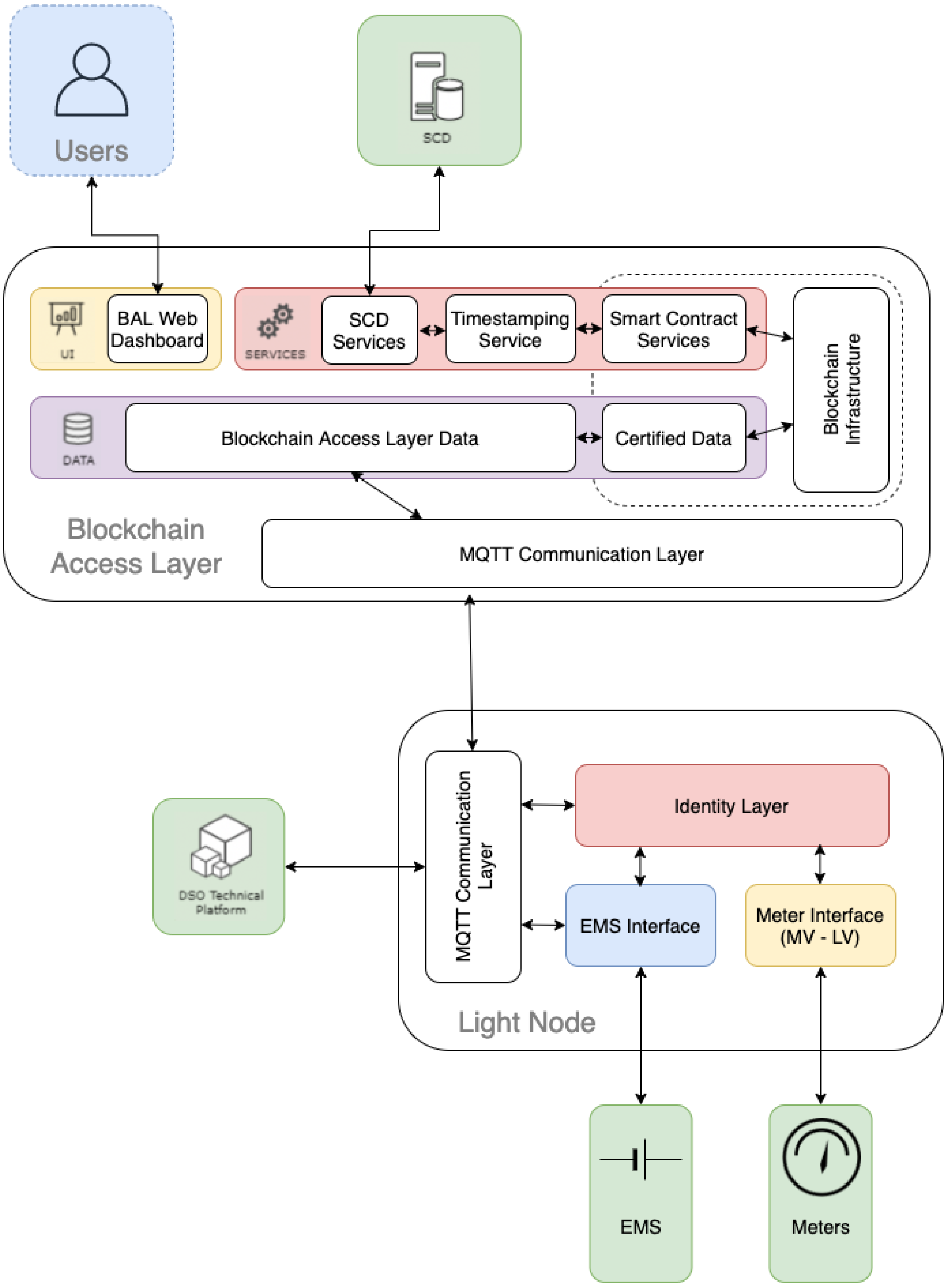

Figure 1: Access Layer Architecture

3 Light Node

The Light-Node (LN) is an edge device installed on the customer's side, which allows gathering of metering data coming from the smart meters installed on Distributed Energy Resources' (DERs) premises. It receives the flexibility activation commands and makes them available to the customer Energy Management System (EMS) (e.g. Storage System, Smart Homes Devices, etc.).

The following sub-chapters describe the architecture of the Light Node apparatus and the related communication protocols.

3.1 Architecture

The Light Node consists of two-layers and two interfaces:

- ***MQTT Client**,* which provides connection to the DSO Technical Platform and the Blockchain Access Layer;
- ***Identity Layer**,* which provides a digital identity (public and a private keys) for the Light Node to sign and send the data packet to Blockchain Access Layer;
- ***EMS Interface***, which interacts with several Energy Management Systems (e.g. Battery Management Systems, Smart Homes Devices, Power Plant Management Systems);
- ***Meter Interface***, which interacts with several Meters technologies (e.g. Low Voltage and Medium Voltage Meters).

3.1.1 MQTT Client

The MQTT Client provides connection to the DSO Technical Platform (DSOTP) and the Blockchain Access Layer. MQTT Client is used to receive Set Point from DSOTP and to send data to the Blockchain Access Layer. It is implemented through MQTT.js, an open source javascript library [3].

3.1.2 Identity Layer

The Identity Layer is a Node.js [4] microservices that signs data packets through the identity keys of the Light Node. The Identity Layer uses also Web3.js [5], an open source javascript library that implements several wallet functions. For instance, it allows to add, remove, crypt decrypt and load accounts in the wallet, as well as create new wallets. All these functionalities include the cryptographic functions required by the blockchain platform.

3.1.3 EMS Interface

The EMS Interface sends the *Set-Point* to the Customer's EMS. The communication protocol used depends on the EMS installed at customer's side:

- Industrial application: In this environment, the communication protocol is Modbus RTU or Modbus TCP;
- Energy Community application: In this environment, the communication protocol is REST API or MQTT;
- Electric Vehicle Charger application: The communication protocol is REST API or Modbus RTU or Modbus TCP.

The EMS Interface, through MQTT Client, sends the Activation Dataset to the Blockchain Access Layer signed by the Identity Layer. The EMS Interface is developed using Node.js and open-source library.

3.1.4 Meter Interface

The Light Node communicates with the Smart Meter in the three different ways explained below, that can be activated in the configuration phase of the device.

The related communication channel depends on the electric meter capabilities:

- ***Smart Meter (for Medium Voltage customers):*** integration through an Industrial Protocol (Modbus TCP or Modbus RTU);
- ***Smart Meter 2G (for Low Voltage customers):*** integration through a powerline interface (PLC-C as known as Chain 2);
- ***Smart Meter 1G (for Low Voltage customers):*** integration through additional meters (smart devices capable to retrieve the necessary electrical parameters).

The Meter Interface reads metering data, and then, through the MQTT Client, sends the Meter Data signed by the Identity Layer to the Blockchain Access Layer. The Meter Interface is developed using Node.js and open-source library.

3.2 Communication

The Light Node includes several communication protocols to communicate with external components and Platone platforms which can be understood from the Table 1.

In particular, the Light Node is able to communicate with the DSO Technical Platform, the Blockchain Access Layer and, through the Blockchain Access Layer, with the SCD, as well as with the Meters installed at the PoD level and customer devices such EMS etc. (see Figure 1)

Table 1: Light Node Communication

Components A	Components B	Communication Protocols
Light Node	EMS	REST-API (Smart Objects) MQTTs (Smart Objects) Z-Wave (Smart Home) Modbus TCP (UPM) Modbus RTU (Storage and PV)
Light Node	Meters	PLC-C (Chain 2 – Smart Meters) Modbus TCP (Meters MV) MQTTs (Cloud to Light Node Integration)
Light Node	DSO Technical Platform	MQTTs
Light Node	Blockchain Access Layer	MQTTs

4 Blockchain Access Layer

The Blockchain Access Layer (BAL) is an architectural layer that adds a further level of security and trustworthiness to the framework. It certifies data coming from the Light-Node device and runs Smart-Contracts.

The following sub-chapters describe the architecture of the Blockchain Access Layer and the related communication protocols.

4.1 Architecture

The Blockchain Access Layer consists of a five-layer architecture:

- **UI Layer** includes an easy-to-use web dashboard that allows all the stakeholders to explore Blockchain Transactions and to DSOs to monitor the Light Node's status;
- **Services Layer** provides the business logic, including the SCD communication, Timestamping services and Smart Contract Services;
- **Data Layer** provides the management of the Blockchain Access Layer and Light Node data and the registration of Timestamping Information of Light Node (Metering and Set-Point).
- **MQTT Broker** provides bi-directional communication channel with the connected Light Node;
- **Blockchain Layer** provides the DLT features, in Blockchain Infrastructure transactions are arranged in blocks and placed in a P2P network.

4.1.1 UI Layer

The User Interface (UI) Layer includes an easy-to-use web dashboard that allows all the stakeholders to explore Blockchain Transactions and to DSOs to monitor the Light Node's status. The UI Layer will be provided as a Web Application developed with ReactJS [6].

4.1.2 Service Layer

The Service Layer provides several microservices to enable SCD data sharing, Timestamping functionalities and Smart Contract Integrations. More in detail, the SCD data sharing implements an Apache Kafka [7], an open-source distributed event streaming platform. Timestamping functionalities are Node.js [4] microservices that aggregate data through merkle-tree to optimize timestamping functionalities. Smart Contract services are Node.js microservices that interact with Blockchain interface through Web3.js an open-source javascript library that connects to Ethereum JSON-RPC.

4.1.3 Data Layer

The Blockchain Access Layer provides a Data Layer that is a Database Abstraction Layer. It provides an API (Application Programming Interface) which unifies the communication between the Light Node and the NoSQL database. More in detail, the API is implemented via the REST APIs using open source framework Express.js [8] and Express Gateway [9], the NoSQL database is an instance of open source database MongoDB [10]. The API gateway provides resources for Authentication, Light Node functionalities, Logging, Caching and Security.

4.1.4 MQTT Broker

The Blockchain Access Layer architecture includes a MQTT Broker, which is a server that receives all messages from the clients and then forwards the messages to the appropriate destination clients. The MQTT Broker provides connection to the associated Light Node.

The MQTT broker is developed using Aedes [11], an open source Node.js MQTT broker, which can be used as a Standalone Service or embedded in another Node.js application.

Specifically, the MQTT Broker and the integration services with the Light Node ensure:

- Light Node connection
- Anomalies Detection
- Meter Data Reception
- Set-Point Reception

4.1.5 Blockchain Layer

The blockchain service layer is based on a blockchain infrastructure that includes Ethereum Blockchain nodes and smart contracts services. The Blockchain layer is developed using Hyperledger Besu [12] Nodes: an open-source Ethereum Nodes designed to be enterprise-friendly for both public and private permissioned network use cases. Hyperledger Besu includes several consensus algorithms including PoW, and PoA (IBFT, IBFT 2.0, Etherhash, and Clique) [13]. Its comprehensive permission schemes are designed specifically for being used in a consortium environment.

In particular, the smart contracts ensure that all the data (metering and set point) included in the Blockchain Access Layer and Light Nodes are certified thanks to the blockchain infrastructure.

All these characteristics enable a blockchain-driven energy marketplace that:

- Ensures energy transactions certification;
- Tracks and controls the registration and validation of energy data.

4.2 Communication

The Blockchain Access Layer architecture includes several communication protocols to communicate with the solution's components. A specific architectural component dedicated to communication mechanisms, provides a greater flexibility to the Blockchain Access Layer, which can cover different solutions and integrate different external systems.

In particular, the communication layer offers an interface for the integration of the DSOTP and the SCD. Furthermore, this layer allows for the internal layers of the stream (i.e. Light Node, EMS and Meters) to communicate between each other.

The following table summarizes the communication protocols used among the components and actors involved in the Blockchain Access Layer:

Table 2: Blockchain Access Layer Communication

Components A	Components B	Communication Protocols
Blockchain Access Layer	Light Node	MQTTs
Blockchain Access Layer	Shared Customer Database	KAFKA REST-API
Blockchain Access Layer	Users	Web Interface

5 Shared Customer Database

The Shared Customer Database (SCD) collects of data and information and makes them available to all the actors involved in the Italian Demo ecosystem. SCD collects, organizes and stores all the data/measurements related to the DERs and make them available to the other components of the system and to different stakeholders involved, according to authorization rules specifically defined to fulfil security and privacy requirements. The communication modalities between the Shared Customer Database and the other components and platforms are characterized by a huge amount of high frequency data exchange.

The following sub-chapters describe the architecture of the Shared Customer Database and the related communication protocols, by focusing on data treatment and data archiving.

5.1 Architecture

The exchanged information flows concern the personal and technical data and the measurements associated with the flexibility market. Each resource is uniquely identified by a PoD code, (i.e. the connection point between the customer on field assets and the distribution network).

The communication modalities between the SCD and the other components and platforms are characterized by a huge amount of high frequency data exchange to be able to manage, hypothetically, a high number of PoDs.

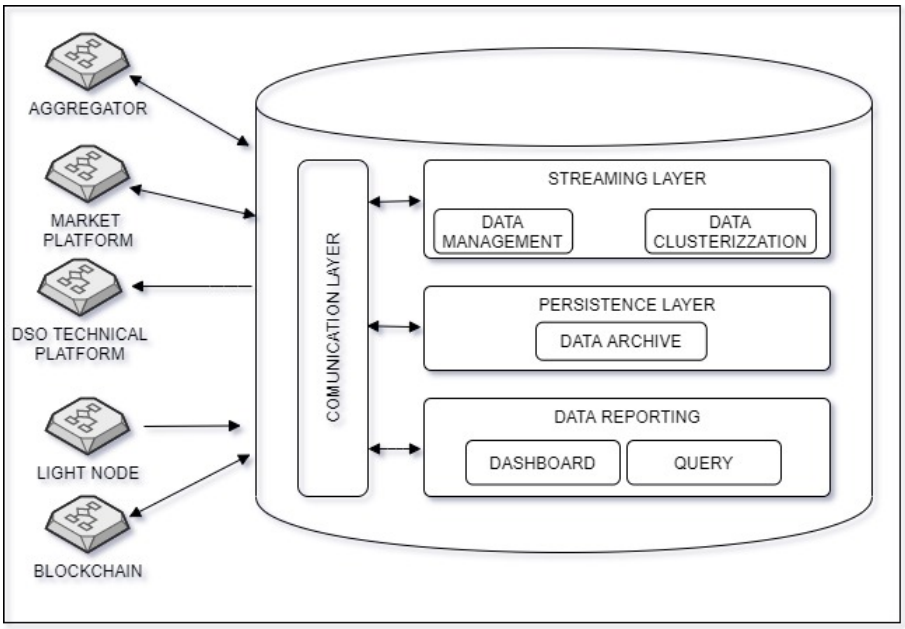

Figure 2: Shared Customer Database Architecture

The SCD consists of a four-layer architecture visualized in Figure 2 above:

- **Communication layer**, the layer used by external actors to read and write within the SCD;
- **Streaming layer** for the processing of data from the field. In the trial use cases, this will be possible thanks to a tool used for transferring the processed data to the persistence layer for archiving purposes. This tool is named Ingestion Component;
- **Persistence layer** for data storage. It is characterized by a non-relational database and by a database with strong data search and aggregation skills;

- **Data Reporting** for the exposure of data processed in a ductile way to the ones who wish to monitor the trend of data flexibility.

5.1.1 Communication Layer

Regarding the communication layer, custom components that expose Web REST API using the Open API 3.0 [14] standard for interface definition have been released. Those components have been developed using a “reactive programme” that allows to use all the computational resources available in non-blocking mode.

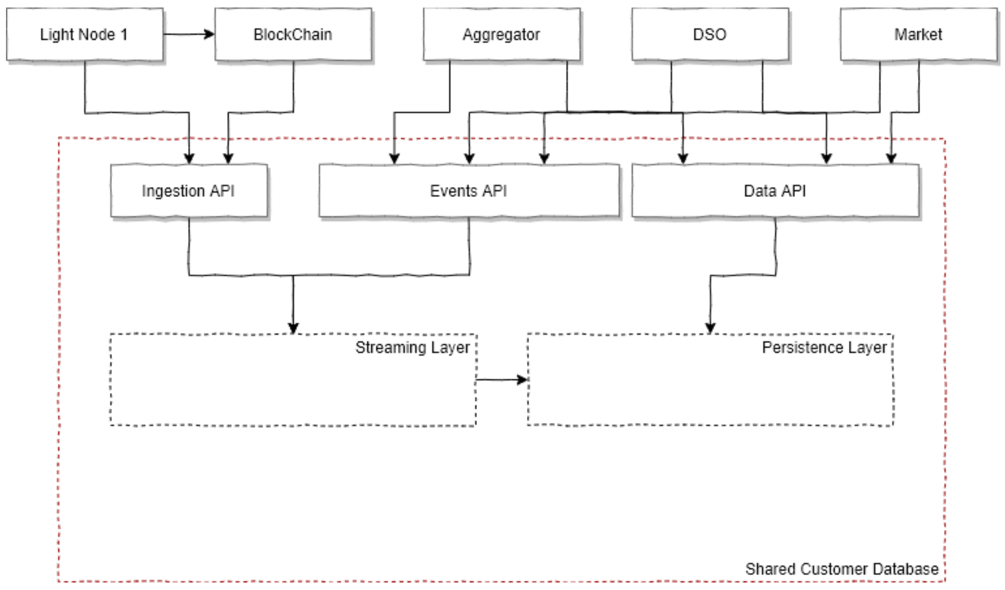

Figure 3: Communication Layer

The following API subsets are foreseen:

1. **Ingestion API**
   Ingestion API, used for data writing, is implemented by developing custom software components in a microservice logic. The technology used to release Ingestion APIs is the Node.js [4] and the Express.js [8] web framework. These components will display HTTP / REST endpoint and will be configured to scale horizontally;
2. **Data API**
   Data API, used for reading data stored in the SCD, as well as Ingestion APIs, Data APIs are implemented by developing custom software components in a microservice logic. The technology used to release Data APIs is the Node.js [4] and the Express.js [8] web framework. These components will display HTTP / REST endpoint and will be configured to scale horizontally
3. **Events API**
   Events API, used for reading/writing real-time data coming from the field, are organized through the Kafka components described below:
   a. **Kafka API**

      Communication between clients and the Kafka [7] cluster takes place through a binary protocol over Transmission control protocol (TCP). The Open Source project Kafka includes officially supported Java APIs which can be used to build client applications. Since Kafka is a well-known and widely used component there are a number of established and supported APIs for other programming languages.

      The five main Event APIs used are the following:

i. *Producer API*: it allows an application to publish a record flow on one or more topics ("topic" is the technical name indicating the category of record flow data stored in Kafka);
ii. *Consumer API*: it allows an application to subscribe to one or more topics and to elaborate the record flows produced;
iii. *Streams API*: it allows an application to act as a streaming processor, consuming an input flow from one or more topics and producing an output flow on one or more topics;
iv. *Connector* API: it allows connection Kafka topics to applications or systems outside the cluster;
v. *Admin API*: it allows the management of topics, broker and other Kafka objects

b. Kafka Rest Proxy

Rest Proxy is a module that provides an http / REST interface to the Kafka cluster, thus giving the possibility to produce and consume messages without using the native Kafka protocol and its client. It also exposes APIs that allow to verify the cluster status and perform administrative actions. Within the Platone Italian Demo architecture, it represents an additional possibility of communication with the SCD for the other components of the system, suitable in particular for sending data to the SCD. Here follows the list of the main components of the Kafka Rest Proxy:

i. Metadata component: this component permits the reading of most of the metadata clusters (topic, broker, partitions and configurations). These can be read through GET requests for matching URLs;
ii. Component for Producers' requests: thanks to this component the API accepts producers' requests for specific topics or partitions and forwards them in a consistent way with the request; it is also possible to configure producers' settings in a global way, for example it is possible to enable the compression type option in order to reduce storage and network overhead;
iii. Component for Consumers: it is the component that enables groups of consumers to read records in the topics. The offset commit is automatic or explicitly requested by the consumer;
iv. Load Balancing component: it is the component that relates the data load balancing, by supporting multiple running instances to balance the load.

The REST API supports HTTP and HTTPS protocols.

c. API Gateway

All the APIs intended for the external stakeholders are exposed via an API Gateway, which is the component that deals with various aspects related to REST API management: monitoring, logging, security, access statistics, access limits.

Regarding the authorization protocol, OAuth 2.0 [16] is used, and it is recognized as a de facto standard for the protection of REST API.

The product chosen for this component is Kong Gateway [17]. It was chosen since that it is an open source API Gateway based on Nginx and that is extendable through plugins if extra features and services in addition to core ones are needed.

5.1.2 Streaming Layer

Regarding the streaming component, the Open Source solutions Apache Kafka, Apache Zookeeper and Kafka Connect have been selected, as represented in Figure 4. Apache Kafka is an open source product, representing one of the main components for data streaming architectures. Apache Zookeeper is an open source component instrumental to the management of a Kafka cluster. Kafka Connect is a component of Kafka ecosystem, used to connect Kafka to external components.

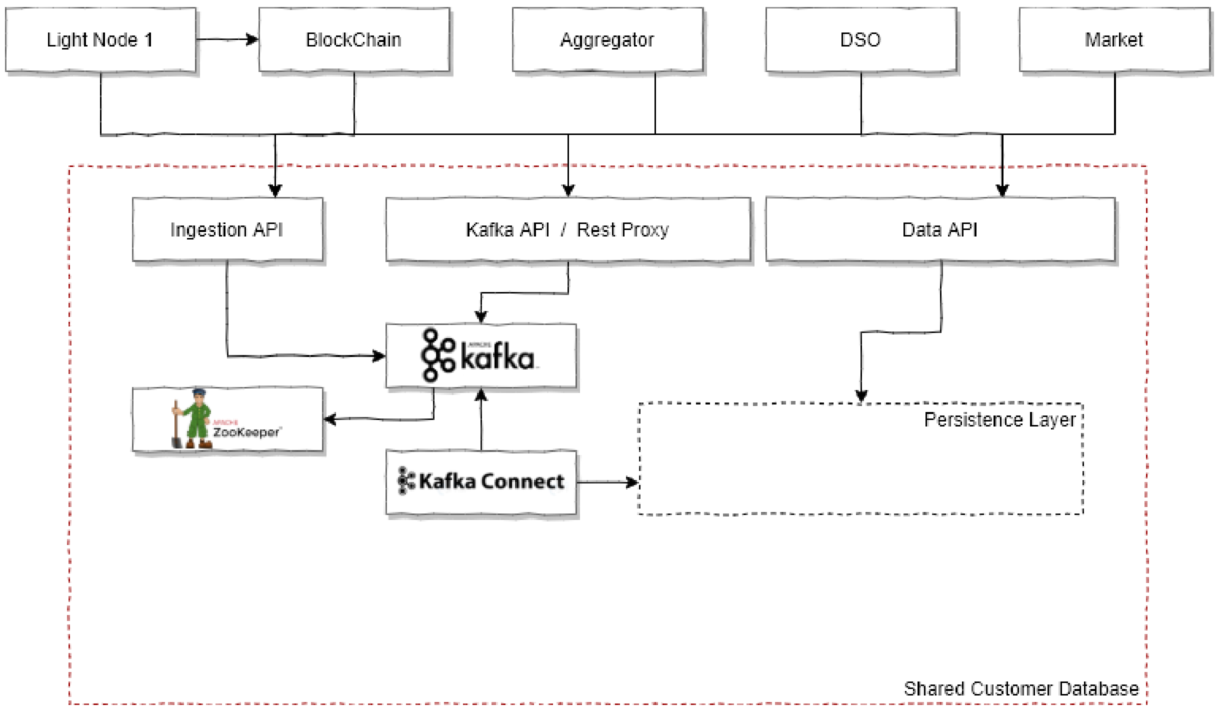

Figure 4: Streaming Layer

1. **Apache Kafka [7]**

   Kafka is an open-source stream processing software developed by the Apache Software foundation [15] with the aim of providing a platform that is able to manage and process data in real time. In Kafka, two main actors are identified:

   a. *Producers*: are those who produce input data streams for Kafka;
   b. *Consumers*: are those who, by making a subscription, can consume the flows of records and, possibly, process them.

   Kafka is usually run in clusters. It stores record flow data in categories that are called "topics". Each record consists of a key, a value and its timestamp. A Kafka topic is always a multi-subscriber type, meaning that it can have one or more consumer. For each topic, Kafka keeps a partitioned log, as described in the following figure:

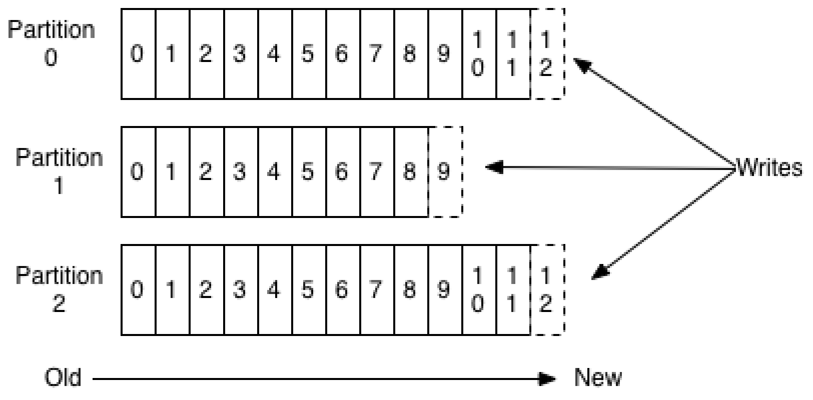

Figure 5: Log Topic Kafka

Each partition is an ordered and unchangeable sequence of records, each one having its own sequential ID (identification) called “offset”, which uniquely identifies it within the partition. A Kafka cluster permanently stores all the published records, regardless of whether they were consumed or not, throughout a configurable retention period without affecting the cluster’s performance. The offset is used to read data by consumers who can manage it independently. The consumers can advance in a linear way, allowing to read the most recent values or they can restore a previous value to rework the data of the past. This mode of reading means that there is no impact on the Kafka cluster or other consumers present in the cluster, as each consumer is responsible for their readings. Partitions are distributed and replicated, in a configurable way, between the nodes of a Kafka cluster in order to overcome any anomalies. Each partition has a node that acts as a “leader” and other nodes that act as a “follower”: The leaders handle all requests to read and write partitions while the followers passively replicate the leader. In the event that a leader fails to handle a request, one of the followers will automatically become leader, this is because each node acts as a leader for some partitions and as follower for others, so that the balance within the cluster is guaranteed. The producers publish the data on one or more topics chosen by them, and they are responsible for choosing the record to be assigned to a particular partition within the topic. This can be done with a round robin, a balancing algorithm used to balance the load by rotating on the available servers, or by according to a semantic partition function, for example based on a key in the record.

2. **Apache Zookeeper [18]**

   The management of a Kafka cluster is carried out by Zookeeper, a high-level software developed by Apache. Zookeeper monitors both cluster node status and topics or partitions. The main features of Zookeeper are:

   a. *Controller election*: responsible for ensuring the follower-leader relationship in all partitions. For example, if a leading node fails for some reason, it is the controller’s task to alert the remaining followers to act as the partition leader in order to secure the service.

   b. *Configuration topic*: manages the configuration of all the topics, including the list of existing topics, the number of partitions for each topic, the location of the replicas.

3. **Kafka Connect [19]**

   Kafka Connect is a tool for scalability and reliably streaming data between Apache Kafka and other data systems. It makes it simple to quickly define connectors that move large data sets into and out of Kafka. Kafka Connect can ingest entire databases or collect metrics from all your application servers into Kafka topics, making the data available for stream processing with low latency. An export connector can deliver data from Kafka topics into secondary indexes like Elasticsearch or into batch systems such as Hadoop for offline analysis

5.1.3 Persistence Layer

Persistence layer hosts the NoSQL MongoDB and Elasticsearch databases:

- *Elasticsearch* [20]: NoSQL database that will store measurements data from the Light Node and Blockchain Platform;
- *MongoDB* [10]: NoSQL database in charge of recording the data relating to the PoD and the customers.

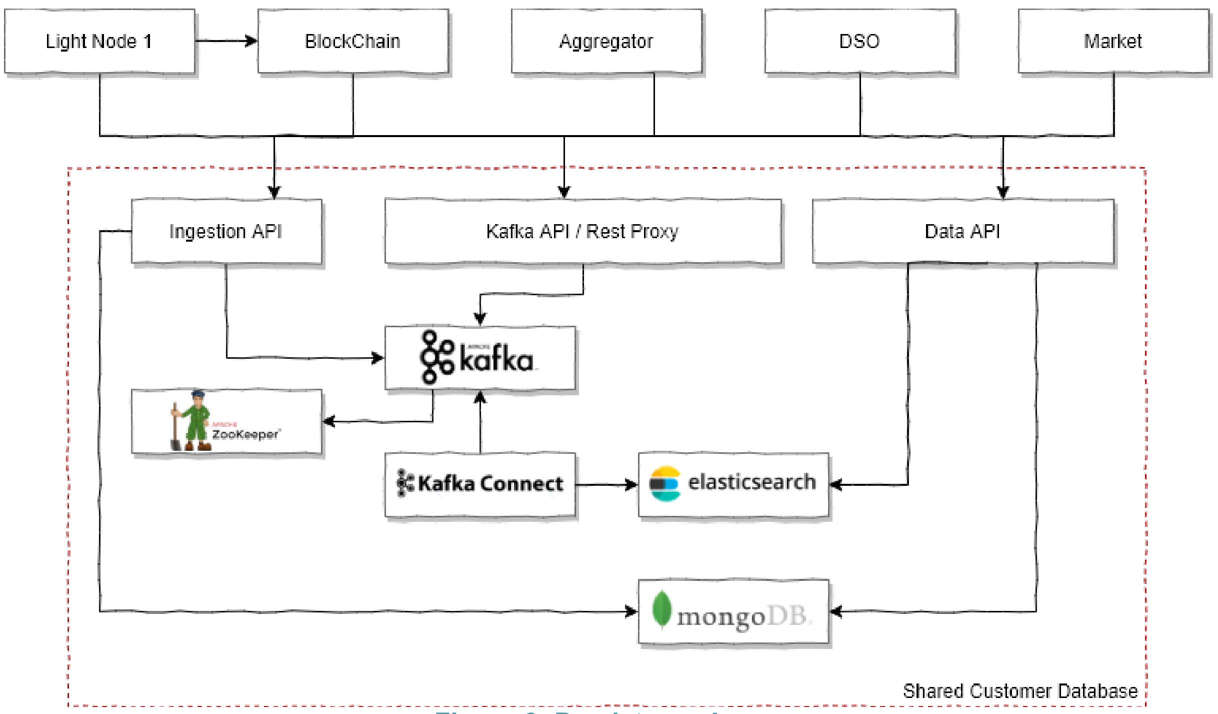

Figure 6: Persistence Layer

Elasticsearch

This Open source NoSQL database was initially created as a high-performance search engine based on Lucene [21]. Its main feature is the concept of Inverted Index, an index consisting of a list of all the keywords that appear in any document and, for each word, a list of the documents in which it appears. In this way the search for the data is almost instantaneous. Elasticsearch is defined as a platform in "near real time" because the time elapsed from when a data is indexed until it becomes searchable is about 1 second. An index is represented by several shards (bricks) which can have one or more replicas. Each shard is a Lucene index. Elasticsearch ensures that no node should have on its storage a shard and its replica. Another key feature of Elasticsearch is resilience, that is the ability to adapt to change and to reconfigure itself autonomously. For this reason, the platform is scalable horizontally.

In the Elasticsearch, clusters present different roles:

- *Master Node*: these are the nodes assigned to the cluster management. They contain the cluster metadata and coordinate operations between the remaining nodes;
- *Data Node*: nodes responsible for writing, reading and aggregating data operations;
- *Ingest Node*: the nodes that perform transformation on input data addressed to data nodes.

MongoDB

MongoDB is an open source NoSQL and no relational document-oriented database. It differs from classic relational databases which are based on diagram and tables because it treats its own data in JSON format, called BSON that means JSON Binary. BSON documents consist of ordered lists of elements where each element is composed of the value-field pair. The values of the fields can include other documents, arrays or array of documents. MongoDB stores the documents in collections, elements that can be considered as similar to the tables of the classic relational database.

Here are some of the main features of MongoDB:

- *Query Language*: it is an ad hoc query language in JSON format that allows to filter and sort according to any field, regardless of how it can be nested within the document;
- *Indexing*: any field can be indexed. Secondary indices, unique indices, scattered indices, geospatial indices and full-text indices are also available;

- *High reliability*: This is guaranteed by replica sets that represent two or more copies of the data. Each replica can have the role of primary replica, when it allows writing and reading operations to be carried out on the data, and secondary replica, when it has a copy of the data. In the event of failure of the primary replica, the replicated set initiates an election process to identify which of the secondary replicas may become primary;
- *Sharding*: is an operation which allows horizontal scalability. By defining a sharding key it is possible to determine how the data of a collection will be distributed on the various nodes. MongoDB also allows a balancing mechanism that enables moving the data from a more charged shard to a less charged one in order to have a balance within the cluster.

Within the Platone Italian Demo, MongoDB will be used as a storage of the records of the PoD and the customers. The description of the communication modalities between Aggregator Platform, DSO Technical Platform and MongoDB will be described in detail in D3.3 *Delivering of technology* (to be delivered in May 2021).

5.1.4 Data Reporting

Data reporting is a component of the SCD that manages the data, exposing them differently, depending on the informative purpose. It is possible to present the information according to:

- Type of service;
- Geographical distribution;
- Power or set point for flexibility participation;
- Aggregator entity.

The tool used to create reference dashboards is Grafana, an open source cross-platform software. Grafana allows to attach a multitude of time-series data source such as Elasticsearch, Prometheus and Graphite. Grafana allows the creation of dashboards using graphs, visuals and gauges. In addition, it is possible to implement an alert functionality based on predefined and ad hoc rules. For example, it is possible to configure an alert when a certain threshold is exceeded or when a particular event occurs. Alert notifications can be of various types, such as email, Slack channels or Webhook. In Grafana, it is also possible to manage the roles and access of users through the creation of groups/users. In fact, it is possible to create groups and users and assign them a predefined role. The available roles are the following:

- Admin: Instance administrator, who can create, edit and read all the dashboards and can create groups and enable users to sign-in;
- Editor: A user acting as an editor and can read and/or edit the dashboard to which he/she is enabled. The editor cannot enable any other user;
- Reader: A user with reader role, who can only view the dashboards to which he/she is enabled, without any possibility of modification.

5.2 Communication

The SCD architecture includes several communication protocols to communicate with the other Platone components and actors.

In particular, the Communication Layer offers an interface for the integration of Aggregator Platform, DSOTP, Market Platform, TSO Simulator and Blockchain Access Layer (including interface to Light-Nodes).

The following table summarizes the communication protocols used between the SCD and the other components:

Table 3: Shared Customer Database Communication

Components A	Components B	Communication Protocols
Shared Customer Database	Aggregator Platform	REST API Apache Kafka
Shared Customer Database	DSO Technical Platform	REST-API
Shared Customer Database	Market Platform	REST-API Apache Kafka
Shared Customer Database	TSO Simulator	REST-API
Shared Customer Database	Blockchain Access Layer	REST API & Apache Kafka

5.2.1 Data treatment and data archiving

The data present in the SCD will be made available to be consulted by other systems for a period of about 2 months from their entry into the system. At a later stage, they can be "frozen" as probably no longer subject to query. In the event that the "frozen" data should be made available for any ad hoc query, they can be restored in the cluster through Restore API, which allows performing operations on the data, such as changing data or settings.

Elasticsearch exposes REST APIs in JSON format on the HTTP protocol, which allows access and management. One of the features of the platform is the ability to create and apply policies for managing the life cycle of indices (Index Lifecycle Management – ILM). Policies can activate different actions:

- Rollover: when an existing index reaches a certain size, number of documents or age, it redirects the index alias – key index used for a prompt data recovery - to write to a new index;
- Shrink: it reduces the number of primary shads in an index;
- Force merge: it merges to reduce the number of segments in each shard and free up space used by deleted document;
- Freeze: it creates a read-only index, minimizing the footprint in memory;
- Delete: it permanently removes an index, deleting all its data and metadata.

The Index Lifecycle Management (ILM) generally defines four phases for managing the life cycle of an index:

- Hot: the index is actively updated and interrogated;
- Warm: the index is no longer updated, but it is still available at the interrogation stage;
- Cold: the index is not updated anymore and is rarely interrogated, for this reason it is subject to slower answers in the query phase;
- Delete: the index is no longer needed and can be deleted without any problems.

The four phases described above will be adapted and configured according to the needs and requirements identified for the final solution, as well as the policies.

Certified measurements from the Blockchain Platform will be recorded within the cluster in JSON format. Given the nature of Elasticsearch, the data will be organized into different indexes; each one will be divided into several shards. Each shard will have its own replica that will in most cases reside in the same node where the primary shard is present. In this way, the required measure will be also available in case of any anomalies on a particular node of the Elasticsearch cluster. Each index will also have a reference to the PoD identifier and timestamp of the measurement, in order to reconstruct the complete sequence to a specific time interval.

Based on the considerations above, input measurements will be recorded on a daily index, for example "Measurements-yyyy-MM-dd" where "yyyy-MM-dd" are the references of the year, month and day of the creation of the index. The following day begins the rollover procedure that allows to create a new index for the current day and changes the status of the index related to the previous day in “Warm”. Since the measurements will be no longer updated, the incremental snapshot of the previous day index on bucket AWS S3 will also begin at this stage. As final solution, it was decided to adopt a retention of 2 months, so the measures will be available to be consulted by other systems for this period of time, and then they can be “frozen” (or archived) as no longer subject to query.

6 Conclusion

The work realized so far for the definition of the Access Layer of the Italian Demo system architecture allows to make an important step in the development and implementation process of the Italian Demo. Indeed, the Access Layer described in this document is the first prototype of the common interface that connects customers with all the market players and that will be delivered by the end of May 2021 together with the other components of the Italian Demo System Architecture (D3.3). The System Architecture will be then tested in field by involving end-users of selected target areas of the city of Rome.

The Access Layer architecture, its components and communications mechanisms were designed in cooperation between Areti and Apio, also involving all the other WP3 partners, in order to ensure a proper integration with all the other system components.

In particular, Light Node is configured as a device that can interface with other existing devices, such as Smart Meters, and on which different software components can be installed, even at a later date, such as to make it easily scalable.

The Blockchain itself is not used as a database, hence the Access Layer architecture is scalable and replicable. In fact, initially the Light Node verifies the signatures by trusting the data recorded in the Blockchain Access Layer, after executing timestamping and forwarding data to the SCD. Blockchain Access Layer thus allows the flexibility market to be opened up to end users and interested stakeholders, guaranteeing the security of transactions that are certified.

The SCD is the repository of all the data concerning the resources participating in the trial and of all the elements needed for settlement (set-point, measurements, customer responses, etc.). The SCD not only enables customers and aggregator to contribute to the grid management, but also makes available to System Operators several measurements to improve grid observability. The management of access to SCD can be modified according to national regulatory requirements. Moreover, the database is easily scalable and configurable so that it can be adapted to various needs (e.g. providing data to service provider for innovative services).

Thanks to the Blockchain technology, the Access Layer makes impossible to tamper with data. In this way, all the stakeholders involved and the Platone platforms do not have to verify the reliability of the information provided since the Access Layer ensures the provision of secure and already certified data.

7 List of Tables

Table 1: Light Node Communication 11
Table 2: Blockchain Access Layer Communication 13
Table 3: Shared Customer Database Communication 21

8 List of Figures

Figure 1: Access Layer Architecture 9
Figure 2: Shared Customer Database Architecture 14
Figure 3: Communication Layer 15
Figure 4: Streaming Layer 17
Figure 5: Log Topic Kafka 17
Figure 6: Persistence Layer 19

9 List of References

[1] European Commission, “2050 long-term strategy”, [Online]. Available: https://ec.europa.eu/clima/policies/strategies/2050_en

[2] Platone Deliverable 1.1 “*General functional requirements and specifications of joint activities in the demonstrators*”, [Online]. Available: https://www.platone-h2020.eu/data/deliverables/864300_M12_D1.1.pdf

[3] MQTT.js – Open Source Javascript Library on GitHub, https://github.com/mqttjs/MQTT.js

[4] Node.js – Identity Layer, https://nodejs.org/en/

[5] Web3.js – Open Source Library”, [Online]. Available; https://web3js.readthedocs.io/en/v1.3.4/

[6] ReactJS – JavaScript library for building user interfaces, https://reactjs.org

[7] Apache Kafka – Open-source distributed event streaming platform, https://kafka.apache.org/

[8] Express JS – Framework for app development, https://expressjs.com/it/

[9] Express Gateway – Framework for app development, https://www.express-gateway.io/resources/

[10] Mongo DB – Open Source Database https://www.mongodb.com/try/download/community

[11] Aedes – MQTT Broker, https://github.com/moscajs/aedes

[12] Hyperledger – Distributed ledger software, https://www.hyperledger.org/use/besu

[13] Blockchain Consensus Algorithm – https://academy.binance.com/en/articles/what-is-a-blockchain-consensus-algorithm

[14] Open API – https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.2.md

[15] Apache – https://www.apache.org/

[16] The OAuth 2.0 Authorization Framework, [Online]. Available: https://tools.ietf.org/html/rfc6749.htm

[17] Kong – API gateway, https://konghq.com/kong/

[18] Apache ZooKeeper, https://zookeeper.apache.org/

[19] Kafka Connect – Tool for scalably and reliably streaming data https://docs.confluent.io/platform/current/connect/

[20] Elasticsearch – Distributed, RESTful search and analytics engine, https://www.elastic.co/elasticsearch/

[21] Lucene – https://lucene.apache.org/

10 List of Abbreviations

Abbreviation	**Term**
API	Application Programming Interface
BAL	Blockchain Access Layer
DERs	Distributed Energy Resources
DLT	Distributed Ledger Technology
DSO	Distribution System Operator
DSOTP	DSO Technical Platform
EMS	Energy Management System
EV	Electric Vehicle
JSON	JavaScript Object Notation
LN	Light Node
MQTT	Message Queuing Telemetry Transport
NoSQL	No Structured Query Language
P2P	Peer to Peer
PoD	Point of Delivery
SCD	Shared Customer Database
UI	User Interface