Platone

PLATFORM FOR OPERATION
OF DISTRIBUTION NETWORKS

Platone

PLATform for Operation of distribution NEtworks

D2.1 v1.0

Platone Platform requirements and reference architecture (v1)

Project Name	Platone
Contractual Delivery Date:	31.08.2020
Actual Delivery Date:	31.08.2020
Main responsible	Ferdinando Bosco (ENG)
Workpackage:	WP2 – Platform Implementation and Data Handling
Security:	P
Nature:	R
Version:	v1.0
Total number of pages:	75

Abstract

The Platone Open Framework aims to create an open, flexible and secure system that enables distribution grid flexibility/congestion management mechanisms, through innovative energy market models involving all the possible actors at many levels (DSOs, TSOs, customers, aggregators). The Platone Framework is an open source framework based on blockchain technology that enables a secure and shared data management system, allows standard and flexible integration of external solutions (e.g. legacy solutions), and is open to integration of external services through standardized open application program interfaces (APIs).

This document mainly delivery two outputs: the Platone Reference Architecture and the Platone Platforms Requirements.

The Platone Reference Architecture represents the software architecture of the Platone project, including the list of components with their interfaces, and the list of functions.

The Platone Platform Requirements includes all the functional and non-functional requirements for the design and development of the Platone Platforms.

The Platone Reference Architecture and Platone Platforms requirements are the basis for the implementation of the Platone Open Framework that will be integrated, tested and evaluated in three different demo sites: Greece, Germany and Italy. Each of these demo sites will integrate different parts of the framework.

Keyword list

Platone Reference Architecture, Platone Platforms Requirements, Platone Open Framework, Open Source, Blockchain, Energy Market

Executive Summary

The energy system is facing an incredible revolution whose end target is the creation of a new energy scenario widely dominated by renewable energy sources and mostly based on distributed energy generation. At the centre of this process is the distribution network where the majority of the new energy sources are and will be connected. Flexibility is a key resource in a scenario in which the grid is more and more changing from being a load-driven system to a generation-driven system, given the partial control on energy intake from renewable energy sources. This process implies also that the changes are not only related to the operational aspects but also to the market element. Digitalization is a key enabler of this process, opening the way to smart and efficient management of data sources in a secure way and making the separation between market and operation less and less meaningful.

Platone project proposes an innovative approach for supporting the DSOs and other involved stakeholders in the energy transition phase. Platone aims to support the observability of the network and the exploitation of the flexibility for solving both the volatility of renewable energy sources and the less predictable consumption patterns.

The Platone solution consists of a layered set of platforms to meet the needs of system operators, aggregators and end users, named **Platone Open Framework**.

The Platone Open Framework aims to create a fully replicable and scalable system that enables distribution grid flexibility/congestion management mechanisms through Peer-to-peer (P2P market models involving all the possible actors at many levels (DSOs, TSOs, customers, aggregators)). The key components for an open framework are a secure shared data management system, standard and flexible integration of external solutions (e.g. legacy solutions), and openness to external services through standardized open application program interfaces (APIs). The Platone Open Framework offers a two-layer platform and innovative components that allow targeting all the Platone objectives.

Authors, contributors and reviewers

Main responsible		
Partner	**Name**	**E-mail**
ENG		
	Ferdinando Bosco	ferdinando.bosco@eng.it
Author(s)/contributor(s)		
Partner	**Name**	
ENG		
	Ferdinando Bosco	
	Vincenzo Croce	
SIEMENS		
	Brunella Conte	
	Carlo Arrigoni	
RWTH		
	Jonas Baude	
RSE		
	Carlo Tornelli	
	Fabrizio Garrone	
	Antonio Vito Mantineo	
Reviewer(s)		
Partner	**Name**	
ARETI		
	Gabriele Fedele	
	Antonio Vito Mantineo	
	Stavroula Tzioka	
	Eleni Daridou	
Approver(s)		
Partner	**Name**	
RWTH		
	Padraic McKeever	

Table of Contents

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 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), customers and aggregators. In particular, the DSO 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 see the DSO evolve into a new form: a market enabler for end users and a smarter observer of the distribution network. By defining this innovative two-layer architecture, Platone removes 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 in 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”.

The Platone solution consists of a two-layer architecture named **Platone Open Framework**. The Platone Open Framework includes the following components:

Blockchain Service Layer**:** this layer enables the deployment of different blockchain-based components, providing a blockchain infrastructure and Smart Contracts services. In the context of Platone, the Platone Market platform is an example of blockchain-based platform deployed on it.

Platone Market Platform: it allows the support of wide geographical area flexibility requests from TSOs and local flexibility requests from DSOs. These are matched with offers coming from aggregators, resolving conflicts according to pre-defined rules of dispatching priorities. All the market operations are registered and certified within the blockchain service layer, ensuring a transparency, security and trustworthiness among all the market participants.

Blockchain Access Layer**:** this layer adds a further level of security and trustworthiness to the framework. It is an extension of the physical infrastructure and performs multiple tasks, among which are data certification and automated flexibility execution through Smart Contracts. It includes the Blockchain Access Platform and the Shared Customer Database.

Platone Blockchain Access platform: it implements all the functionalities offered by the blockchain technology through smart contracts and provides an interface for the integration of the data coming from the physical infrastructure.

Platone Shared Customer Database: it contains all the measurements, set points and other needed data collected from customer physical infrastructure. It allows the other components of the Platone Open Framework to access data in an easy way and without compromising security and privacy.

Platone DSO Technical Platform: it allows DSOs to manage the distribution grid in a secure, efficient and stable manner. It is based on an open-source extensible microservices platform and allows to deploy, as Docker containers, specific services for the DSOs and execute them on Kubernetes. The Data Bus layer, included on the DSO Technical Platform, allows integration both of other components of the Platone framework and of external components (e.g. DSO Management System) with a direct connection to the classical supervisory control and data acquisition (SCADA) system adopted by the DSO and served by standard communication protocols.

The design and the implementation of the Platone Platforms have been conducted starting from use cases, scenarios and requirements defined together with demo participants. The Platone Open Framework will be released as open source software for satisfying those requirements and ready to be exploited, integrated, tested and evaluated in three different demo sites: Italy, Greece and Germany.

1.1 Task 2.1

Starting from overall requirements and use cases provided by T1.1, Task 2.1 focuses on the definition of the Platone reference architecture and platforms requirements. The Platone reference architecture will be the base for the implementation of the Platone framework including the development of the components (T2.2, T2.3 and T2.5) the definition of the interoperability mechanisms and communication protocols (T2.4) and the delivery of the Platone integrated framework prototype (T2.6)

Functional requirements will represent the list of functional properties that need to be implemented and finally supported within the context of the Platone framework. Non-functional requirements will concern security, performance, interoperability and scalability aspects.

1.2 Objectives of the Work Reported in this Deliverable

This deliverable provides the first version of the Platone Reference Architecture together with functional and non-functional requirements expected for the Platone Platforms. An updated version of this deliverable is expected on M30 (February 2022).

The main goal of the reference architecture is to provide an overview of the Platone Open Framework from different point of views, in order to allow all stakeholders involved to understand the characteristics and potential of the framework. In addition, the reference architecture together with functional and non-functional requirements is the starting point for the design, development and release of the Platone Platforms.

1.3 Outline of the Deliverable

Chapter 2 describes the methodology applied for design of Platone architecture and the collection of the requirements.

Chapter 3 provides the Platone reference architecture. It mainly focus on blockchain technology, since it is the core technology of the Platone Open Framework, the list of components expected to be implemented or integrated within the framework and how the different demos' architectures will exploit the Platone Open Framework.

Chapter 4 represents the functionalities offered by the different Platone Platforms through a logical view of the architecture. In the chapter the Platone Platforms are described more in detail and all the information collected by other technical partners regarding other external systems are reported.

Chapter 5 represents the processes included in the Platone Open Framework through a process view. It also includes a brief description of the interoperability mechanisms and communication protocols.

Chapter 6 represents the development view of the Platone architecture. It outlines all the technologies and programming languages used within the different Platone Platforms.

Chapter 7 represents the deployment or physical view of the Platone architecture. It described the two different approaches expected for the deployment of the Platone Platforms: Cloud (Software-as-a-Service) or on premises (via Docker Containers).

Chapter 8 reports the list of functional and non-functional requirements expected for all the Platone Platforms: Market Platform, DSO Technical Platform, Blockchain Access Platform and Shared Customer Database.

Finally, Chapter 9 discusses the conclusions of the deliverable.

1.4 How to Read this Document

This document reports the Platone Reference Architecture with a big focus on Platone Platforms implemented within WP2. Other external systems, implemented within other WPs (WP3, WP4 and WP5) are listed and briefly described. A greater level of detail will be included in the deliverables of the respective WPs.

Use cases and scenarios, used for defining the functional and non-functional requirements, are available in D1.1 [2], D4.1 [3] and D5.2 [4].

2 Methodology

This section presents the approach and methodology that was followed to define the first version of the Platone reference architecture and the list of functional and non-functional requirements, as well as the technical specifications, for the architectural components.

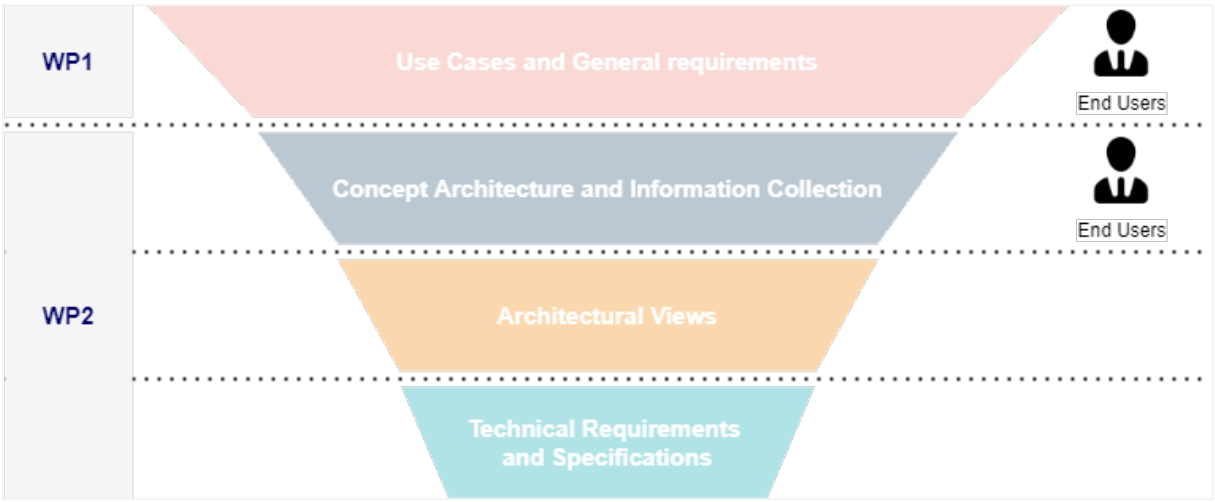

Figure 1: Platone architecture design methodology

The activity carried out for the design of Platone's reference architecture and the definition of technical requirements and specifications, is divided into 4 macro-phases:

1. Use cases definition and General requirements collection (conducted in WP1)
2. Concept architecture definition and collection of the architectural and hardware components description and technical specifications
3. Design and representation of the architecture in a standard model
4. Definition of functional and non-functional requirements for the architectural components

2.1 Use Cases and General requirements

The first step for defining the Platone reference architecture and platforms' requirements was conducted in the context of WP1 but it was fundamental as input of the entire process.

As described in D1.1 [2] the result of this activity was to describe and compare the different Platone demonstrations architecture, enhancing the understanding of the peculiarities of each demo. The output of this activity was:

- The definition and comparison of the **use cases** for each different demo (using the IEC 62559 standard)
- The mapping on **SGAM architecture** at project level
- The collection of **general requirements**

2.2 Concept architecture design and components description

In this second stage we used the inputs coming from the first phase in WP1, together with the initial architecture described in in Platone DoW [5] (Figure 2), for extending the concept of the Platone Open Framework. The main goal in this stage was to refine the initial solution, to allow the integration of the Platone framework within the different Demos architectures and define the functionalities expected.

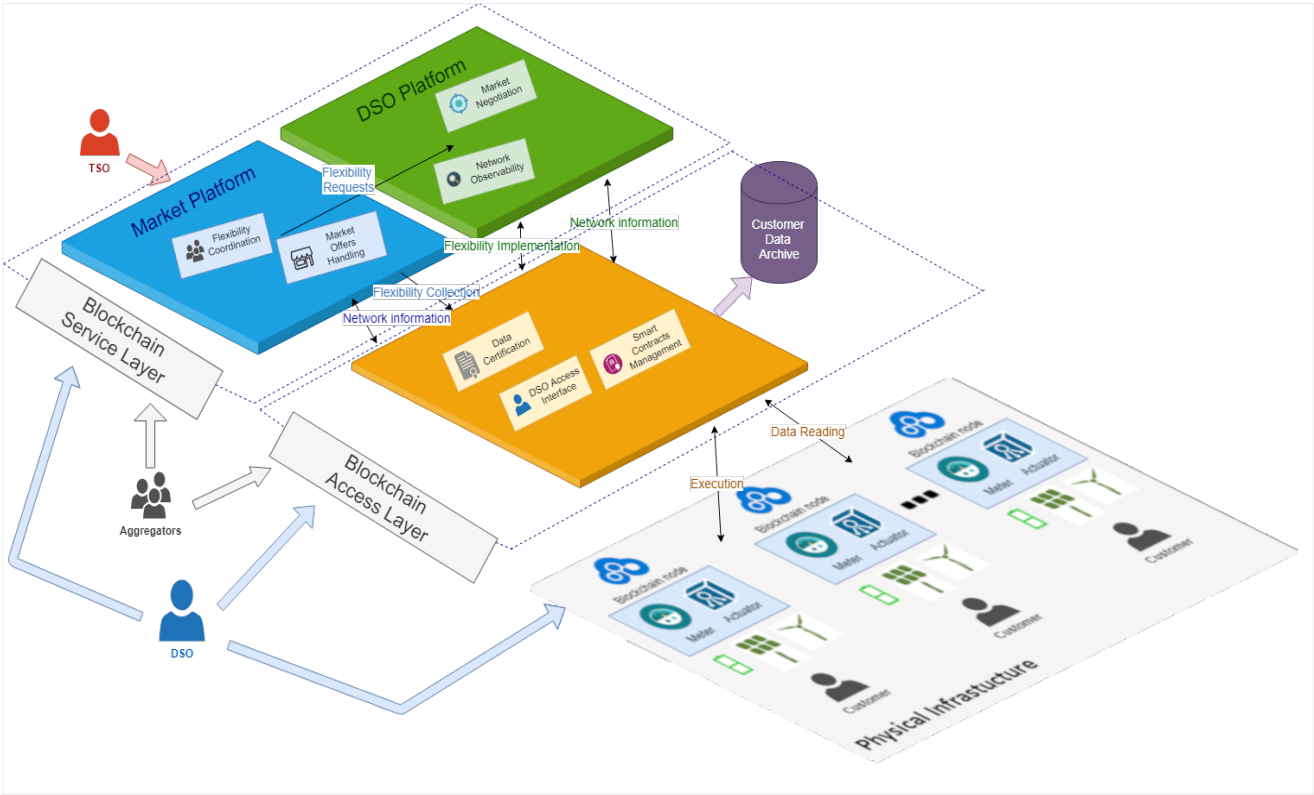

Figure 2: Initial Platone Framework Architecture

As initial step, we provided to all the Demo leaders and technical partners a template for collecting technical information about the architectural components (systems) and hardware components (devices). These templates are attached in this document in the Annex A

Once collected all the information regarding the components, we arranged a first of two „Architecture Workshops“ in which technical providers and demo leaders consulted each other to get a common idea of what the Platone framework has to offer in terms of both architecture and functionalities.

The first Architectural Workshop (the second will be arranged for the second release of the Platone architecture and technical requirements in M30, February 2022), was held the 15th of July 2020 and it mainly focused on three activities:

- WP2 partners presented the concept architecture and WP2 architectural components to other partners
- Demo Leaders (WP3, WP4 and WP5) presented their point of view about the Platone architecture and how they foresee to use and exploit it in the different demo architectures
- Open discussion among all the partners with the main goal to find a common solution for the architecture and define the list of functionalities expected

The output of this first phase was the Platone conceptual architecture and the list of the components that are part of or interact with it.

2.3 Model for describing the architecture

Once defined the general concept of the Platone architecture, it was important to find an architectural representation that would address all the concerns of the different stakeholders.

The 4 + 1 View Model [6] describes software architecture using five concurrent views, each of which addresses a specific set of concerns and allows various stakeholders to find what they need in the software architecture.

- Logical View: The components and the functionalities
- Process View: Communication between processes, components and services.

- Physical/Deployment View: deployment approaches
- Development/Implementation View: development details of the components
- Use Case View: the fifth view. The use cases are used to identify architectural elements and to illustrate and validate the architecture design

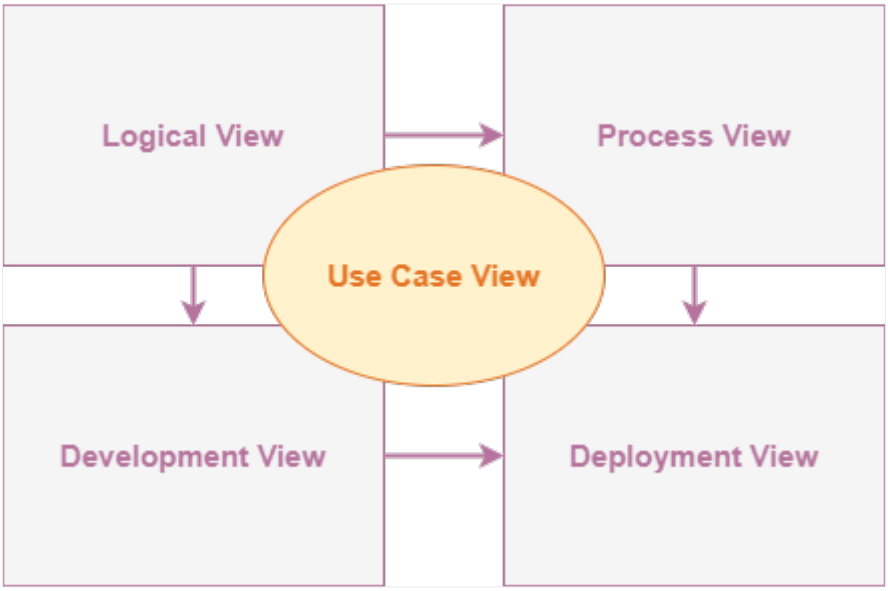

Figure 3: 4 + 1 View Model

Following the 4 + 1 model, each view is described with “generic” notations: any notations, tools or design methods can be used, especially for the logical and process analysis. UML offers many types of diagrams useful to depict an application design or technical solution in detail.

These diagrams can be used to describe the different views of the 4+1 model. Figure 4 shows which UML diagrams are best suited for each view.

However, the Platone Open Framework is more than a software solution but rather an enterprise architecture proposal for DSOs. In order to represent the whole architecture and to show how applications and infrastructure supports the DSO business processes we decided to don’t use UML for drawing the diagrams but an alternative, compatible with UML that that focuses more on enterprise modelling scope. We decided to use ArchiMate [7], an open and independent enterprise architecture modelling language and Archi [8], an open source modelling toolkit to create ArchiMate models.

ArchiMate modelling language was developed to have a common language for many different areas of an enterprise and was designed to co-exist with domain-specific languages like UML and BPMN. An element from the ArchiMate model may be related to a single element in the UML diagram, but also to a complete UML diagram (e.g. Sequence diagram, Activity diagram). Thus, the ArchiMate drawings will be integrated with UML diagrams every time a more detailed specification is needed.

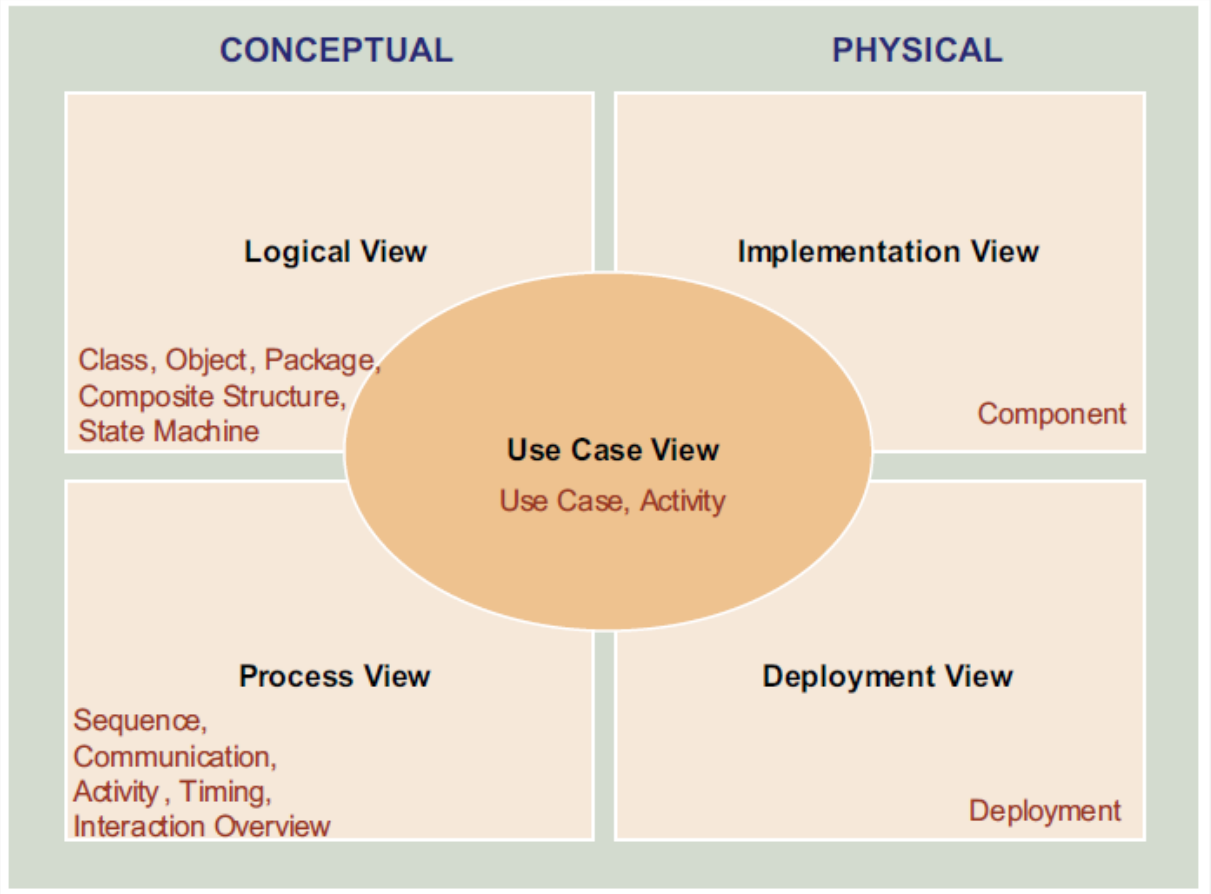

Figure 4: UML diagrams allocated to the views on the 4+1 View Model [9]

2.4 Definition of technical requirements and specifications

This final phase has the main goal to define all the technical requirements (functional and non-functional) and specifications expected for the Platone platforms developed within the WP2.

For the requirements specification, the IEC 62559 standard use case template is used. Focusing on one function or component at a time, the information exchanges with external actor are defined using “Scenario” section of the template. This part describes how the selected component interact with other parts of the system, thus specifying the functional requirements for the component. Moreover, the template sections “Information exchanged” and “Protocol” allow to describe any type of non-functional requirement (e.g. data models and formats, applicable standards, timing constraints, privacy and security requirements). The use of suitable identifiers for each information exchanged or requirement, with link to additional tables, enables a better management of non-functional requirements.

3 Reference Architecture

This chapter provides an overview of the Platone reference architecture introducing the two layers of the Platone Framework along with the included architectural components.

3.1 Terminology

Table 1: Platone Terminology

Term	Description
Platone Open Framework	The Open Source Platone Framework that includes all the Platone Platforms together with interface and mechanisms for communication and integration with external systems
Platone Platforms	The three core systems of the Platone Open Framework: Platone Market Platform, Platone DSO Technical Platform and Platone Blockchain Access Platform (that includes Platone Shared Customer Database)
Platform	Complex collection of systems, interfaces and processes integrated with each other for providing a set of functionalities and services
Reference Architecture	A reference architecture provides a template solution for an architecture for a particular domain. It also provides a common vocabulary for all the stakeholders involved.
Requirement	A requirement is a single documented physical or functional need that a particular design, product or process aims to satisfy.
Functional Requirement	Functionalities, behaviour, and information that the solution needs
Non-Functional Requirement	The conditions under which the solution must remain effective, qualities that the solution must have, or constraints within which it must operate (reliability, testability, maintainability, availability, performance)
Architectural View	An architectural view is a representation of one or more aspects of an architecture that illustrates how the architecture addresses the concerns held by one or more of its stakeholders. A View is a part of an Architecture Description that addresses a set of related concerns and is tailored for specific stakeholders. A View is specified by means of a Viewpoint, which prescribes the concepts, models, analysis techniques, and visualizations that are provided by the View.
Actor	An actor classifies a role played by an external entity that interacts with the subject (e.g., by exchanging signals and data), a human user of the designed system (Person) some other system or hardware using services of the subject.
Person	A human user or entity that interacts with the subject under development
System	A software (Component) or Hardware (Device) that interacts with the subject under development

Component	A Component represents an encapsulation of functionality aligned to implementation structure, which is modular and replaceable. A Component is a self-contained unit. As such, it is independently deployable, re-usable, and replaceable. A Component performs one or more Application Functions. It encapsulates its contents: its functionality is only accessible through a set of Application Interfaces.
Device	A Device represents a physical IT or OT resource upon which system software and artifacts may be stored or deployed for execution. A Device is a specialization of a Node that represents a physical IT or OT resource with processing capability. It is typically used to model hardware systems such as mainframes, PCs, or routers. Usually, they are part of a node together with system software
Application Interface	An Application Interface represents a point of access where application services are made available to a user, another application component, or a Node. An Application Interface specifies how the functionality of a Component can be accessed by other elements. An Application Interface exposes Application Services to the environment.
Application Process	An Application Process represents a sequence of application behaviours that achieve a specific result. An Application Process describes the internal behaviour performed by a Component that is required to realize a set of services. An Application Process may realize Application Services. Other Application Services may serve (be used by) an Application Process.
Application Function	An Application Function represents automated behaviour that can be performed by an Application Component. An Application Function describes the internal behaviour of an Application Component. If this behaviour is exposed externally, this is done through one or more services. An Application Function may realize one or more Application Services. Application Services of other Application Functions and Technology Services may serve an Application Function.
Application Service	An Application Service represents an explicitly defined exposed application behaviour. An Application Service exposes the functionality of components to their environment. This functionality is accessed through one or more Application Interfaces. An Application Service is realized by one or more Application Functions that are performed by the component.
Flexibility Service	The capability to change power supply/demand of the system as a whole or a particular unit, for responding to particular needs of the network. Some possible examples of flexibility services are load balancing, congestion management, voltage control, inertial response and black start.
Smart Grid Control	The Smart Grid Control is a digitalization of the grid and smart meters implemented for consumers. With Smart Grid Control, the demand and

	supply of electricity can be adjusted with automated real time communication between the devices. This can increase the flexibility of the power system, increase the network capacity and reduce demand of additional storage.
Flexibility Market	Flexibility market help energy networks to manage energy flows and create market signals to motivate changes in energy supply and demand, integrating smart meters, smart appliances, renewable energy resources and energy efficient resources accordingly. In Platone context, the Market Platform is component that enables the flexibility market, making available a virtual place where the requests of flexibility match the offers
Platone Flexibility Market	Open, non-discriminating, economic dispute settlement blockchain-based infrastructure, to give to both customers and aggregators the possibility to become flexibility market players more easily. This solution will observe the DSO evolving into a new role: a market enabler for end users and a smarter observer of the distribution network.
Blockchain	A blockchain is a digital record of transactions. The name comes from its structure, in which individual records, called blocks, are linked together in a single list, called a chain.
Blockchain Infrastructure	The blockchain data are stored on nodes (compare it to small servers). Nodes can be any kind of device (mostly computers, laptops or even bigger servers). All nodes on a blockchain are connected to each other and they constantly exchange the latest blockchain data with each other so all nodes stay up to date. They store, spread and preserve the blockchain data. The entire connection of all these nodes forms the blockchain infrastructure. [10]
Smart Contract	A smart contract is a self-executing contract with the terms of the agreement between buyer and seller being directly written into lines of code. The code and the agreements contained therein exist across a blockchain. The code controls the execution, and transactions are trackable and irreversible. The Smart Contracts facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performing of credible transactions without third parties. [11]
Interoperability	Characteristic of a system to work with other systems in a clear and standardized way, without any restrictions.
Interchangeability	The ability that an object (device, component) can be replaced by another without affecting system
Demo Site	A demo site is a pilot test of the Platone Open Framework in a specific geographic area. Platone has three demo sites: in Italy, Greece and Germany. Each demo site will run its own demo architecture integrating the Platone Open Framework. The demo sites will run in different locations of a specific geographic area depends on the use cases.
Demo Architecture	A specific demo architecture represents the list of specific demo platforms and functionalities that will be used in the different demo sites.

3.2 Concept Architecture

As already declared in introduction, one of the main goals of the **Platone Project is to create an open framework that is:**

- **Open**, to be extended and integrated with external services providing standard interfaces
- **Flexible**, in terms of integration with already existing external solutions (e.g. legacy platforms)
- **Secure**, for data handling, using blockchain technology for data certification and integrity and for data interoperability, using standard communication protocols and following the data security and data privacy best practices

All these characteristics and objectives have led us to plan the release of the Platone Framework as an **Open Source Framework**.

Open Source can bring the following benefits to the Platone Framework:

- Provide for better visibility, outreach and re-use of the results;
- Increase overall impact of the project on the scientific community;
- Pave the road to the Platone Open Framework becoming a valuable asset for DSOs and their operation;
- Engage customers and other stakeholders creating communities with strong relationships and common goals.

All the Platone Platforms, aim to pursue these objectives and to have these characteristics.

The Market Platform enables the creation of an open energy marketplace creating the ecosystem 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.

Furthermore, the DSO Technical Platform ensures an easy integration of external systems both at data level and service level.

Finally, the Blockchain Access Layer allows the integration of data coming from the physical infrastructure adding a further level of security and trustworthiness to the framework, exploiting the blockchain and smart contracts technologies.

The Figure 5 represents the reference architecture of the Platone Open Framework.

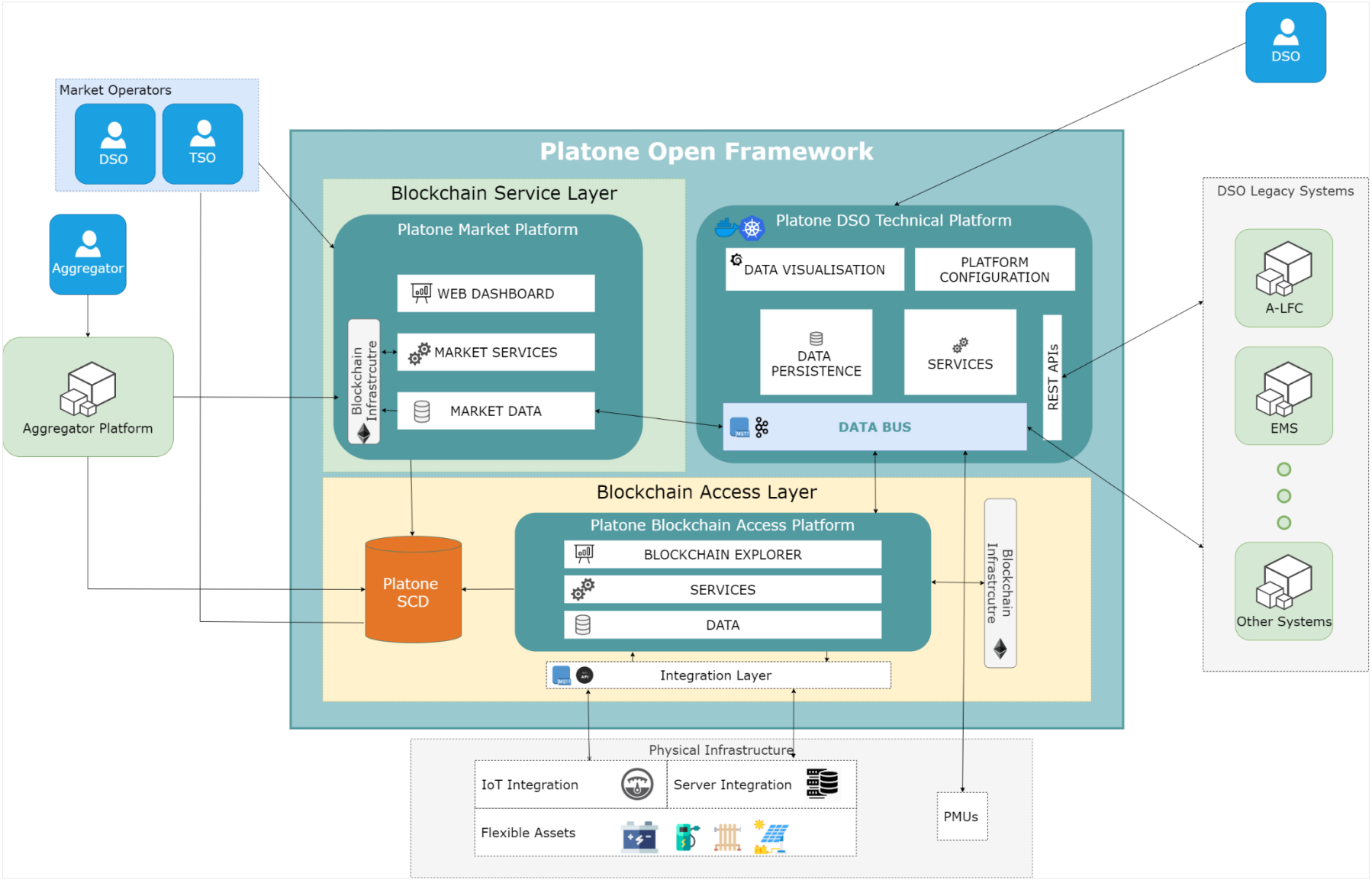

Figure 5: Platone Open Framework

3.3 Blockchain Technology

A Distributed Ledger (or Distributed Ledger Technology, DLT) is a distributed, tamper-proof registry, not controlled by a single institution. The blockchain technology belongs to the category of Distributed Ledger technologies in which transactions are grouped in blocks and chained through cryptographic hashes into an ongoing chain.

Different blockchains may differ in the consensus mechanisms and programming capabilities.

Consensus Mechanisms:

Consensus mechanisms are protocols that make sure all blockchain nodes are synchronised with each other and agree on on a single data value or a single state of the blockchain network.

These consensus mechanisms are crucial for a blockchain in order to function correctly. They make sure everyone uses the same blockchain at the same moment. Everyone can submit things to be added to the blockchain, so it is necessary that all transactions be constantly checked and that all nodes constantly audit the blockchain. Without good consensus mechanisms, blockchains are at risk of various attacks [12].

Before Bitcoin [13], there were many iterations of peer-to-peer decentralized currency systems that failed because they were unable to answer the biggest problem when it came to reaching a consensus. This problem is called “Byzantine Generals Problem”. [14] To solve this problem, Bitcoin introduced the Proof-of-Work (PoW) [15] consensus mechanism and other blockchains implemented and used other consensus mechanisms (such as Proof-of-Stake, Proof-of-Capacity, etc.) [16].

Considering the consensus mechanism, blockchains differ in the definition of the nodes’ participation in the distributed network and the roles that they can perform. In particular, we can distinguish between public and private blockchains.

Public Blockchains (also called “permissionless”) are defined in this way because they require no authorization to access the network, perform transactions or participate in the verification and creation of a new block. Anyone can participate (read and write) in the blockchain network. Public blockchains are decentralised, no one has control over the network, and they are secure in the sense that the data cannot be changed once validated on the blockchain.

On the other hand, a private blockchain is a permissioned blockchain. Permissioned Blockchains are subject to a central authority that determines who can access is authorized to be part of the network. This authority defines what roles a user can play within it, also defining rules on the visibility of recorded data. The permissioned Blockchains therefore introduce the concept of governance and centralization in a network that is born as absolutely decentralized and distributed.

Programming capabilities;

Considering the programming capabilities, we can differentiate between blockchains programmable via simple scripting (e.g. Bitcoin Blockchain) and blockchains providing Turing-complete computational capabilities, enabling the creation of “smart contracts” (e.g. Ethereum Blockchain).

Ethereum was the first blockchain supporting smart contracts and it is still the most notable example of a Turing-complete programmable blockchain, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state-transition functions. Smart contracts, cryptographic "boxes" that contain value and only unlock it if certain conditions are met, can also be built on top of the platform, with vastly more power than that offered by Bitcoin scripting because of the added powers of Turing-completeness, value-awareness, blockchain-awareness and state [17].

3.3.1 Blockchain in the energy sector

Companies looking to implement blockchain technology into wholesale electricity distribution focus on connecting end-users with the grid. Blockchain technologies combined with IoT devices enables consumers to trade and purchase energy directly from the grid rather than from retailers. Blockchain technology can provide consumers greater efficiency and control over their energy sources. Additionally, an immutable ledger provides secure and real-time updates of energy usage data [18].

As reported in the “Blockchain in Energy” report [19] by Wood Makenzie shows that 59% of blockchain energy projects are for the facilitation of electricity trading. This field is directly correlated with the primary application of blockchain, the cryptocurrency trading.

Other relevant use cases are:

- the usage of blockchain to support an electric power system that keeps the current power grid with *grid transactions*;
- the application of blockchain and cryptocurrencies to fund *energy financing* [20]
- the record of sustainability attributes
- the usage of charging infrastructure to sell charging services to EV owners [21]

More in general, the blockchain use cases in the energy sector can be classified into eight large groups, namely [22]:

1. Metering, billing and security
2. Cryptocurrencies, tokens and investment
3. Decentralised energy trading
4. Green certificates and carbon trading
5. Grid management
6. IoT, smart devices, automation and asset management
7. Electric e-mobility
8. General purpose initiatives

3.3.2 Blockchain in Platone

Blockchain Technologies and Smart Contracts play a key role in the Platone framework architecture since it is present in both the access layer and the service layer.

As described in Platone DoW [5], the usage of the blockchain technology at these two levels brings several interesting advantages:

- new schemas of coordination among customers are possible such as Peer2Peer trading
- transparent unmodifiable data management and sharing is preserved and guaranteed
- multi-party data sharing can be seamlessly extended to data collected in the field for operational purposes and not for market reasons.

Platone blockchain architecture will not start from scratch, but will leverage on the ongoing work carried out by ENG partner in the H2020 eDREAM project [23]. Platone will evolve, adapt and upscale the eDREAM open blockchain hybrid architecture, able to be deployed on the top of a variety of flexibility devices or integrated with stakeholders legacy platforms (as aggregators or DSOs internal systems).

In particular, the Platone blockchain architecture will offer multi-value functionalities covering economic transactions, business logic (e.g. grid control), data management and interoperability mechanisms.

Economic Transactions:

Ethereum smart contracts can directly manage economic transactions between two (or more) actors on a blockchain, being completely self-enforcing. It is possible to transact cryptocurrency (Ether) or customised tokens.

Currently there are two main categories of customized tokens on Ethereum: ERC-20 [24] for fungible [25] tokens that are interchangeable and not unique and so can be used to represent a value like a currency note, and ERC-721 [26] for non-fungible tokens, representing a unique asset like a collectible good.

Platone will implement the usage of tokens as a way to reward or penalise users involved in Market Operation. This approach will be better described in chapter 4.1.

Grid Control:

Platone provides mechanisms, based on smart contracts, for ensuring that the flexibility aggregation and local energy exchange are trackable and tamper-proof.

Data Management:

All the data are registered at the level of an individual prosumer (smart meter) or individual market operator (DSO/TSO/Aggregator) and then stored as immutable transactions. This allow provision of two important features: **data provenance** and **data immutability**.

Energy data (including measurement and set points) are not stored "as is" in the blockchain infrastructure since even if theoretically possible, the direct usage of a blockchain to store a big amount of data is impractical in terms of costs and performance. For this reason, the Platone framework includes an off-chain storage mechanism, based on the **Shared Customer Database**.

This approach will be better described in Chapter 4.3

Interoperability:

The replication and storage of data in a decentralised network (opposed to individual applications accessing their own information) reduces the entry barrier for new players, leading to a more competitive environment of products and services.
The Platone framework includes two different distributed ledgers: the Blockchain Service Layer and the Blockchain Access Layer. In particular, in the context of the Italian demo two different partners (ENG and APIO) will implement two different distributed ledgers solutions.

For this reason, protocols such as Interledger [27] (ILP, Interledger Protocol) or specific Interledger component, as for example the SOFIE Interledger, will be investigated in the context of Platone, aiming to achieve the integration of the two ledgers maintaining the advantages provided by the distributed ledgers of transparency, security and trust.

In particular, SOFIE Interledger component [28] is an open source component developed and tested in the context of H2020 SOFIE project [29], in which ENG is responsible of the application of the blockchain technology in energy IoT context.

SOFIE Interledger component enables activity on one ledger to trigger activity on another ledger in an atomic transaction. As shown in Figure 6, SOFIE Interledger links two different ledgers, one in the Initiator role (Ledger A) and one in the Responder role (Ledger B). The Interledger component run on a server and listens for events from the Initiator, which triggers the Interledger to call the Responder.

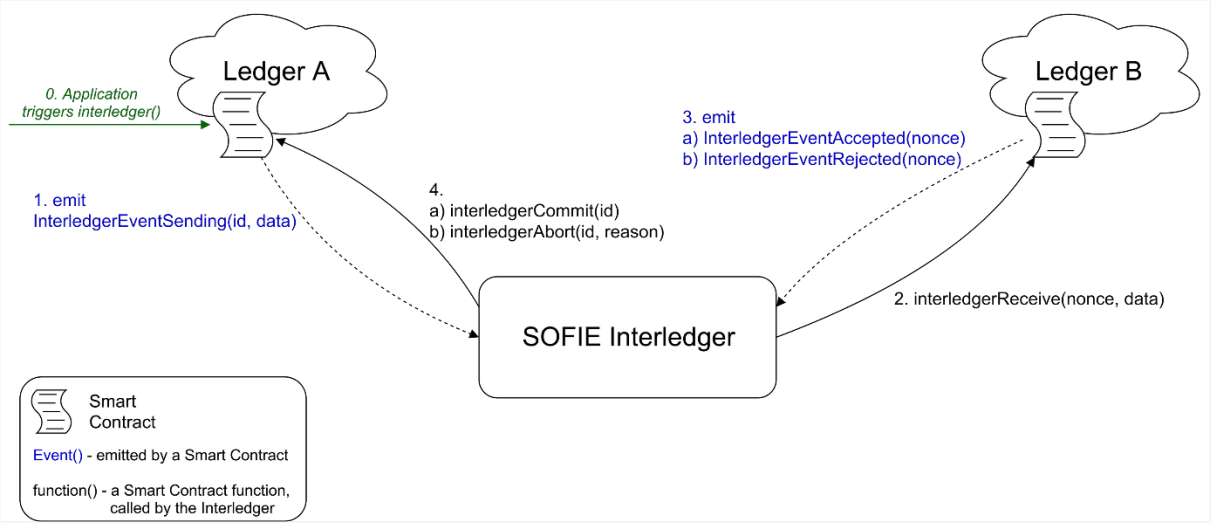

Figure 6: SOFIE Interledger component

3.4 Architectural components

During the design process of the architecture definition, the partners who will develop architectural components have been identified. The main purpose of this phase was the identification of the architectural components that should be developed or integrated within the overall Platone architecture. During the first round of information collection, a basic template was created and circulated with

requested information concerning main functionalities, dependencies, inputs needed and outputs provided. In collaboration with WP1 and the technical partners, a complete component list has been drawn up. The main purpose here is to define the list of architectural components that will be implemented (from scratch or starting from existing software) for satisfying the requirements expected in the different use cases presented in WP1.

The list of architectural components along with the associated WP and partners' responsibilities is presented in the table below.

Table 2: List of Platone Architectural Components

Component	WP	Responsible Partner	Contributing Partners
Platone Market Platform	WP2	ENG	
Platone DSO Technical Platform	WP2	RWTH	ENG, SIEM
Platone Blockchain Access Platform	WP2	ENG	
Platone Shared Customer Database	WP2	ENG	
Aggregator Platform	WP3	ACEA	SIEM
Italian DSO Technical Platform	WP3	ARETI	SIEM
Italian Blockchain Access Platform	WP3	APIO	
Italian Shared Customer Database	WP3	ARETI	
Aggregator-Customer App	WP3	ACEA	APIO
Energy Management System	WP3	APIO	
Operational Systems (including Supervisory Control and Data Acquisition - SCADA)	WP3	ARETI	
TSO Simulator	WP3	ENG	
DSO Data Server	WP4	HEDNO	NTUA
Algorithm for DER control	WP4	NTUA	
State Estimation Tool	WP4	NTUA	
Algorithm for ancillary services	WP4	NTUA	
EMS - Avacon Local Flex Controller (A-LFC)	WP5	Avacon	
BESS Data Management Backend	WP5	Avacon	
Sensor & Controller Data Management Backend	WP5	Avacon	

For all the components, a detailed description is provided in Chapter 4 including the currently known technical specifications.

3.5 Architectures of the Demos

3.5.1 Italian Demo Architecture

The Italian Demo Architecture, shown in Figure 7, represents the architecture that will be deployed in the Italian demo site and was designed following the Use Cases UC-IT-1 and UC-IT-2, described in D1.1 [2], and the related scenarios.

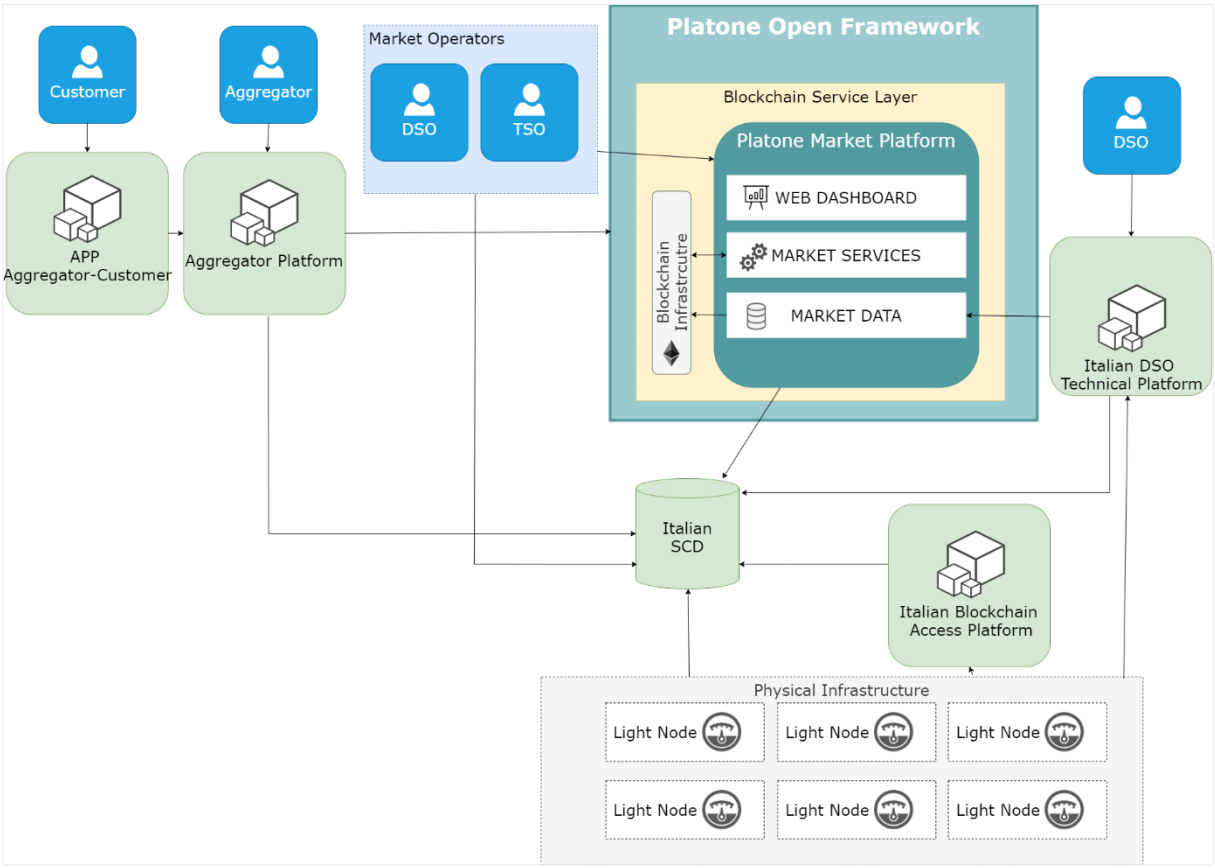

Figure 7: Italian Demo Architecture

The main purposes of the Italian Use Cases are to prevent congestion issues and avoid voltage violations in transmission and distribution systems by exploiting flexibility resources, considering all the phases concerned (procurement, activation and settlement) in the day-ahead and real time flexibility market.

In particular, three different processes are considered:

- Detection of congestion issues or voltage violation on the distribution grid by the DSO Technical Platform and definition of local flexibility requests, in the event the issue cannot be solved through the DSOTP's own solutions.
- Definition of congestion issues on the transmission network by the TSO (simulated in TSO simulator) and request of flexibility to solve them in HV grid.
- Gathering by the Aggregator Platform of flexibility offers from customers in LV and MV and offering to the Market.

These processes will be the starting point for the Flexibility Market operations.

All day ahead flexibility requests (from DSO and TSO) and offers (from Aggregator Platform) are stored in the Platone Market Platform, which matches first the offers with the DSO's requests, and orders them economically; then, it repeats the same procedure with the TSO requests.

The list of awarded offers is sent to Italian DSOTP for evaluating the grid constraints violations. Finally, the Platone Market Platform receives the list of offers compliant with local grid constraints and sends it to all the stakeholders.

At this step, the Aggregator Platform sends a reservation to the FR Owner for the resources that will be selected for the day-ahead market. The same steps are also followed in the Real Time sessions. Indeed, in these Market sessions, the offers to be matched with DSO and TSO Real Time requests are the ones still valid because not matched in previous market sessions.

The service activation phase begins when the DSO and TSO need flexibility. The DSO and the TSO communicate to the Market Platform to move a specific offer. The Market Platform sends the order to the Italian DSOTP, which divides it for every POD and dispatches the set point to the Light Nodes. The Light Nodes make available the set points to the BMS and measures the electrical quantities to be sent to the Italian SCD for evaluate the energy flexibility.

For the settlement phase, the Platone Market Platform acquires data from the Italian SCD and calculates the difference between market baseline, evaluated by BRP, and electrical quantities measured in the same time frame, uploaded in the Italian SCD by Light Nodes. The Platone Market Platform runs the settlement algorithm and finds the outcomes. Settlement outcomes are transmitted to the Aggregator Platform, the DSO and the TSO.

Finally, the DSO pays the flexibility to the Aggregator, who can pay the fee to the FR Owner.

More details in the Italian Demo architecture will be provided with the first release of the architecture expected in M18 (February 2021).

3.5.2 Greek Demo Architecture

The Greek Demo Architecture, shown in Figure 8, represents the architecture that will be deployed in the Greek demo site and was designed following the Use Cases UC-GR-1, UC-GR-2, UC-GR-3, UC-GR-4 and UC-GR-5, described in D1.1 [2] and D4.1 [3], and the related scenarios.

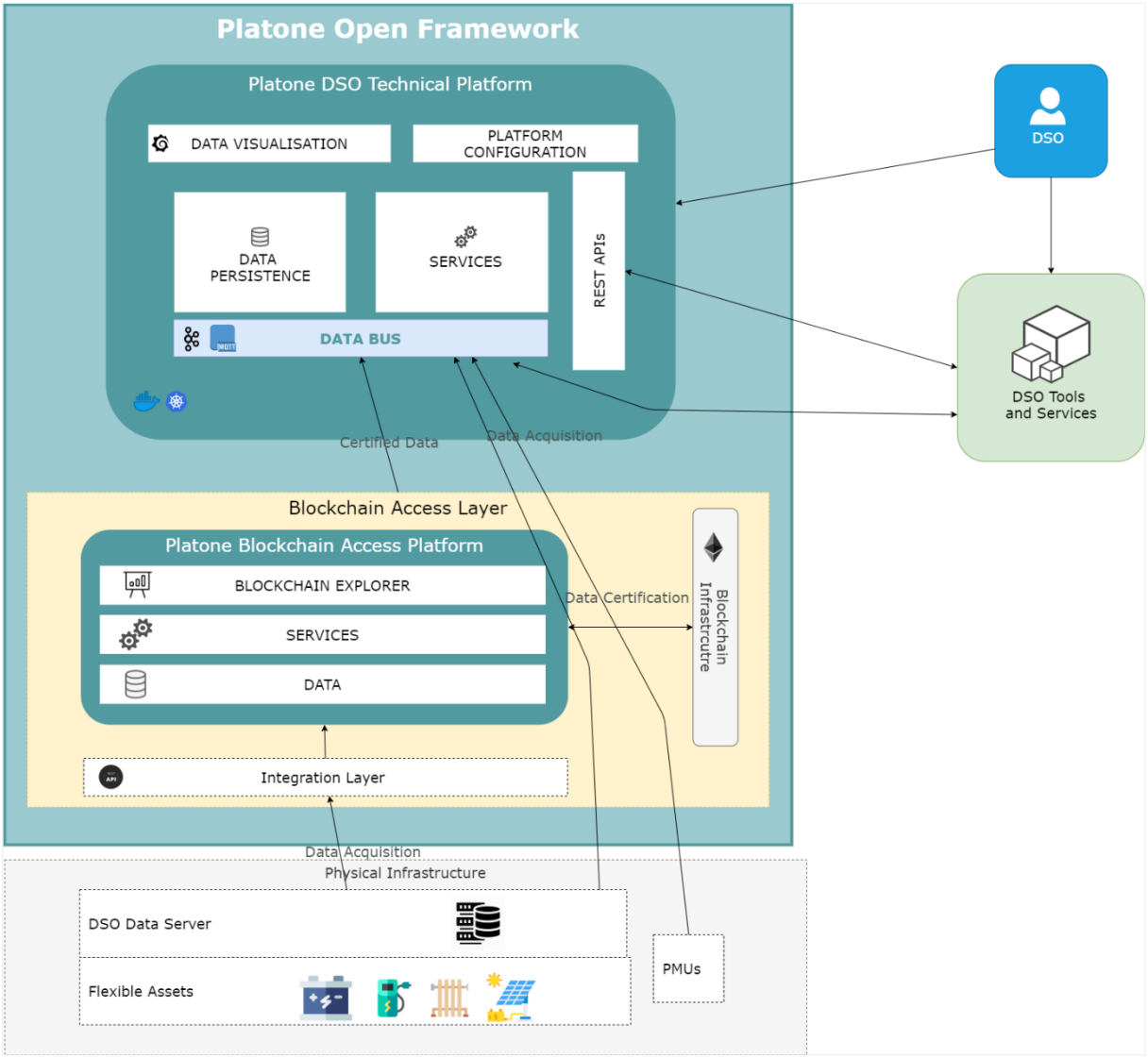

Figure 8: Greek Demo Architecture

The Greek demo architecture uses as core components the Platone DSO Technical Platform (DSOTP), and the Platone Blockchain Access Platform (BAP). DSOTP is the IT environment that includes all the tools and services that enable advanced monitoring and control of the grid. BAP is the platform that certifies measurements and customer data. The principal goals of Greek demo are to allow the DSO to achieve better observability of the distribution network via an advanced SE tool and whether adopting variable network tariffs, enables a more efficient operation of the distribution network or even the provision of ancillary services to the TSO by the end users of the distribution network.

More details on Greek Demo architecture can be found on D4.1 [3].

3.5.3 German Demo Architecture

The German Demo Architecture, shown in Figure 9, represents the architecture that will be deployed in the German demo site and was designed following the Use Cases UC-GE-1, UC-GE-2, UC-GE-3 and UC-GE-4, described in D1.1 [2] and D5.2 [4], and the related scenarios.

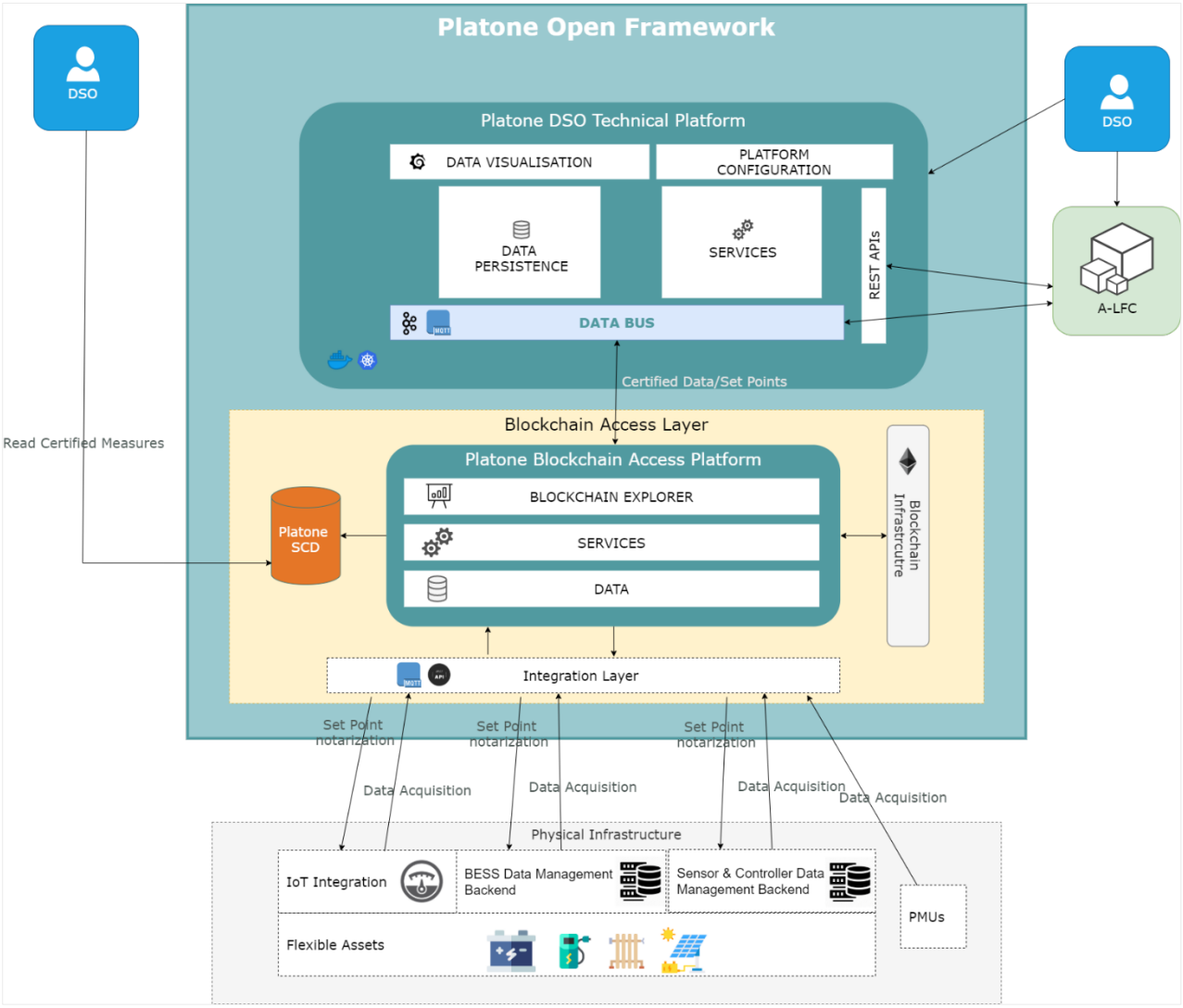

Figure 9: German Demo architecture

The German Demo architecture foresees the integration of the EMS (named Avacon Local Flex-Controller) with the Platone framework, with the main goal of monitoring and balancing a local network and implementing new strategies of energy supply: In more detail:

- Maximize the consumption of local generation, minimize the demand from the feeding grid and maximize the duration of an islanding period;
- Adhere to a fixed power value at the point of connection defined by a third party (e.g. DSO request or in response to a market signal);
- Satisfy the energy deficit left by insufficient local generation within previously defined timeslots (“Bulk supply”),
- Export the energy surplus generated by excess local generation within previously defined timeslots (“Bulk-export”).

The Avacon Local Flex Controller (ALF-C) will monitor the Local Energy Community’s generation, demand and available flexibility and send setpoints to local storages and flexible loads to fulfil requests

set by the user. The ALF-C will be integrated in the Platone DSOTP and connected to systems located in the field, such as sensors in customer households and the secondary substation providing measurement data. These systems, in turn, will be connected to Platone BAP for the certification of the measurements.

More details on German Demo architecture can be found on D5.2 [4].

4 Logical View

The logical view presents the architectural elements that deliver the system's functionalities to the end-users.

More in detail the logical view includes:

- Functional Components constitute clearly defined parts of the system that have specific responsibilities, perform distinct functions and dispose well-defined interfaces that allow them to be connected with other components.

- Dependencies are channels, indicating how the functions of a component can be made available to other components. An interface is defined by the inputs, outputs and semantics of the provided operation/interaction.

- External (third-party) entities are connectors (described as dependencies) which represent other systems, software programs, hardware devices or any other entity that communicates with the system.

The Figure 10 represents the logical view of Platone Architecture with a big focus on business processes, actors and architectural components involved.

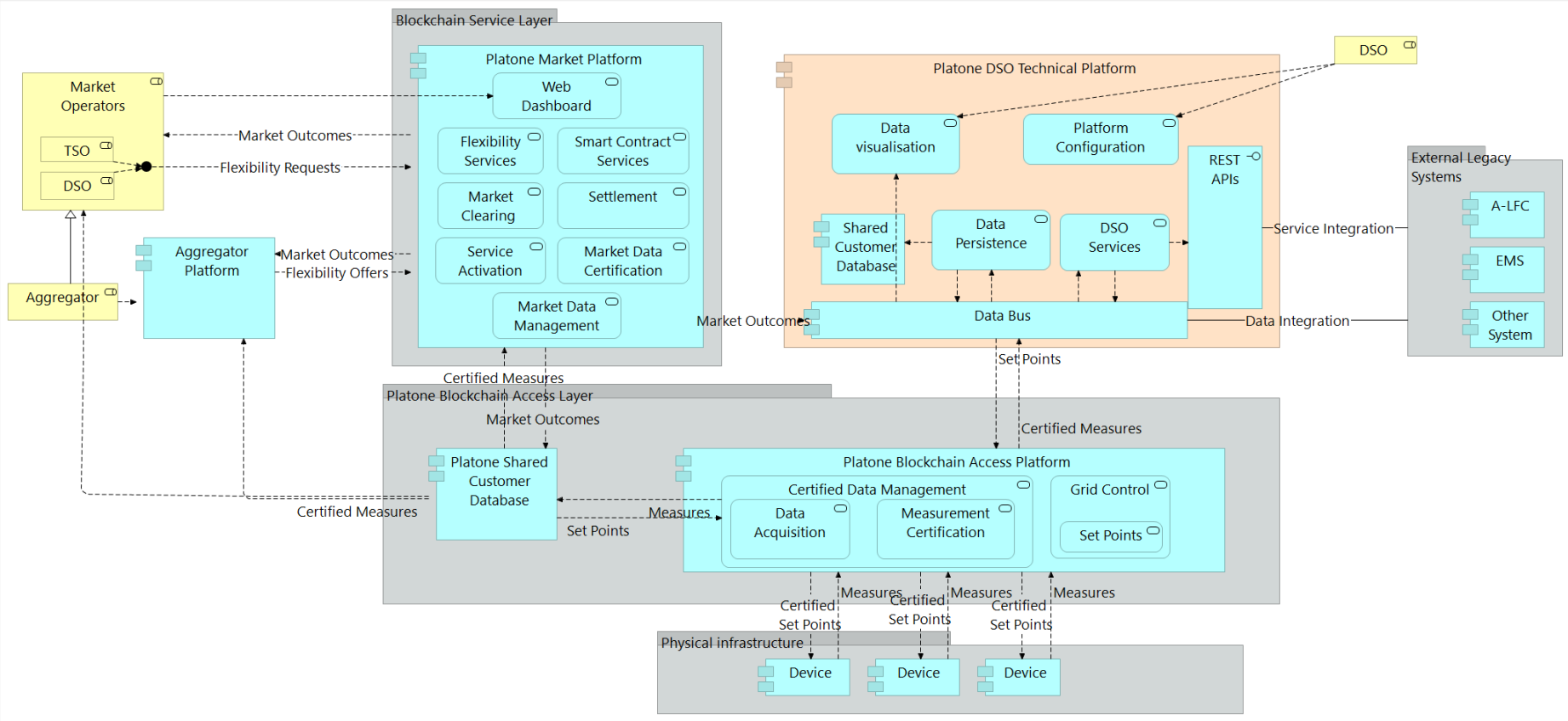

Figure 10: Logical View and functional components

The following paragraphs provide a detailed description of the core architectural components of the Platone framework, named Platone Platforms, and a brief overview of the other systems (both architectural components and hardware devices) that will interact with the Platone framework.

4.1 Platone Market Platform

The Platone Market platform is one of the core components of the Platone Open Framework. This is a blockchain-based platform that enables the management of wide geographical area flexibility requests from TSOs and local flexibility requests from DSOs. The flexibility requests are matched with offers coming from aggregators, resolving conflicts according to pre-defined rules of dispatching priorities. All the market operations are registered and certified within the blockchain service layer, ensuring a higher level of transparency, security and trustworthiness among all the market players.

The Figure 11 represents the internal architecture of the Market Platform:

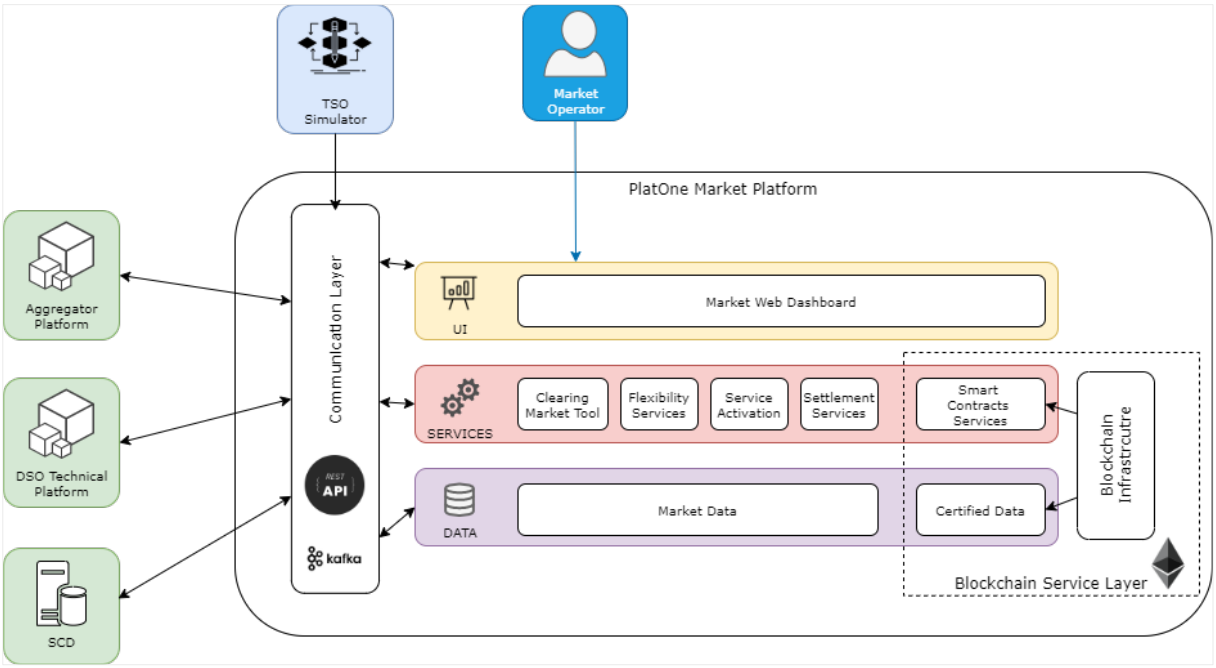

Figure 11: Platone Market Platform Architecture

The Market Platform consists of a three-layer architecture:

- **UI Layer** includes a web dashboard that allows to market players (DSOs, TSOs and aggregators) to manage their own market operations;
- **Services Layer** provides the business logic, including the market-clearing tool, the flexibility services, the settlement services and smart contract service;
- **Data Layer** provides the management of the market data and the registration of the market operations within blockchain infrastructure.

The **communication layer** allows the integration of external components and internal communication among the different layers within the Market Platform. It provides both synchronous communication interfaces (REST APIs) and asynchronous communication interfaces (Message Broker).

In particular, the Market Platform provides the following main functionalities.

Flexibility Services and Clearing Market Algorithm:

The Market Platform is able to receive flexibility services requests from DSOs and TSOs together with flexibility offers from aggregators, via REST APIs or through the web platform.

After the closing of market session, the Market Platform performs an economic phase of the Market Clearing, matching the DSOs and TSOs requests with the aggregators' offers. This clearing activity has the main purpose of satisfying the request of the DSO, which therefore will always have priority over other requests.

More in detail, the Market Platform's first step is to find, among the various offers of the aggregators, those that meet the DSO request. All the offers that fulfil the request are ordered according to an optimisation algorithm, based on a configurable multi-objective function. This optimization algorithm is based on a Non-Dominated Sorting Genetic Algorithm (NSGA-II) which provides a set of optimized solutions characterised by different suitable values with respect to the different objective functions of the optimisation process. The objective functions are defined following the indications coming from the user requirements and in particular from the DSO and could include for example, in addition to an economic factor, also an index of reliability of the flexibility providers involved (Aggregators and end-users).

Once the DSO request is satisfied, the Market Platform also tries to satisfy any remaining TSO requests. The result of this activity are the market outcomes.

Service Activation:

After the market clearing and technical validation of the market outcomes, the Market Platform is able to receive the requests of activation provided by DSOs and TSOs (via REST APIs) and to provide the aggregation of these activation requests to all the other stakeholders through the communication layer. The output is a list of set points to be activated by the different PODs.

Smart Contract Services:

The Platone Market Platform is integrated on a Blockchain Service Layer that implements a set of functionalities through Smart Contracts. This integration allows enabling a blockchain-driven energy marketplace. The blockchain service layer and the smart contract based features are treated more in detail in chapter 4.1.1.

Settlement:

After the flexibility services execution, the Market Platform acquires the data for the validation of the flexibility, analyses this data together with the respective flexibility offers, creates the Settlement outcomes, and communicates them to market players.

This process allows to DSO (or TSO if the related service was requested from it) to pay for the received flexibility service and to the aggregator to perform the settlement of the flexibility resources under their jurisdiction.

4.1.1 Blockchain Service Layer and blockchain-driven energy marketplace

The blockchain service layer is based on a blockchain infrastructure that included Ethereum blockchain nodes and smart contracts services.

In particular, the smart contracts ensure that all the processes and data flow included on the Market Platform are certified thanks to blockchain infrastructure as well as to "tokenize" the settlement outcomes enabling a token-based remuneration process that the DSO and/or TSO can exploit for payments.

The remuneration process is implemented with the usage of ERC-20 tokens [24] as a way to reward or penalise users involved in Market Operation. In particular, the tokens will be defined in a specific smart contract and assigned to prosumers in exchange for the flexibility provided. The policy for the token assignment is completely customizable and the aggregator will be responsible for specifying this policies.

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

- Ensures energy transactions certification;
- Tracks and controls the registration and validation of energy data and market data;
- Publishes bid/offer actions by Market Participants.
- Performs energy bids/offers matching and clearing price computation
- Performs a fully-transparent settlement based on tokenization

4.2 Platone DSO Technical Platform

The Platone DSO Technical Platform is another core component of the Platone Framework. The platform is based on work done in the H2020 project SOGNO [30] [31] and is capable of hosting different micro services aiming at simplifying e.g., the grid monitoring or market interactions for a DSO.

The DSO Technical Platform should respond to specific technical and operational requirements, including:

- High availability
- Scalability both horizontally and vertically
- Flexibility and modularity
- Centralized monitoring and logging
- Reduced operational cost

Figure 12 illustrates the architecture of the DSO Technical Platform. The core component of the DSO Technical Platform is the data bus. It is implemented by means of a message broker to which all services can publish and / or subscribe in order to exchange data with other services, with field devices, or with external systems. Data from field devices or external systems can be made available in the data bus either directly or through the Communication Layer of the Platone Market Platform as described and illustrated in Chapter 4.1. Depending on the use case requirements, this integration can be unidirectional or bidirectional meaning data can be sent back (e.g. set points) to field devices or external systems.

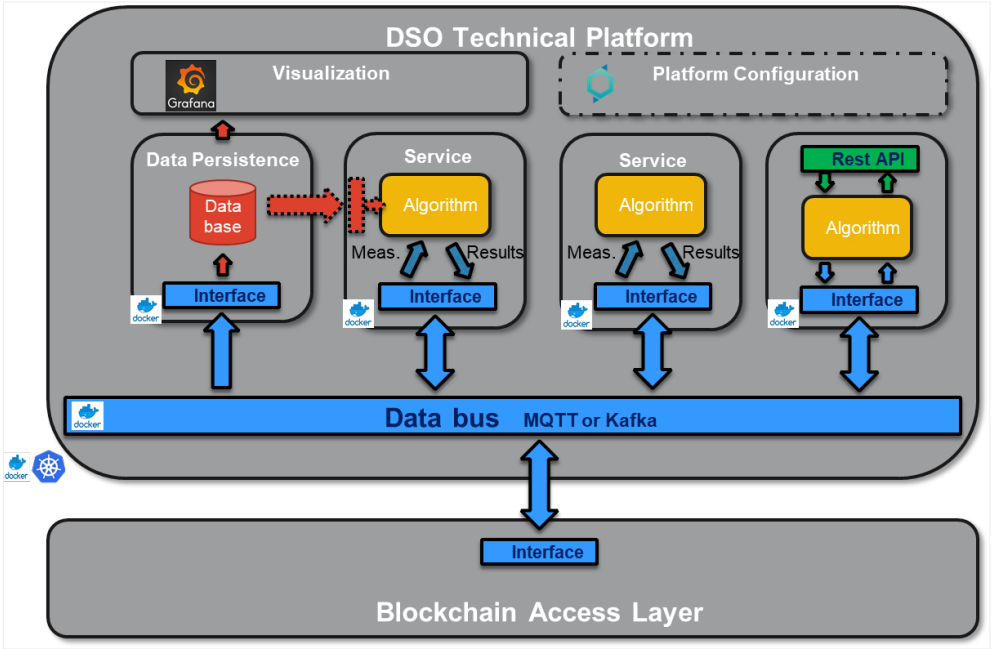

Figure 12: Platone DSO Technical Platform architecture

Data persistency within the platform is achieved by a dedicated service that stores data persistently in a database. The database will be suitable for storing time series data (e.g. measurements). The data in the database will be available to the services on the platform in order to avoid redundant databases for multiple services. Furthermore, data can be visualized in custom Grafana [32] based dashboards.

All services running on the DSO Technical Platform are deployed in individual containers that are orchestrated by Kubernetes [33], an open-source system for automating deployment, scaling, and management of containerized applications. Besides the main business logic of the service, all containers are equipped with a platform specific interface to the data bus and optionally a data base interface and/or a REST API endpoint. The programming language for implementing the main algorithms or business logic of the service is not specified in order to reduce development overhead for bringing new or existing

services to the platform. A future release of the platform might also contain a dashboard for simple creation and configuration of services.

For the initial release, services running on the DSO Technical Platform can be started with a static configuration or can implement custom service APIs. These APIs allow e.g. for a service configuration or for triggering of services on request, e.g. by an external system. These APIs can be synchronous REST APIs or asynchronous services based on the central data bus of the platform.

4.2.1 Architectural Principles

In order to satisfy all the requirements listed at the beginning of the Chapter 4.2, several key architectural principles have been identified and will be followed

- Microservices
- Strong decoupling
- Openness
- Data Security
- Continuous Supervision (monitoring and logging)

Microservices:

Core system functions are divided into microservices. Microservices contribute to assure the high availability and scalability of the system. Microservices' communication can be both synchronous and asynchronous.

Synchronous communication is based on REST API or Web Services. Every microservice can offer a REST API/Web Services to access its data and interact with it. The API model can be divided in two macro API sets: private (platform-internal) APIs and public (platform-external) APIs.

Asynchronous communication between services is based on the Data Bus. Once data are on the bus, they are handled by a set of services that are able to scale up based on system load requirements. The microservices running on the platform can be orchestrated with container orchestration solutions such as Kubernetes in order to achieve high scalability and to ensure the availability of the services.

Strong decoupling:

Each system and architectural component run independently. The specific communication pattern exploiting REST APIs/Web Services between the main system modules enables isolation and decoupling of internal functionalities.

Openness:

The system is open to be integrated, using standard IT interfaces, into a larger ecosystem of applications, including business intelligence and data analytics.

Data Security:

The system covers data security and privacy aspects following the best for protecting data from unauthorized access and data corruption throughout its lifecycle. Data security includes data encryption, hashing, tokenization, and key management practices that protect data across all applications and platforms.

Continuous Supervision (monitoring and logging):

Applications developed using the microservice architecture need to be monitored as any other type of distributed system. DSO Technical Platform should provide a centralized solution for monitoring and logging for all microservices and components needed by the platform.

4.3 Platone Blockchain Access Layer

The Blockchain access layer, as shown in Figure 13, is an architectural layer included in the Platone Open Framework that adds a further level of security and trustworthiness to the framework. It is an extension of physical infrastructure and performs multiple tasks, among which the data certification and automated flexibility execution through Smart Contracts.

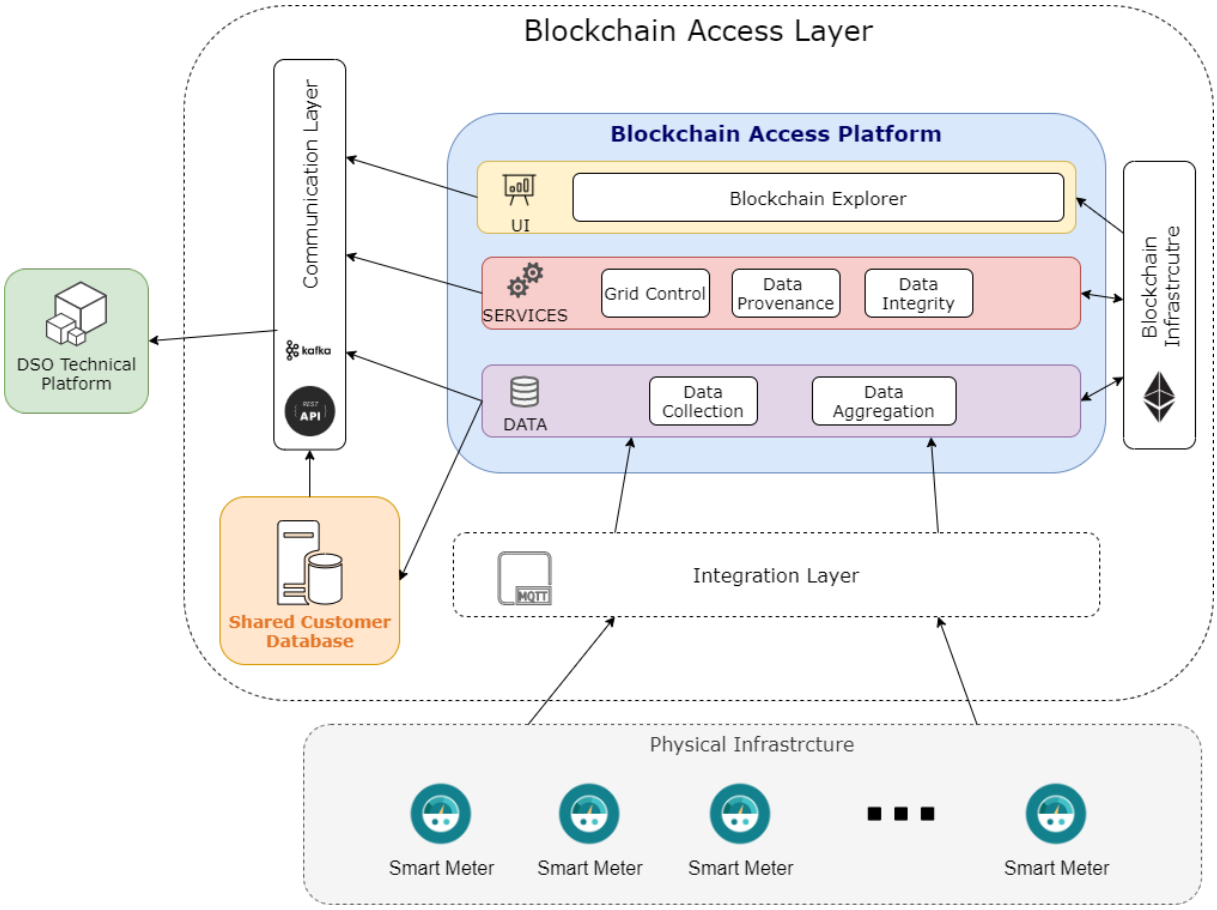

Figure 13: Platone Blockchain Access Layer architecture

This architectural layer includes two different components:

- **Platone Blockchain Access platform**, that implements all the functionalities offered by the blockchain technology through smart contracts and provides an interface for the integration of the data coming from the physical infrastructure
- **Platone Shared customer database**: it contains all the measurements, set points and other needed data collected from customer physical infrastructure. It allows the other components and stakeholders of the Platone Open Framework to access data in an easy way and without compromising security and privacy.

It also includes:

- **Integration Layer**, that allows the integration of data coming from the smart meters using standard communication protocols for IoT (e.g. MQTT)
- **Communication Layer**, enable the communication among the different internal layers of the Blockchain Platform, the SCD and external components (e.g. DSO Technical Platform). It will provide standard communication mechanisms like REST APIs and Message Broker.

- **Blockchain infrastructure,** include a private implementation of Ethereum Blockchain infrastructure including some Ethereum nodes. The blockchain nodes will contain all the transactions registered on the blockchain as well as the smart contracts deployed for implementing the features offered by the Blockchain Access Platform. The blockchain nodes may or may not coincide with the actors involved in the electrical grid. It is realistic to imagine a scenario in which large producers may afford the host locally their own full node, while small prosumers or consumers may host a Light Node (e.g. in the Italian Demo architecture, see Table 21) or choose to trust a third party node.

4.3.1 Platone Blockchain Access Platform

The blockchain technology used in the Platone project offers multiple levels of functionalities. In particular, the Blockchain Access Layer will provide:

- Data Management, all energy-monitored data are registered at the level of an individual prosumer (smart meter) and then stored as immutable energy transactions. This allows assurance of the provenance and the immutability of the energy data
- Grid Control, the blockchain can be used to control flexibility services and energy transactions. Smart contracts can be applied to prosumers' flexibility aggregation and local energy trading, making the transactions trackable and tamper-proof.

Data Immutability:

Blockchain technology can guarantee the immutability of data records once those records have entered the system. The data structures used to store the energy transactions in the ledger assure the provenance property by enacting their tracking back until the moment of their registration in the blockchain.

The distributed ledger is a collection of blocks, linked back using hash pointers, each block storing a set of valid transactions on the registered digital assets. The linked list is an append-only data structure. Any changes that would appear in previous registered nodes would lead to inconsistencies, because the hash pointer of that block would change. If one needs to change the content of a previous block, all the following blocks will need to be re-hashed and re-linked to obtain a consistent updated data structure. The advantage brought by this structure is the tamper proof log on all the transactional information contained in the blocks. Furthermore, because this is an append-only type of data structure (new blocks are always added at the head of the chain) it offers reliable historical information and preserves the order in which the energy transactions are registered.

The probability of changing the value of the transacted energy asset in a block by an attacker decreases with the number of blocks following that block in the append-only linked list.

Data Provenance:

In order to ensure the ownership of energy data, the prosumer provides the signature over the transfer transaction showing that the energy asset is his/hers, thus authenticating and validating the transfer. Whenever a new prosumer joins the blockchain network, a new account and associated contract is created, and the user account address is linked to its smart meter.

In this way, each prosumer who generates energy can register energy assets in the distributed ledger, based on the information provided by the associated smart meter, by signing and registering an energy transaction having as a receiver its own contract account address.

4.3.2 Platone Shared Customer Database

Even if theoretically possible, the direct usage of a blockchain for data storage is impractical in terms of costs and performance. While for some kind of high-value transactions this cost may be affordable (e.g. the notarisation of a real-estate transfer), for frequent low-value transactions, such as fifteen minutes readings from a smart meter, this becomes a prohibitive limit. To minimise the costs, and avoid fluctuations, it is crucial to optimise resources, minimising the usage of the on-chain storage capabilities.

Now, in M12 of the project, we are investigating two possible solutions: The first one is a hybrid solution built on top of distributed databases (e.g. Cassandra [34]) and Ethereum smart contracts, while the second one is based on the integration of BigchainDB [35] with the integration layer.

The first solution was already tested within eDREAM project and proposes a technique for tamper-evident registration of smart meters' energy data and associated energy transactions using digital fingerprinting which allows the energy transaction to be linked with the hashed version registered on the blockchain, while the plain data is stored off-chain. The prototype was implemented using Ethereum and smart contracts for the on-chain components, Cassandra as database and RabbitMQ as messaging broker [36].

4.4 Other Systems

As described in the methodology chapter, the Platone framework will interact with other components provided by demo WPs. Below are presented the main characteristics of these systems that will be described in detail in deliverables of specific demo WPs at later stages of the project.

4.4.1 Components and Services

4.4.1.1 German Demo

Table 3: ALF-C technical details

Name of Component/Service:	Avacon Local Flex-Controller (ALF-C)
Type	Service
Functionality	ALF-C monitors the state of network elements and behaviour of customers, predicts behaviour and energy demand & generation and controls flexible elements in order to maintain a predefined set point for the power exchange between local system and mid voltage feeder.
Input Connections & Interfaces	- Field Data (PMU): P, V, I, cos(phi) - Customer Data: Demand, generation (web service) - Battery System Information: P(t), SOE, SOC, alarms (MODBUS TCP via LTE) - Weather data & forecast (web service / API)
Output Connections & Interfaces	- Set point P(t) for battery (sub 60 sec.) - Set point P(t) for flexible customer (domestic battery, heat pump, A/C)(15-min) - Alarm (technician, fire brigade)
Software Requirements/Development Language	- Deployed in MS Azure, use case algorithms realized in Python - Utilizes Platone Technical Platform & Blockchain Access Layer for upstream data
Hardware Requirements	Cloud deployment, no specific requirements identified so far

Status of the development of the component	To be developed from scratch
WP and Task reference	WP5 - T5.2, T5.3

Table 4: Sensor & Controller Data Management Backend technical details

Name of Component/Service:	Sensor & Controller Data Management Backend
Type	Service
Functionality	The Sensor & Controller Data Management Backend handles first level data acquisition from household batteries and relays set points from ALF-C to the household. It monitors the situation of the battery system and provides required data to ALF-C.
Input Connections & Interfaces	- Battery sensors (cell-level, system level, SOE, SOC) - Set point P(t) from ALF-C (15-Min)
Output Connections & Interfaces	- ALF-C (all relevant system data & alarms)
Software Requirements/Development Language	- Proprietary software provided by battery vendor
Hardware Requirements	Cloud-based
Status of the development of the component	Commercially available
WP and Task reference	WP5 - T5.2, T5.3

Table 5: BESS Data Management Backend technical details

Name of Component/Service:	BESS Data Management Backend
Type	Service
Functionality	The BESS Data Management Backend handles first level data acquisition from the BESS and relays set points from ALF-C to BESS. It monitors the situation of the battery system and provides required data to ALF-C and other services (e.g. fire brigade).
Input Connections & Interfaces	- Battery sensors (cell-level, system level, point of connection, SOE, SOC, hazards /alarms) - Set point P(t) from ALF-C (sub 60-sec)
Output Connections & Interfaces	- ALF-C (all relevant system data & alarms)

Software Requirements/Development Language	- Proprietary software provided by battery vendor
Hardware Requirements	Cloud-based
Status of the development of the component	Commercially available
WP and Task reference	WP5 - T5.2, T5.3

4.4.1.2 Greek Demo

Table 6: DSO Data Server technical details

Name of Component/Service:	DSO Data Server
Type	Component
Functionality	A dockerised-database that hosts DSO data (both master data and AMR-metrics for MV and LV customers).
Input Connections & Interfaces	Data from different telemetering centers is assembled and uniformed in an offline API with the purpose of producing an incremental Docker image. The image is uploaded on Docker Hub image repository (process similar to using an ftp service, but with Docker) and then downloaded and executed from the Data-Server side.
Output Connections & Interfaces	Another API was developed with the purpose of connecting and querying the database and gathering the appropriate data (scheduled every 15 mins). The API then transforms the data into an xml whose data model, format and structure are defined in the CIM 61968-9 standard. Finally, it publishes the produced xml into a broker using MQTT protocol.
Software Requirements/Development Language	The database version used is the corresponding Docker image of MYSQL 8.0.19. The APIs are written in the Python programming-language using various external packages and libraries, to name the most important mysql-connector or Sqlalchemy for database connection handling and Pandas library for most data-related tasks.
Hardware Requirements	No specific hardware requirements identified so far
Status of the development of the component	Partially developed
WP and Task reference	WP4

Table 7: Algorithm for ancillary services technical details

Name of Component/Service:	Algorithm for ancillary services

Type	Service
Functionality	This component will generate appropriate charges for using the distribution network to incentivize optimal DER operation by Aggregators, avoiding any network limit violations when a request for an ancillary service (here Frequency Support) arrives to the Aggregator.
Input Connections & Interfaces	State Estimation Tool
Output Connections & Interfaces	Network prices (tariffs) are send to Aggregators
Software Requirements/Development Language	This algorithm is developed in Julia programming language using JuMP and Gurobi Solver.
Hardware Requirements	An ordinary computer
Status of the development of the component	Design completed
WP and Task reference	WP4, Task 4.3.1

Table 8: Algorithm for DER control technical details

Name of Component/Service:	Algorithm for DER control
Type	Service
Functionality	This component will generate appropriate charges for using the distribution network to incentivize optimal DER operation by Aggregators, while mitigating any network limit violations. Line and voltage limits are concerned
Input Connections & Interfaces	State Estimation Tool
Output Connections & Interfaces	Network prices (tariffs) are send to Aggregators
Software Requirements/Development Language	This algorithm is developed in Julia programming language using JuMP and Gurobi Solver.
Hardware Requirements	An ordinary computer
Status of the development of the component	Design completed
WP and Task reference	WP4, Task 4.4.1

Table 9: State Estimation Tool technical details

Name of Component/Service:	State estimation tool
Type	Service
Functionality	The State Estimation Tool aims to ensure that high quality estimative of the network state will be acquired in real-time conditions under various network operating scenarios. This tool filters the available measurement data, comprising actual measurements obtained from active metering devices and pseudo-measurements, i.e. data derived from load forecasting or RES scheduling for network observability accomplishment, in order to identify measurement with gross errors (bad data), to suppress measurement errors, to reconcile inconsistent data and, ultimately, to estimate the actual operational network state.
Input Connections & Interfaces	SCADA, DMS, GIS, AMR, DSO Data server
Output Connections & Interfaces	Algorithms for DER control and Ancillary Services
Software Requirements/Development Language	N/A
Hardware Requirements	N/A
Status of the development of the component	Design completed
WP and Task reference	WP4, Task 4.2

4.4.1.3 Italian Demo

Table 10: Operation Systems technical details

Name of Component/Service:	Operational Systems (including Supervisory Control and Data Acquisition - SCADA)
Type	Component
Functionality	Control system architecture managed by DSO for high-level process supervisory management, its main functionalities are: - Network diagnostic; - Remote grid control management; - Gather signals and measurement from the fields

Input Connections & Interfaces	Voltage and current sensors in Primary substation (IEC 61850) Remote Terminal Unit in Secondary Substation (IEC 61850)
Output Connections & Interfaces	To be defined
Software Requirements/Development Language	C, Java, Javascript
Hardware Requirements	**Virtualized hardware:** VMware ESXi 6.7 model HPE ProLiant DL380 Gen10 equipped with CPU Xeon Platinum, 256GB DDR4 RAM, double power supply 500W, 400Gb SSD MU, Smart Array P440/2G Controller, 4 NIC 1Gb, 2 NIC 10Gb **Standard virtual machine:** vCPU 4 VM RAM Cfg (GB) 16 Host RAM Use (GB) 10 VM DISK Cfg (GB) 96 VM DISK Actual (GB) 48
Status of the development of the component	Already developed
WP and Task reference	WP3 – T3.3.1

Table 11: Aggregator Platform technical details

Name of Component/Service:	Aggregator Platform
Type	Component
Functionality	Platforms used by Aggregator to take part in the flexibility market, the main functionalities are: - Evaluating the baseline of DER group; - Defining the volume-price for the offer; - Despatching the flexibility order; - Defining the customer revenues and penalties.
Input Connections & Interfaces	• Market Platform (REST APIs) • SCD (REST APIs and/or Message Broker)
Output Connections & Interfaces	• Market Platform (REST APIs) • SCD (REST APIs and/or Message Broker)
Software Requirements/Development Language	This platform follows the Microservices Architecture patterns. The microservices are organized in the following packages:

	• Data acquisition Package • Core Package • Site Package • Energy Management Package • Algorithms Package The development languages and technology used are: • Node.js/JavaScript • Kafka • Redis • Docker • Kubernetes • Sencha Extjs • React • MongoDB • MQTT • C
Hardware Requirements	Platform available on cloud. It is deployed on Kubernetes. Depending on the scalability needs org the business logic, its microservices are grouped into Kubernetes pods and released or updated independently.
Status of the development of the component	Partially developed: the core of the platform already exists. New services and interfaces will be developed
WP and Task reference	WP3 – T3.4.1

Table 12: App Aggregator-Customer technical details

Name of Component/Service:	App Aggregator-Customer [Included in Aggregator Platform]
Type	Component
Functionality	Application to increase the awareness and the involvement of the customer in the flexibility market
Input Connections & Interfaces	• Aggregator Platform (REST APIs)
Output Connections & Interfaces	• End User (REST APIs)

Software Requirements/Development Language	Cross Platform Framework for mobile/desktop applications (Ionic) ReactJS as UI Library Typescript as programming language
Hardware Requirements	No specific Hardware requirements identified so far
Status of the development of the component	to be developed from scratch
WP and Task reference	WP3 – T3.4

Table 13: Italian Blockchain Access Platform technical details

Name of Component/Service:	Italian Blockchain Access Platform
Type	Component
Functionality	Platform that certifies customer data for the flexibility
Input Connections & Interfaces	Light Node • PLC-C (Chain 2) to 2G Smart Meters • Modbus TCP (LAN) • Modbus RTU (RS485) • REST API
Output Connections & Interfaces	Shared Customer Database: • REST – API • KAFKA (Optional)
Software Requirements/Development Language	The Blockchain Access Layer Platform will be composed at least of the following modules and Development languages: • Web Dashboard – ReactJS • Backend and services – NodeJS • Data Layer – MongoDB • Blockchain infrastructure – Ethereum / Quorum • Smart Contracts Services - Solidity
Hardware Requirements	Light Node will be composed at least of the following modules: • Communication module to Chain 2, 2G counter (PLC-C) • RS-485 serial module • LTE / HSPA + / UMTS / GPRS / GSM multi-band cellular communication module • Wi-Fi and Bluetooth module • Ethernet LAN module

	• Power supply module • ARM system-on-module (e.g. Cortex A5) with at least 512MB RAM, storage memory with a minimum capacity of 8GB preferably Flash memory and possibility of an expansion via microSD • RTC - Real Time Clock module
Status of the development of the component	to be developed from scratch
WP and Task reference	WP3 – Task 3.2

Table 14: EMS technical details

Name of Component/Service:	Energy Management System
Type	Component
Functionality	The system is installed in customer premises and performs the follow main functionalities: - Monitoring of energy consumption and production - Activation of set point coming from Light-Node - Manage operation building systems, smart appliances and devices - Customer energy awareness
Input Connections & Interfaces	The input connections and interfaces available depend from Customer's EMS. The Light Node is able to exchange data with Energy Management System by: • REST-API • Modbus TCP • Modbus RTU (RS485)
Output Connections & Interfaces	The output connections and interfaces available depend from Customer's EMS. The Energy Management System shall be able to exchange data with Light Node by: • REST-API • Modbus TCP • Modbus RTU (RS485)
Software Requirements/Development Language	Customers Energy Management Systems should be integrated by the following modules: • Integration Microservices – NodeJS

	For some customers, an Energy Management Systems should be the one developed by Apio in the past. In this case, the EMS is composed by: • Web Dashboard – ReactJS • Backend and services – NodeJS • Data Layer – MongoDB
Hardware Requirements	In addition to the ones for the connections and interface, hardware requirements are the typical ones used by Energy Management System. The Energy Management System developed by Apio in past experience is composed by: • ARM system-on-module (e.g. Cortex A5) with at least 512MB RAM, storage memory with a minimum capacity of 8GB preferably Flash memory and possibility of an expansion via microSD; • Wi-Fi and Bluetooth module; • Ethernet LAN module; • Power supply module;
Status of the development of the component	Already developed (limited to Apio EMS)
WP and Task reference	WP3 – task 3.2.2 (limited to Apio EMS)

Table 15: Italian DSO Technical Platform technical details

Name of Component/Service:	Italian DSO Technical Platform
Type	Component
Functionality	Platform used by DSO to manage the network, the main functionalities are: - Calculate the flexibility requests; - Assessment of the flexibility offers; - State estimation
Input Connections & Interfaces	• Market Platform (REST APIs) • SCD (REST APIs and/or Message Broker) • Light Node (REST APIs and/or Message Broker)
Output Connections & Interfaces	• Market Platform (REST APIs) • SCD (REST APIs and/or Message Broker) • Light Node (REST APIs and/or Message Broker)

Software Requirements/Development Language	C, Java, Javascript, Matlab, Gams and other Programming Languages that will be defined
Hardware Requirements	Hardware requirements still to be defined
Status of the development of the component	Partially developed: the main tools are been implemented, but some services and interfaces will be developed again
WP and Task reference	WP3 – T3.3.1

Table 16: Italian SCD technical details

Name of Component/Service:	Italian Shared Customer Database (SCD)
Type	Component
Functionality	SCD hosts the data for characterize the flexibility, its main functionalities are: - Storage of energy flexibility data; - Share the data with all the stakeholders;
Input Connections & Interfaces	• Italian Blockchain Access Platform (REST APIs) • Aggregator Platform (REST APIs and/or Message Broker) • Market Platform (REST APIs and/or Message Broker) • Light Node
Output Connections & Interfaces	• Italian DSO Technical Platform (REST APIs) • Aggregator Platform (REST APIs and/or Message Broker) • Market Platform (REST APIs and/or Message Broker) • TSO SIMULATOR
Software Requirements/Development Language	The Shared Customer database will have different communication layers, as Kafka, Apache Zookeeper and Fluentd, and also MongoDB and Elasticsearch for data storage. The Programming Languages will be Java and JavaScript.
Hardware Requirements	The hardware requirements will be tailored to the needs of the project on amazon cloud.
Status of the development of the component	To be developed from scratch, currently we are writing the functionality requests.
WP and Task reference	WP3 – Task 3.2.3

Table 17: TSO Simulator technical details

Name of Component/Service:	TSO Simulator

Type	Component
Functionality	The TSO Simulator simulates possible TSO's network congestion and then provide to Market the flexibility requests;
Input Connections & Interfaces	TSO Simulator will work with historical data, other external input are not necessaries
Output Connections & Interfaces	TSO Simulator will provide flexibility requests to the Market Platform using HTTPs requests and exploiting Market Platform REST APIs
Software Requirements/Development Language	The historical data will be stored in a No-SQL database (MongoDB) and the services will be implemented in NodeJs and ExpressJS Framework.
Hardware Requirements	No particular hardware is required
Status of the development of the component	From Scratch
WP and Task reference	WP3

4.4.2 Hardware devices

This list reports the hardware devices that will be installed on the field of the different demos. At the moment (M12) some device parameters are not identifiable and will be included with the second version of the deliverable expected for M30 (February 2022).

Table 18: Customer-owned Flexibility specification

Name of Device	Customer-owned Flexibility (Domestic batteries, electric heating)
Description	Devices on customers' premises which can respond to ALF-C control signals. Batteries to charge / discharge on demand, heaters / heat pumps to modulate momentary power demand / heat load.
Functionality	Respond to power setpoints defined by ALF-C
Measurement	Power, voltage, current, reactive power, temperature, SOE, SOC
Measurement Range	TBD
Measurement Resolution	TBD
Accuracy	TBD
Data Connections	Ethernet / Wi-Fi, REST API
Data Output Format	TBD
Data Rate	TBD
Data Availability	TBD

Table 19: Large Batter Energy System specification

Name of Device	Large Battery Energy System
Description	Stationary grid scale battery to provide the required flexibility to enable use cases in pilot network.
Functionality	Charges and discharges in response to ALF-C setpoints, provides all relevant data to ALF-C
Measurement	Power, voltage, current, reactive power, temperature, SOE, SOC
Measurement Range	TBD
Measurement Resolution	TBD
Accuracy	TBD
Data Connections	Cellular, REST API
Data Output Format	MODBUS TCP
Data Rate	TBD
Data Availability	TBD

Table 20: PMU specification

Name of Device	Phasor Measurement Unit (PMU)
Description	PMU collects data on field level
Functionality	Required to provide data for use case execution
Measurement	Power, voltage, current, cos(phi)
Measurement Range	TBD
Measurement Resolution	TBD
Accuracy	TBD
Data Connections	Ethernet, Wi-Fi or cellular
Data Output Format	IEC 61850, MQTT, UDP-IP
Data Rate	TBD
Data Availability	TBD

Table 21: Light Node specification

Name of Device	Light Node
Description	Device that reads, arranges, certifies in Blockchain and sends the measurement and data provided by Smart Meter to the SCD (shared customer database). Moreover, the device receives set point from Italian DSO Technical Platform and make it available to Customer apparatus (e.g. EMS).
Functionality	• Gather the high granularity measurements (every 4 sec.) provided by Smart Meter; • Certifies on Blockchain the data; • Make available set-point to customer's apparatus (e.g. EMS)
Measurement	The measurements read by the Light Node are: • Energy Active and Reactive productions and consumptions (15 minutes, Daily and Monthly); • Power Active and Reactive productions and consumptions (4 seconds, 15 minutes, daily and monthly); • EMS Responses; • BMS Responses;
Measurement Range	The Light Node just read the measure make available by connected Smart Meter (or other meters or apparatus connected to Light Node).
Measurement Resolution	N.A. The Light Node just reads the measurement made available by connected Smart Meter (or other meters or apparatus connected to Light Node).
Accuracy	N.A. The Light Node reads the measurement made available by connected Smart Meter (or other meters or apparatus connected to Light Node).
Data Connections	• Communication module to Chain 2, 2G counter (PLC-C); • RS-485 serial module; • LTE / HSPA + / UMTS / GPRS / GSM multi-band cellular communication module; • Wi-Fi and Bluetooth module; • Ethernet LAN module;
Data Output Format	The Light Node will send Data Output in JSON standard form.
Data Rate	The Light Node collects data at least 4 seconds time interval

Data Availability	Light Node continuously sends the Data Stream to the Shared Customer Database. It is also possible to ask additional data directly to the Light Node (if necessary)

Table 22: EV Charging station specifications

Name of Device	Flexible Resources [EV charging station]
Description	Public charging points equipped with a socket of 3,7 kW and another of 22 kW
Functionality	Smart charging of the EV
Measurement	Too early to define it
Measurement Range	Too early to define it
Measurement Resolution	Too early to define it
Accuracy	Too early to define it
Data Connections	Too early to define it
Data Output Format	Too early to define it
Data Rate	Too early to define it
Data Availability	Too early to define it

Table 23: Storage System specification

Name of Device	Flexible Resources [Storage System]
Description	Storage System consisting in Lithium battery and related Management System installed in premises of the Low Voltage customer
Functionality	Increase the flexibility of the LV customers. The related Management System is able to manage the battery operation

	(charging/discharging) in order to provide flexibility services accordingly to grid requests.
Measurement	Too early to define it
Measurement Range	Too early to define it
Measurement Resolution	Too early to define it
Accuracy	Too early to define it
Data Connections	Too early to define it
Data Output Format	Too early to define it
Data Rate	Too early to define it
Data Availability	Too early to define it

Table 24: LV Circuit breaker with IED specifications

Name of Device	Sensors [LV Circuit breaker with IED]	
Description	LV Switches embed with Intelligent Electronic Devices (IED) that gather the measurements (V, I, cos φ,…) for every LV line	
Functionality	Observability of LV network	
Measurement	V on bus-bar, and I, P, Q, and cos φ, for every line	
Capacity	Rated insulation voltage Rated current Rated short-circuit breaking cap. Closing power in short circuit	690 V 400 A 20kA 400V cosϕ=0,25 42kA(cr) 400V cos φ=0,25[1]
Measurement Resolution	N.A.	
Accuracy	N.A.	
Data Connections	Modbus RTU	
Data Output Format	Modbus RTU frame	

[1] (cr) is Italian “cresta”, meaning peak.

Data Rate	Up to 19200 bps
Data Availability	on demand

Table 25: Voltage and current sensor specification

Name of Device	Sensors [Voltage and current sensors]
Description	Device installed in Primary Substation to measure the electrical quantities and send them to SCADA.
Functionality	Observability of the MV network
Measurement	V for every bus-bar, and I for every MV feeders
Measurement Range	Rated primary current of application: up to 3 200 A Rated primary voltage of application: up to 40.5 kV
Measurement Resolution	N/A
Accuracy	Current accuracy class: up to 0.5/5P630 Voltage accuracy class: up to 0.5/3P
Data Connections	Cable connector type: Twin BNC
Data Output Format	Analog output
Data Rate	Not applicable
Data Availability	Continuous

5 Process View

The process view analysis of the system defines how the system actually works in the runtime environment and how it performs in response to external (or internal) signals. The interactions between the system's actors and system's components are usually data flows representing the information exchanged in parallel or sequential execution of internal tasks

In addition, this view also highlights the communication protocols and the interoperability mechanisms that allow the connection among the different components (internal or external) included in the Platone Framework.

5.1 Interoperability Mechanisms and communication protocols

Interoperability within the Platone Framework will be assured by specifying all the interfaces between the different components of the framework and between the individual services that are operating on incoming data.

Asynchronous streaming data from field devices or legacy systems will enter into the blockchain access layer in a vender specific format and will be fed into the data bus of the DSOTP. The data bus will be based on MQTT or Apache Kafka using JSON formatted messages. The messages can contain data based on existing standards in the respective field (wherever feasible), or can contain custom messages in order to fulfil the needs of the different Platone use cases. However, our work package aims at identifying synergies between use case specific data formats in order to harmonize and specify them accordingly.

Synchronous communication interfaces between components of the Platone Framework or with services running on the DSOTP or on the Market Platform should be implemented with REST APIs that will be specified in accordance with the OpenAPI Specification (OAS) [37]

5.2 Processes and diagrams

From a processes perspective, Platone is aimed to produce a collaborative system of independent components sharing domain specific ontology, data and functional aspects that can be clustered into three main processes:

1) Market interactions
2) Grid operative control/Flexibility Services Activation
3) Measurement Acquisition and Certification

The Market interactions process, as shown in Figure 14 is implemented only in the Italian Demo (WP3) and consists of the following steps:

1- Platone Market Platform collects flexibility offers (from Aggregator Platform) and flexibility requests (from DSOs and TSOs)
2- Platone Market Platform matching flexibility request and offers
3- Platone Market Platform provides market outcomes (the results of the market clearing) to Italian DSOTP for technical feasibility evaluation
4- Italian DSOTP validate market outcomes and provide results to Platone Market Platform
5- Platone Market Platform provides validated market outcomes to all the Market Players (DSOs, TSOs and Aggregators)
6- Platone Market Platform receive setpoints to be activated (see process 2)
7- Italian Blockchain Access Platform collects measurements (see process 3)
8- Italian SCD provide measurements for validation and settlement to Aggregator Platform and Platone Market Platform
9- Platone Market Platform performs settlement and provide economic output to Aggregators, DSOs and TSOs

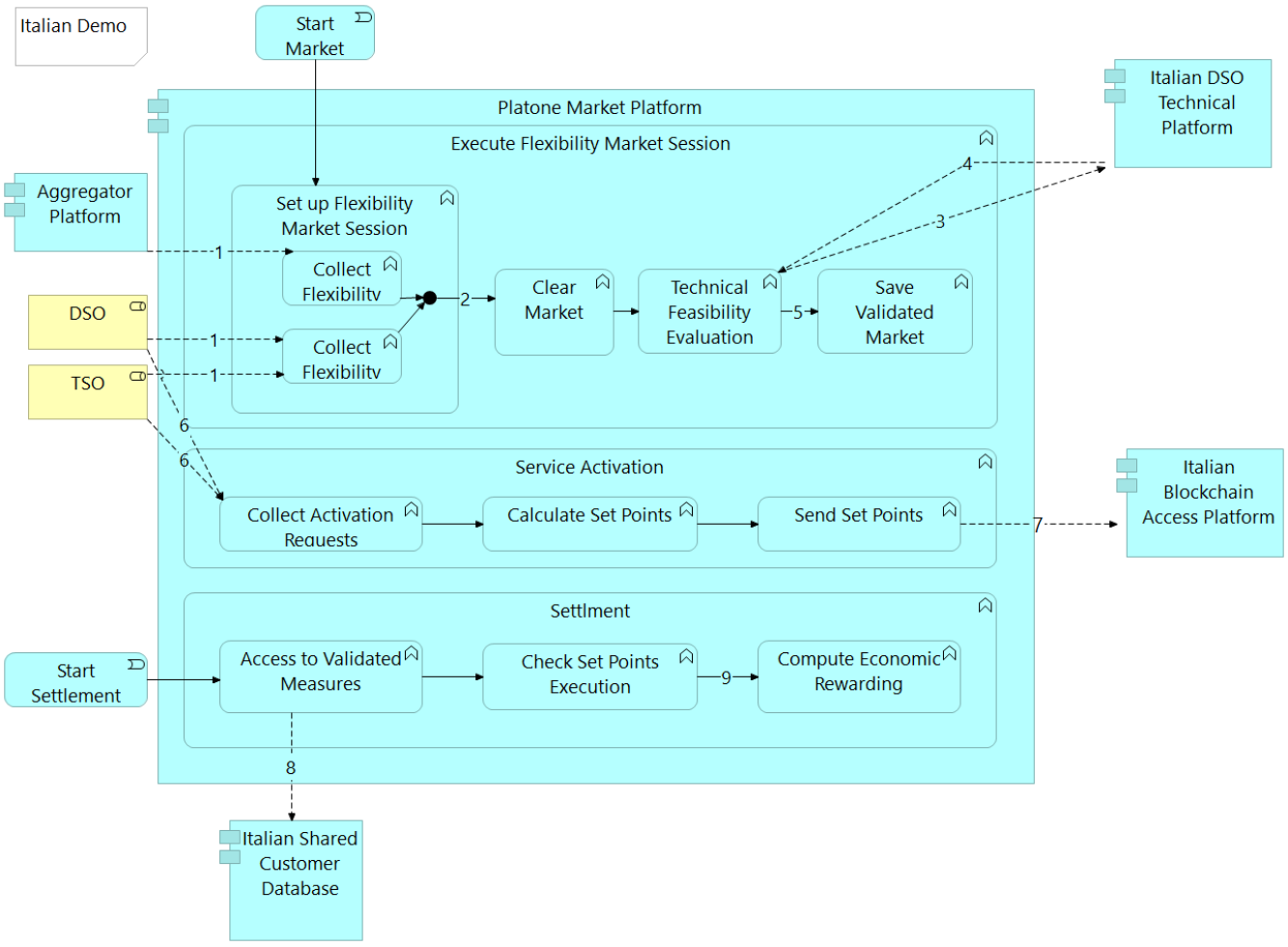

Figure 14: Process View - Market Interactions

The Grid Control and Flexibility Services activation process is implemented in German Demo (Figure 15) and Italian Demo (Figure 16). The process consists of the following steps.

German Demo:

1- Platone DSOTP collects setpoints from external systems (EMS)
2- Platone DSOTP:
   a. provides data to Platone Market Platform and Platone BAP via data bus
   b. uses data for specific DSO services (e.g. data visualization)
3- Platone BAP in in charge of:
   a. Certifying data on blockchain infrastructure
   b. Registering data on Platone Shared Customer Database (SCD)
   c. Providing setpoints to physical infrastructure.

Italian Demo:

1- Platone Market Platform collects setpoints from DSOs and TSOs;
2- Platone Market Platform provide setpoints to Italian DSOTP, Italian SCD and Aggregator Platform;
3- Italian DSOTP:
   a. Provides data to Italian BAP,
   b. uses data for specific DSO services;
4- Italian BAP in in charge of:
   a. Certify data on blockchain infrastructure,
   b. Provide setpoints to Light Nodes.

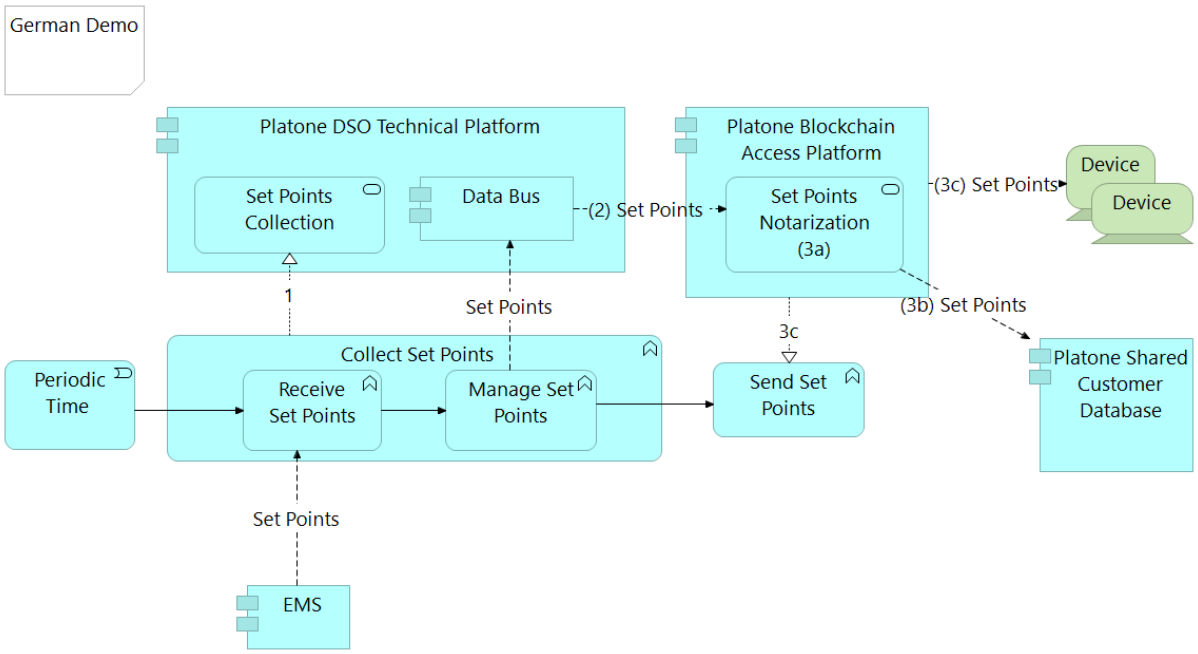

Figure 15: Process View - Grid Control German Demo

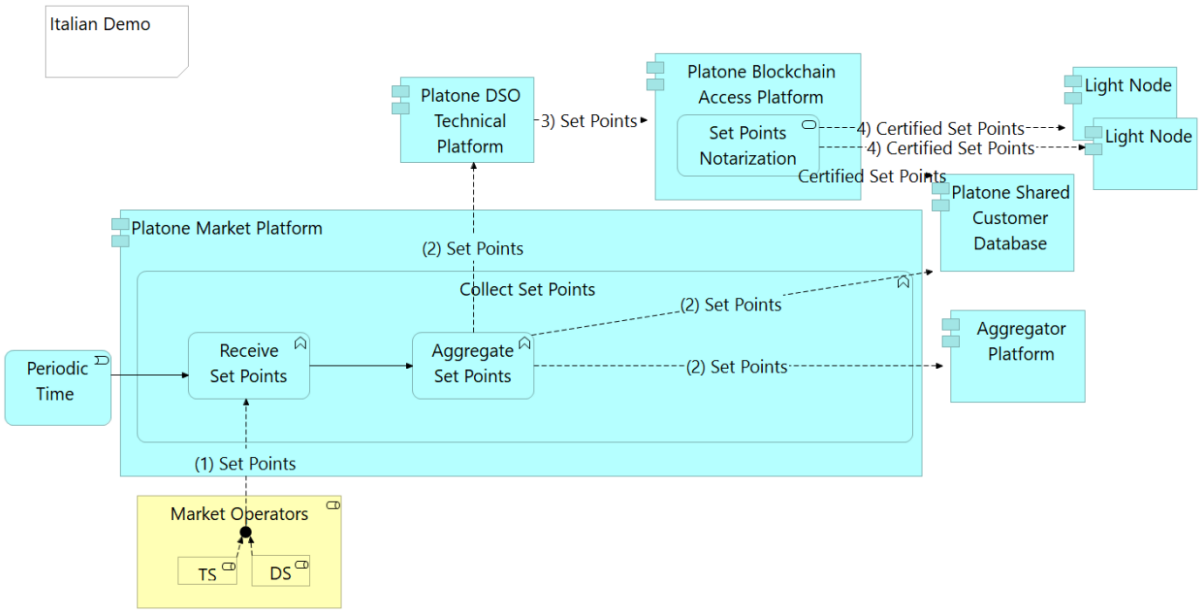

Figure 16: Process View – Flexibility Services Activation Italian Demo

The Measurements acquisition and certification process is implemented in all the three demos with some small differences in the implementation of the Italian demo. As shown in Figure 17, this process consists of the following steps:

1- Platone BAP collects measurement from Physical infrastructure;
2- Platone BAP in in charge of:
    a. Certify data on blockchain infrastructure;
    b. Register data on Platone SCD;
    c. Provide data to Platone DSOTP data bus.
3- Platone DSOTP:

a. provides data to other systems (included Market Platform and external systems) via data bus;
b. uses data for specific DSO services (e.g. data visualization).

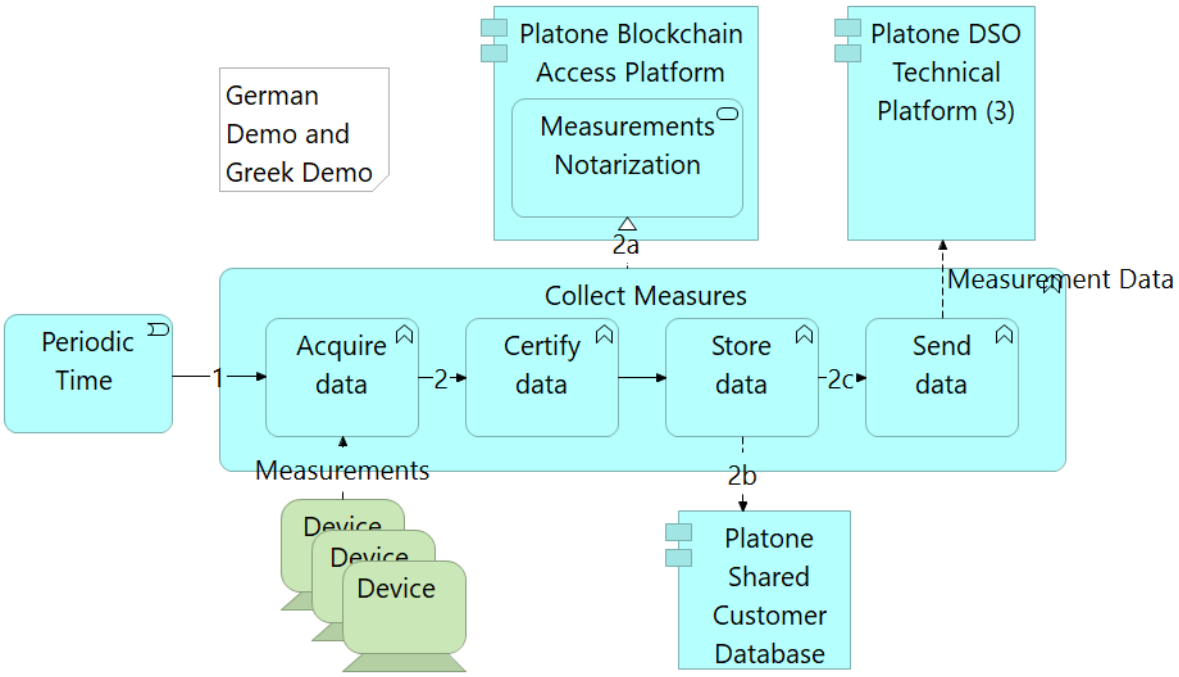

Figure 17: Process View - Measurements Acquisition and Certification

6 Development View

Within this section, we will provide information about the development of the Platone components, including the implementation approach and the programming languages/technologies that may be used by each component.

Each main component of the system should offer to the others a way to communicate with each other. Contents and mode of this communication is the first step to achieve, in order to design well defined boundaries between system components that will allow development groups to choose the best technology and architecture design for their own components.

This complex architecture has a natural inhomogeneity of software, technologies and languages used, as each development group choices are driven by different habits and competence. The complexity of each component can also be very different, bringing developers to adopt different choice and architectures, also considering re-use of already existent software.

The table and the figure below report the list of technologies and tools used within the different Platone Platforms.

Table 26: Platone Platforms technologies

Platform	Application	Data Storage	Presentation	Communication
Platone Market Platform	Javascript NodeJs ExpressJs Ethereum Solidity Docker	MongoDB	Javascript HTML5 CSS/SCSS Vue.js	REST APIs Apache Kafka
Platone DSO Technical Platform	Java / J2EE Node.js/JavaScript Docker Kubernetes Matlab Python Go	Redis ELK (Elasticsearch) PostgreSQL Oracle Timescale Influx	JavaScript CSS/SCSS React Angular Grafana/Prometheus	RabbitMQ Apache Kafka REST APIs
Platone Blockchain Access Platform	Javascript NodeJs ExpressJs Ethereum Solidity Docker	MongoDB	Javascript HTML5 CSS/SCSS	REST APIs Rabbit MQ
Platone Shared Customer Database	Javascript NodeJs ExpressJs	Cassandra/MongoDB		Rabbit MQ REST APIs

	Docker			

Implementation Approach:

The implementation approach for the Platone Open Framework will be based on Iterative and incremental development. [38]

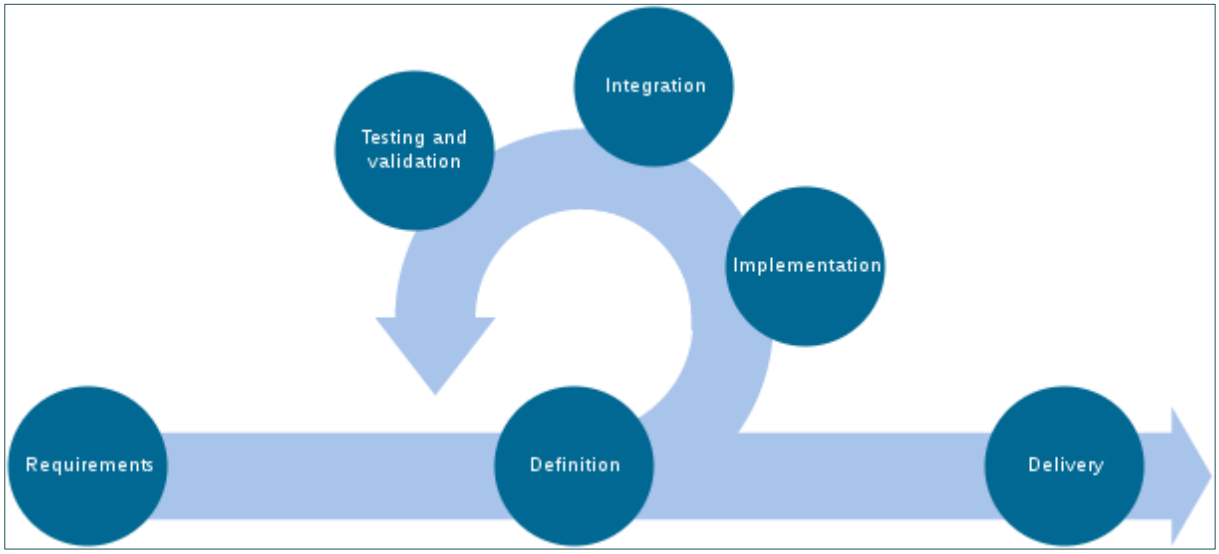

Figure 18: Iterative and incremental development approach

In particular, it foresees an iterative incremental approach based on three main phases. In each phase will be delivered a prototype of Platone Open Framework and Platone Platforms.

Phase 1 (M1-M20), It will include definition use cases, scenarios, user and technical requirements, the definition of reference architecture. The **first version of Platone Platforms** will be released at M18 (February 2021) and the **first version of the integrated Platone Open Framework** will be released at M20 (April 2021). This version of the Platforms will include a subset of the overall functionalities expected.

Phase 2 (M21-M40), it is based on the feedback from Phase 1. The KPIs, scenarios and both user and technical requirements will be refined (M30, February 2022) and a second version of the platforms will be released (M38, October 2022) and integrated (M40, December 2022) in an **intermediate version of the Platone Open Framework**, functionally complete.

Phase 3 (M41-M48), it based on the evaluation results of Phase 2. Usability, user behavior evaluation and impact creation will be analysed and assessed. Furthermore, this phase will take in account the preliminary simulation results, using them as feedback for the **final Platone Open Framework** release (M48, August 2023, end of the project).

Open Source:

As already mentioned, the Platone Open Framework will be released as Open Source. The project roadmap described in the implementation approach will be also followed for the project releases and shared with external communities.

The Platone open source approach foresees:

- Open Source licensing for all the Platone Platforms and Platone Open Framework (e.g. Apache 2.0);
- Usage of a public visible repository for source code (e.g. GitHub);
- Planned releases of the software (following project roadmap);
- Creation of specific documentation (README, FAQ, Change Log, Release Notes, etc...).

7 Deployment View

The Physical View presents aspects of the system that are connected with the realization of the system's components in the physical world.

Platone will provide different approaches for the deployment: on cloud or installation on premises.

The Platone Market Platform and the Blockchain Access Layer (including the Platone Blockchain Access Platform and the Platone Shared Customer Database) will be will be available both in the cloud and for the installation on premises whereas the Platone DSO Technical platform will be released only for installation on premises.

7.1 Cloud Hosting and Software-as-a-service model

The Platone Market Platform, the Platone Blockchain Access Platform and the Platone Shared Customer Database will be deployed and hosted by ENG in its own server farm located in Pont-Saint-Martin (Italy).

In particular, ENG already provides a blockchain infrastructure that consists of a series of Ethereum nodes in which blockchain-based application can run and an API gateway that allows managing all API calls from clients, then routes them to the appropriate component or service.

The Platone components will exploit this blockchain infrastructure and API gateway offering their functionalities in a Software-as-a-Service (SaaS) model.

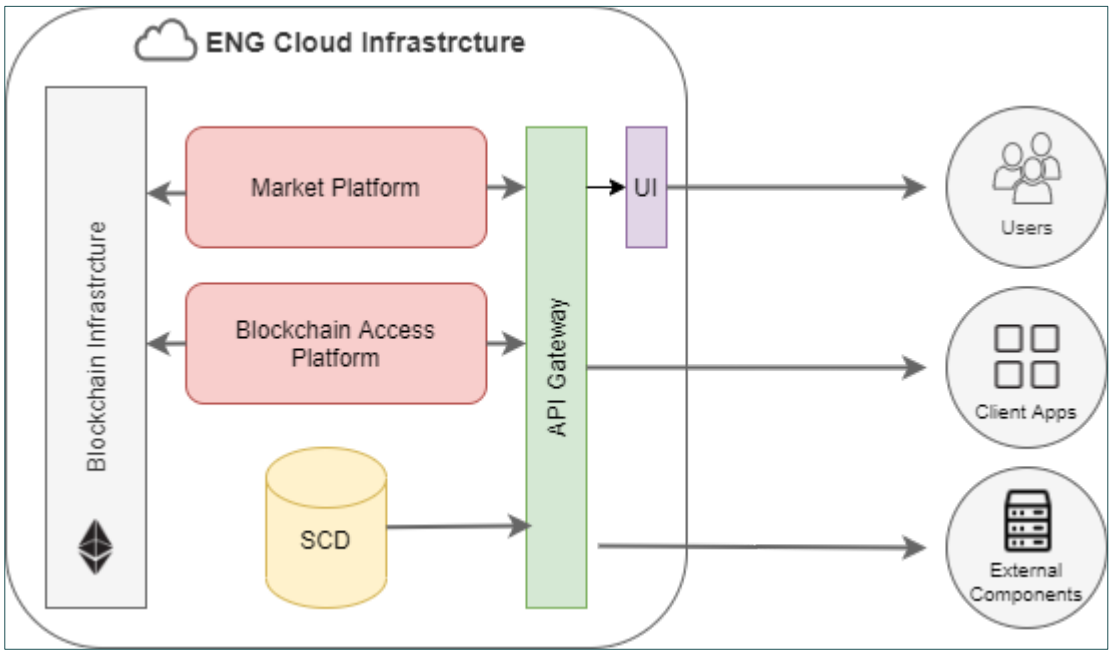

Figure 19: ENG cloud infrastructure

7.2 Installation on premises and containerization

All the Platone Platforms will also be packaged and released for the deployment on demo premises. To facilitate the release of the components, Platone foresees use of the Docker Containerization approach.

Containerization:

This term refers to the use of containers. A container is a virtualized server at the operating system level, for which the virtual instance only concerns the user space, i.e. the application execution environment.

Containers are not installed like traditional software programs, which allows them to be isolated from the other software and the operating system itself. [39]

The isolated nature of containers provides several benefits. First, the software in a container will run the same in different environments. Second, containers provide added security since the software will not affect the host operating system.

Containers also eliminate installation issues, including system conflicts, version incompatibilities, and missing dependencies. The result is a "works on all machines" solution, which is ideal for both developers and end users. It also makes the jobs of network administrators easier, since they can deliver containers to a multiple users without having to worry about compatibility issues.

Therefore, we do not virtualize the processor, storage, network connections, etc. of the physical server, which remain shared between the running containers. This approach means that containers are "lighter" than virtual machines, require fewer resources and can be activated very rapidly and therefore can respond to situations with sudden loads and peaks.

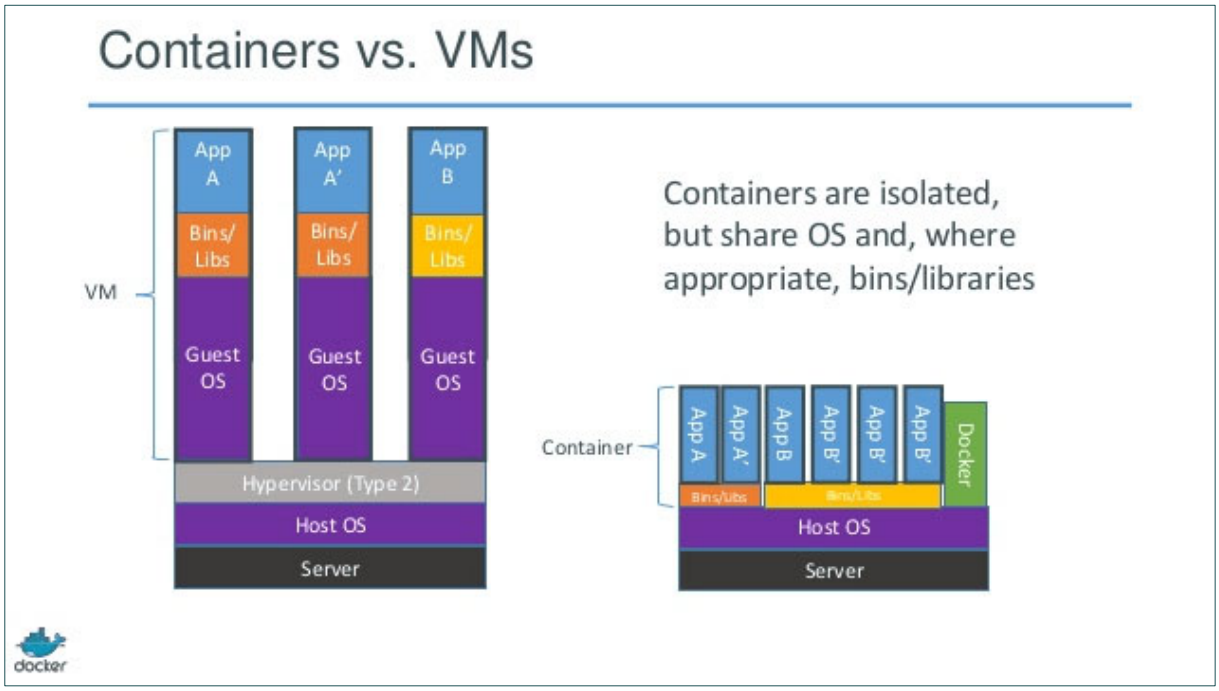

Figure 20: Docker Containers vs Virtual Machines [40]

Docker

Docker is a world-leading CaaS (Container-as-a-Service platform). It is fully open source, under license Apache 2.0 and is the most dominant tool in the container ecosystem at the moment, used in production stage by companies like eBay, Uber and PayPal. The container is a way to package software along with required binaries and settings, isolated on a shared operating system.

Docker makes it easier to create and manage application deployment and release. Once your services have been dockerised in containers, you can deploy those containers to any server with Docker installed and combine them to compose complex applications. You can move them around between hosts for portability.

The key steps involved are the following:

1. Package each Platone Platform as one or more (Docker) container images;
2. Deploy each platform as a container;
3. Run the containerized platform in the demos' infrastructures.

7.3 Deployment Diagrams

The Figure 21 represents the deployment view of the cloud infrastructure. The cloud infrastructure is usable as-a-Service thanks to the API Gateway component that allows the integration of external components and end-user applications. Furthermore, the Integration Layer allows the integration of data coming from the fields in two ways: via MQTT protocol for IoT infrastructure or via TCP/IP protocol for Data server implementation.

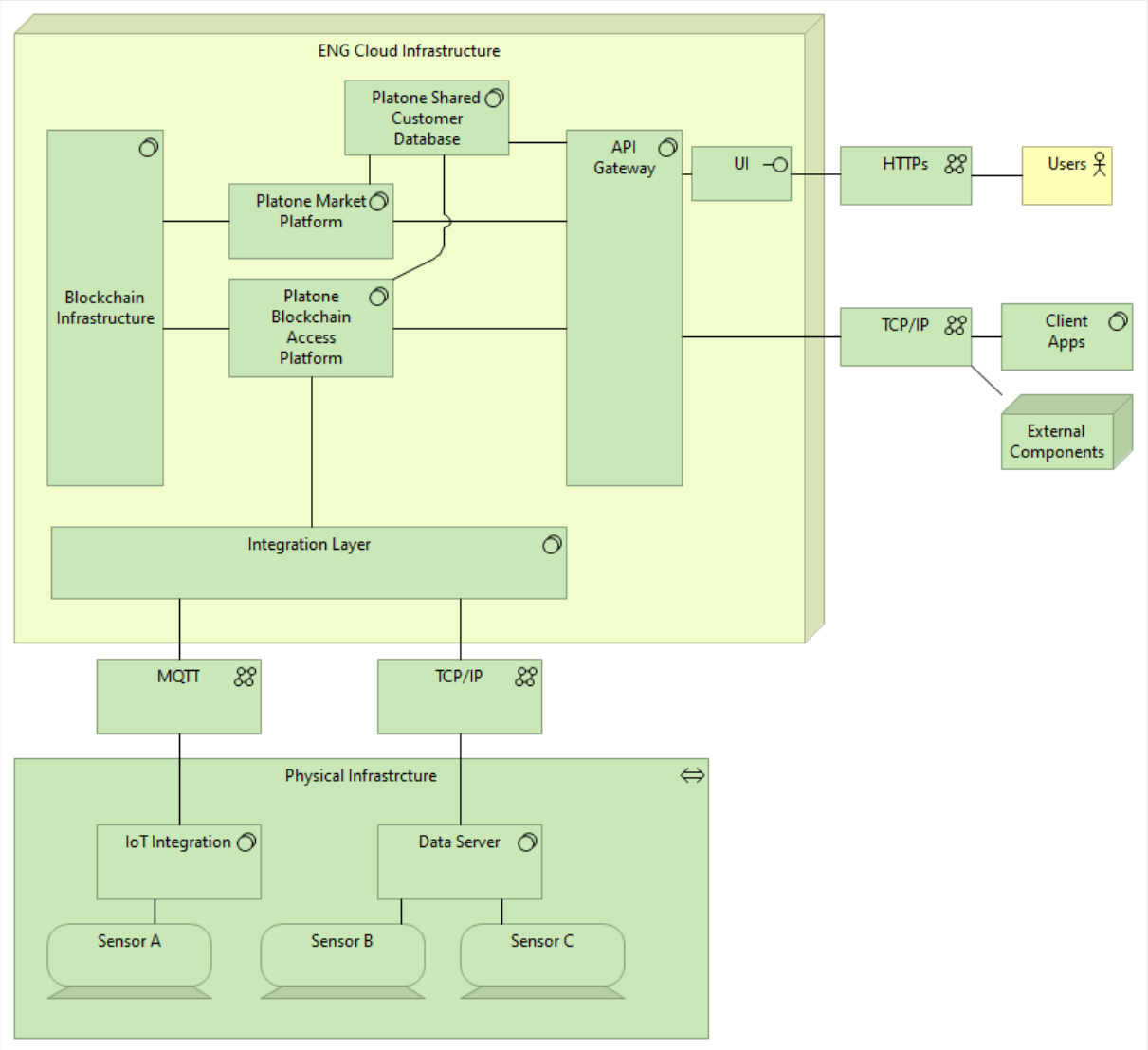

Figure 21: Deployment View - Cloud Infrastructure

8 Requirements on the Platone Platforms

This chapter includes in Table 27 all the functional and non-functional requirements expected for the Platone platforms developed within WP2. The information regarding the other components and services expected in WP3 WP4 and WP5 will be described in the respective deliverable by the responsible WPs.

As already mentioned in the Methodology paragraph, Use Cases, Scenarios and Information flows have been used as the input for defining the list of functional and non-functional requirements for Platone Platforms. All these input information could be found more in detail in D1.1 [2], D4.1 [3] and D5.2 [4].

This list of requirements represents the baseline for the implementation of the three version of the Platone Platforms that will be respectively released in M18 (February 2021), M38 (October 2023) and M46 (June 2024).

The integrated prototype of the Platone Open Framework will include all the Platone Platforms and will be evaluated during the three different demos in Italy, Greece and Germany.

As already mentioned, an update of all these requirements, especially non-functional one, which at the moment are not yet all well defined, due to the lack of some information, is expected in M30 (February 2022), after the first validation of the prototype on the field.

Table 27: Platone Platforms Requirements

Requirement ID	Requirement name	Requirement description	Use Cases
Market Platform – Functional Requirements			
FR-MP-FSM-01	Flexibility Services Management	The Market Platform allows DSOs and TSOs to create flexibility requests in automatic way	UC-IT-1 UC-IT-2
FR-MP-FSM-02	Flexibility Services Management	The Market Platform allows DSOs to create flexibility requests through UI	UC-IT-1 UC-IT-2
FR-MP-FSM-03	Flexibility Services Management	The Market Platform allows Aggregator Platform to create flexibility offers in automatic way	UC-IT-1 UC-IT-2
FR-MP-FSM-04	Flexibility Services Management	The Market Platform acquires and stores all the flexibility requests and offers	UC-IT-1 UC-IT-2
FR-MP-MOMV-01	Market Outcomes Matching and Validation	The Market Platform is able to match flexibility requests and offers through clearing market algorithms	UC-IT-1 UC-IT-2
FR-MP-MOMV-02	Market Outcomes Matching and Validation	The Market Platform is able to provide the Market Outcomes (results of market clearing) to the DSO Technical Platform for the technical validation	UC-IT-1 UC-IT-2
FR-MP-MOMV-03	Market Outcomes Matching and Validation	The Market Platform receives the validated market outcomes from DSO Technical Platform	UC-IT-1 UC-IT-2

FR-MP-MOMV-04	Market Outcomes Matching and Validation	DSOs, TSOs and Aggregators receives Market Day Ahead outcomes from the Market Platform	UC-IT-1 UC-IT-2
FR-MP-SA-01	Services activation	The Market Platform allows to DSOs and TSOs to create service activation requests in automatic way	UC-IT-1 UC-IT-2
FR-MP-SA-02	Services activation	The Market Platform allows to Market participant to create service activation requests through UI	UC-IT-1 UC-IT-2
FR-MP-SA-03	Services activation	The Market Platform is able to aggregate the service activation requests (from DSOs and TSOs) and provide them to all the other stakeholders	UC-IT-1 UC-IT-2
FR-MP-BC-01	Blockchain certification	The Market Platform is able to register on the blockchain all the market data trough Smart Contracts based functionalities	UC-IT-1 UC-IT-2
FR-MP-BC-02	Blockchain certification	The Market Platform allows to Market participant to verify all the market data registered in the blockchain	UC-IT-1 UC-IT-2
FR-MP-S-01	Settlement	The Market Platform is able to read meters measurements from SCD	UC-IT-1 UC-IT-2
FR-MP-S-02	Settlement	The Market Platform performs the settlement comparing the metering data and BRP baseline	UC-IT-1 UC-IT-2
FR-MP-S-03	Settlement	The Blockchain Service Layer is able to provide tokenization system for the settlement through Smart Contracts functionalities	UC-IT-1 UC-IT-2
FR-MP-S-04	Settlement	The Market Platform allows to DSO, TSO and Aggregator to read the settlement outcomes	UC-IT-1 UC-IT-2
Market Platform – Non-Functional Requirements			
P-MP-01	Communication protocols	The Market Platform exposes REST APIs for collecting flexibility requests and flexibility offers	UC-IT-1 UC-IT-2
P-MP-02	Communication protocols	The Market Platform provides a message broker for communicating market results	UC-IT-1 UC-IT-2
DSOTP - Functional Requirements			

FR-DSOTP-DA-01	Data Acquisition	The DSOTP is able to receive Measurements that reflect the network state from DSO Data Server	UC-GR-1 UC-GR-2 UC-GR-3 UC-GR-4 UC-GR-5
FR-DSOTP-DA-02	Data Acquisition	The DSOTP is able to receive data coming from State Estimation Tool	UC-GR-1 UC-GR-2 UC-GR-3 UC-GR-4 UC-GR-5
FR-DSOTP-DA-03	Data Acquisition	The DSOTP is able to receive PMU measurements that reflect the network state	UC-GR-2 UC-GR-3 UC-GR-4 UC-GR-5
FR-DSO-TP-DA-04	Data Acquisition	The DSOTP is able to receive certified measurement from BAP	UC-GR-1 UC-GR-2 UC-GR-3 UC-GR-4 UC-GR-5 UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
FR-DSO-TP-DA-05	Data Acquisition	The DSOTP is able to receive setpoints from EMS	UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
FR-DSOTP-SE-01	State Estimation	The DSOTP is able to trigger the State Estimation Tool via REST API.	UC-GR-1 UC-GR-2 UC-GR-3 UC-GR-4
FR-DSOTP-SE-02	State Estimation	The DSOTP provides the results of State Estimation as estimated state vector to DSO	UC-GR-1 UC-GR-2 UC-GR-3 UC-GR-4 UC-GR-5
FR-DSOTP-PMU-01	PMU Data Integration	The DSOTP is able to integrate PMU and conventional measurements into a unified measurement set for performing observability assessment via the State Estimation Tool.	UC-GR-2 UC-GR-3 UC-GR-4
FR-DSOTP-T-01	Tariffs retrieval	The DSOTP sends to the DSO/Aggregators tariffs that reflect the expected state of the network	UC-GR-3 UC-GR-4
FR-DSOTP-T-02	Tariffs retrieval	The DSOTP is able to receive data coming from the Algorithm for DER	UC-GR-3 UC-GR-4

		Control and Algorithm for ancillary services	
FR-DSOTP-DER-01	Optimal DER dispatching	DSOTP is able to trigger the Algorithm for DER Control via REST API	UC-GR-3
DSOTP – Non-Functional Requirements			
P-DSOTP-01	Communication protocols	DSOTP is able to receive data from PMUs via MQTT protocol	UC-GR-2 UC-GR-3 UC-GR-4 UC-GR-5
P-DSOTP-02	Communication protocols	DSOTP is able to receive data from DSO Data Server via TCP/IP protocol	UC-GR-1 UC-GR-2 UC-GR-3 UC-GR-4 UC-GR-5
P-DSOTP-03	Communication protocols	DSOTP is able to receive setpoints from A-LFC via TCP/IP protocol	UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
T-DSOTP-01	Timing	DSOTP is able to receive measurement every 10 seconds from sensors	UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
T-DSOTP-02	Timing	DSOTP is able to receive measurement every 15 minutes from Data Management Backend	UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
T-DSOTP-03	Timing	DSOTP is able to receive setpoints every 10 seconds for BESS and every 15 minutes for flexible loads and storages	UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
BAP – Functional Requirements			
FR-BAP-DM-01	Blockchain Data Management	The BAP is able to acquire Measurements from network	UC-GR-1 UC-GR-2 UC-GR-3 UC-GR-4 UC-GR-5 UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
FR-BAP-DM-02	Blockchain Data Management	The BAP certifies Measurements via Smart Contracts	UC-GR-1 UC-GR-2

			UC-GR-3 UC-GR-4 UC-GR-5 UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
FR-BAP-DM-03	Blockchain Data Management	The BAP provides certified measurement in a secure way to DSOTP	UC-GR-1 UC-GR-2 UC-GR-3 UC-GR-4 UC-GR-5 UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
FR-BAP-NC-01	Network Control	The BAP is able to receive set points from DSOTP	UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
FR-BAP-NC-02	Network Control	The BAP certifies set points via Smart Contracts	UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
FR-BAP-NC-03	Network Control	The BAP is able to send certified set points to Data Management Backend	UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
BAP – Non-Functional Requirements			
P-BAP-01	Communication protocols	The BAP is able to receive data from sensors via MQTT protocol	UC-GR-2 UC-GR-3 UC-GR-4 UC-GR-5 UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
P-BAP-02	Communication protocols	The BAP is able to integrate data coming from external server via TCP/IP protocol	UC-GR-1 UC-GR-2 UC-GR-3 UC-GR-4 UC-GR-5 UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4
T-BAP-01	Timing	BAP is able to receive measurement every 10 seconds from sensors	UC-GE-1 UC-GE-2

			UC-GE-3 UC-GE-4
T-BAP-02	Timing	BAP is able to receive measurement every 15 minutes from Data Management Backend	UC-GE-1 UC-GE-2 UC-GE-3 UC-GE-4

9 Conclusion

The work done at this stage, allowed us to have a complete overview of the characteristics of the Platone Open Framework and how it can be exploited by integrating it into existing architectures or used in its entirety.

The use of standard methodologies, such as the 4+1 architectural views and IEC 62559, has allowed us to define the characteristics of the Platone Reference architecture and the functional and non-functional requirements for all Platone Platforms in a standardized way.

The results of this work will be the base for the implementation of the Platone Platforms and the integration of the Platone Open Framework within the demo sites architecture. Each demo site will integrate different parts of the framework, test and evaluate them. The iterative implementation approach will allow us to learn from each iteration, enriching and improving the framework throughout the project.

Furthermore, the open source approach will ensure a better outreach and re-use of the results as well as an increment of the impact of the Platone project on the scientific community and in particular on the energy stakeholders.

10 List of Tables

11 List of Figures

12 List of References

[1] European Commission, “2050 long-term strategy,” 2018. [Online]. Available: https://ec.europa.eu/clima/policies/strategies/2050_en.

[2] Platone, D1.1 General Functional Requirements and specifications of joint activities in the Demonstrators, 2020.

[3] Platone, D4.1 Report on the definitions of KPIs and UCs, 2020.

[4] Platone, D5.2 Detailed Use Case Descriptions, 2020.

[5] Platone, Description of Work (DoW), 2019.

[6] P. Kruchten, “The “4+1” View Model of Software Architecture,” in *Architectural Blueprints*, IEEE Software 12, 1995, November, pp. 42-50.

[7] “ArchiMate - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/ArchiMate.

[8] “Archimate Tool,” [Online]. Available: https://www.archimatetool.com/.

[9] V. Muchandi, “Applying 4+1 View Architecture with UML2,” [Online]. Available: www.sparxsystems.com/downloads/whitepapers/FCGSS_US_WP_Applying_4+1_w_UML2.pdf.

[10] J. S., “Blockchain: What are nodes and masternodes?,” 2018. [Online]. Available: https://medium.com/coinmonks/blockchain-what-is-a-node-or-masternode-and-what-does-it-do-4d9a4200938f.

[11] A. Rosic, “Smart Contracts: The Blockchain Technology That Will Replace Lawyers,” [Online]. Available: https://blockgeeks.com/guides/smart-contracts/.

[12] A. Rosic, “Blockchain Consensus: A Simple Explanation Anyone Can Understand,” [Online]. Available: https://blockgeeks.com/guides/blockchain-consensus/.

[13] “Bitcoin - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Bitcoin.

[14] “Byzantine Fault - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Byzantine_fault.

[15] “Proof-of-Work - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Proof_of_work#Bitcoin-type_proof-of-work.

[16] “Different Blockchain Consensus Mechanisms,” 10 November 2018. [Online]. Available: https://hackernoon.com/different-blockchain-consensus-mechanisms-d19ea6c3bcd6.

[17] “Ethereum Whitepaper,” July 2019. [Online]. Available: https://ethereum.org/en/whitepaper/.

[18] “Blockchain in Energy and Sustainability,” [Online]. Available: https://consensys.net/blockchain-use-cases/energy-and-sustainability/.

[19] “Blockchain for Energy 2018: Companies & Applications for Distributed Ledger Technologies on the Grid,” 5 March 2018. [Online]. Available: https://www.woodmac.com/reports/power-markets-blockchain-for-energy-2018-companies-and-applications-for-distributed-ledger-technologies-on-the-grid-58115325.

[20] Bosco, Croce and Raveduto, “Blockchain technology for financial services facilitation in RES investments,” Palermo, 2019.

[21] M. Pustišek, A. Kos and U. Sedlar, “Blockchain-based Autonomous Selection of Electric Vehicle Charging,” *2016 International Conference on Identification, Information and Knowledge in the Internet of Things,* pp. 217-222, 2016.

[22] M. Andoni, V. Robu, D. Flynn, S. Abram, D. Geach, D. Jenkins, P. McCallum and A. Peacock, “Blockchain technology in the energy sector: A systematic review of challenges and opportunities,” *Renewable and Sustainable Energy Reviews,* vol. 100, pp. 143-174, February 2019.

[23] “H2020 eDREAM Project,” [Online]. Available: https://edream-h2020.eu/.

[24] “ERC 20 specification - Github,” [Online]. Available: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md.

[25] “Fungibility - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Fungibility.

[26] “ERC 721 specifications - Github,” [Online]. Available: https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md.

[27] “Interledger Protocol V4,” [Online]. Available: https://interledger.org/rfcs/0027-interledger-protocol-4/draft-6.html.

[28] “SOFIE Interledger component specifications - Github,” [Online]. Available: https://github.com/SOFIE-project/Interledger.

[29] “H2020 SOFIE Project,” [Online]. Available: https://www.sofie-iot.eu/.

[30] “H2020 SOGNO Project,” [Online]. Available: https://www.sogno-energy.eu/.

[31] “SOGNO D4.1 - Definition of overall SOGNO System Architecture,” 31 October 2018. [Online]. Available: https://www.sogno-energy.eu/global/images/cms/Deliverables/774613_deliverable_D4.1.pdf.

[32] “Grafana,” [Online]. Available: https://grafana.com/.

[33] “Kubernetes (K8s),” [Online]. Available: https://kubernetes.io/.

[34] “Apache Cassandra,” [Online]. Available: https://cassandra.apache.org/.

[35] “BigChainDB,” [Online]. Available: https://www.bigchaindb.com/.

[36] C. Pop, M. Antal, T. Cioara, I. Anghel, D. Sera, I. Salomie, G. Raveduto, D. Ziu, V. Croce and M. Bertoncini, “Blockchain-Based Scalable and Tamper-Evident Solution for Registering Energy Data,” *Sensors,* vol. 19, no. 14, 2019.

[37] “Open API specification v3.0.3,” 20 February 2020. [Online]. Available: https://swagger.io/specification/.

[38] “Iterative and Incremental Development - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Iterative_and_incremental_development.

[39] “Container definition,” [Online]. Available: https://techterms.com/definition/container.

[40] “Container Vs Process,” [Online]. Available: https://sites.google.com/site/mytechnicalcollection/cloud-computing/docker/container-vs-process.

13 List of Abbreviations

Abbreviation	Term
ALF-C	Avacon Local Flex-Controller
AMR	Automatic Meter Reading system
API	Application programming interface
BAL	Blockchain Access Layer
BAP	Blockchain Access Platform
BESS	Battery Energy Storage System
BMS	Battery Management System
BPMN	Business Process Model and Notation
CaaS	Container-as-a-Service
CIM	Common Information Model
CPU	Central Processing Unit
DEMI	Distributed Energy Management Initiative
DER	Demand Energy Response
DLT	Distributed Ledger Technology
DMS	Distribution Management System
DoW	Description of Work
DSO	Distribution System Operator
DSOTP	DSO Technical Platform
EMS	Energy Management System
ENG	Engineering Ingegneria Informatica S.p.a.
EV	Electric Vehicle
FAQ	Frequently Asked Questions
FR	Flexible Resource
GIS	Geographic information system
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HSPA	High Speed Packet Access
HTTP	Hypertext Transfer Protocol
HTTPs	Hypertext Transfer Protocol over SSL
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
ILP	Interledger Protocol
IoT	Internet of Things
IT	Information Technology
JSON	JavaScript Object Notation

KPI	Key Performance Indicator
LAN	Local Area Network
LTE	Long Term Evolution
LV	Low Voltage
MQTT	Message Queue Telemetry Transport
MV	Medium Voltage
NSGA	Non-Dominated Sorting Genetic Algorithm
OAS	Open API Specification
OS	Operating System
OT	Operational Technology
P2P	Peer to Peer
PLC	Programmable Logic Controller
PMU	Phasor Measurement Unit
PoD	Point of Delivery
PoW	Proof-of-Work
RAM	Random Access Memory
RES	Renewable Energy Sources
REST	Representational State Transfer
RTU	Remote Terminal Unit
SaaS	Software-as-a-Service
SCADA	Supervisory Control And Data Acquisition
SCD	Shared Customer Database
SE	State Estimation
SGAM	Smart Grid Architecture Model
SoC	State of Charge
SoE	State of Energy
SQL	Structured Query Language
SSD	Solid State Disk
TBD	To be decided
TCP/IP	Transmission Control Protocol / Internet Protocol
TSO	Transmission System Operator
UC	Use Case
UI	User Interface
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications System
WP	Work Package

Annex A Platone Components Detailed Specifications Templates

A.1 Platone Architectural Components Detailed Specifications Template

Table 28: Platone Architectural Components Detailed Specifications Template

Name of Component/Service:	<please write here the name of the architectural element e.g. Market Platform (Component) or State Estimation Tool (Service)>
Type	<*Component* or *Service*>
Functionality	<please write here in free text a short description of the operation of this module/component. A list of functions and operations will be a plus>
Input Connections & Interfaces	<please write the components from which it receives input (input dependencies) and mention if possible the available connection interfaces e.g. API etc.>
Output Connections & Interfaces	<please write the components to which it sends the results (output dependencies) and mention also the available interfaces e.g. API etc.>
Software Requirements/Development Language	<specify any software requirements related to the architectural element, explain the Programming Language that is used during the development of the component>
Hardware Requirements	<specify what hardware requirements are of the module, give specifications about the hardware requirements which are necessary for the best functionality of the component>
Status of the development of the component	<specify if the component is “already developed” or “partially developed” or “to be developed from scratch”>
WP and Task reference	<specify in which WP and specific task the component is developed>

A.2 Platone Hardware Components Detailed Specifications Template

Table 29: Platone Hardware Components Detailed Specifications Template

Name of Device	<name of the device>
Description	<provide a brief statement of the device>
Functionality	<describe how the device functions within the PlatOne workflow>
Measurement	<description of the measurement>
Measurement Range	<minimum to maximum values that can be measured by

	the device (e.g. -40 to +80 °C)>
Measurement Resolution	<level of measurement (e.g. to 0.01 °C)>
Accuracy	<accuracy of the measurement (e.g. ±x% of actual reading)>
Data Connections	<specify the communication networks and protocols involved e.g. USB, GSM, WiFi, Bluetooth etc.>
Data Output Format	<specify the output format of the sensor>
Data Rate	<specify at what rate data is read/extracted/logged>
Data Availability	<specify whether data stream is continuous, periodic, on demand etc.>