Platone

PLATFORM FOR OPERATION
OF DISTRIBUTION NETWORKS

Platone

PLATform for Operation of distribution NEtworks

D2.3

Platone Market Platform (v1)

Project name	Platone
Contractual delivery date:	28.02.2021
Actual delivery date:	28.02.2021
Main responsible:	Ferdinando Bosco (ENG)
Work package:	WP2 – Platform Implementation and Data Handling
Security:	P
Nature:	DEM
Version:	V1.0
Total number of pages:	32

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 accompanies the software delivery of the Platone Market Platform and extends it with an architecture overview and the explanation of a demonstration setup.

The Platone Market Platform is part of the Platone Open Framework that will be integrated, tested and evaluated in three different demo sites in: Greece, Germany and Italy. Each of these demo sites will integrate different parts of the framework.

In particular, the first prototype of the Platone Market Platform will be integrated, tested and evaluated in the Italian Demo Site Architecture.

Keyword list

Platone Market Platform, Flexibility Market, Blockchain Service Layer, Smart Contracts

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.

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 Market Platform** is one of the core components of the Platone Open Framework. It 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, thus 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.

Furthermore, the Platone Market Platform enables an innovative incentivisation mechanism for customer engagement based on blockchain technology, smart contracts, and tokenisation.

The first prototype of the Platone Market Platform focuses on the management of the Flexibility Day Ahead Market, defining all the necessary data models and implementing all the core services and components.

More in detail, it implements an API Gateway for collecting flexibility offers and requests coming from DSOs, TSOs and Aggregators defining a specific data models that allows to match flexibilities with high granularity. It also implements a clearing-matching algorithm based on price prioritization and blockchain-based services for market validation and settlement.

The Platone Market Platform Communication layer includes the API Gateway and the Message Broker for the integration of external services and platforms (e.g., DSO Technical Platform)

Furthermore, a first version of the Market Web Dashboard is released for allowing to the administrator to monitoring all the market activities.

Authors and Reviewers

Main responsible		
Partner	**Name**	**E-mail**
ENG		
	Ferdinando Bosco	ferdinando.bosco@eng.it
Author(s)/contributor(s)		
Partner	**Name**	
ENG		
	Angelo Triveri	
Reviewer(s)		
Partner	**Name**	
HEDNO		
	Eleni Daridou Stavroula Tzioka	
NTUA		
	Panagiotis Pediaditis	
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 acquisition system based on a two-layer approach (an access layer for customers and a distribution system operator (DSO) observability layer) that will allow greater stakeholder involvement and will enable an efficient and smart network management. The tools used for this purpose will be based on platforms able to receive data from different sources, such as weather forecasting systems or distributed smart devices spread all over the urban area. These platforms, by talking to each other and exchanging data, will allow collecting and elaborating information useful for DSOs, transmission system operators (TSOs), 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 blockchain architecture named Platone Open Framework that consists of a series of core components, including the Platone Market Platform.

The main goal of the Platone Market Platform is to enable a secure and transparent Flexibility Market, exploiting blockchain technology and smart contracts, for handling the management of flexibility services, providing market results to all the stakeholders, validating the flexibility provisioning, and performing the settlement outcome with an innovative incentivisation mechanism for improving customer engagement.

The Platone Market Platform will be implemented following the specifications and requirements gathered in the first phase of the project and will be delivered in three different incremental versions.

1.1 Task 2.2

This deliverable is related to the Task 2.2 [2] that aims at the implementation of the Platone Market Platform, following the functional and non-functional requirements defined in D2.1 [3].

1.2 Objectives of the Work Reported in this Deliverable

The objective of this deliverable is to present the first prototype of the Platone Market Platform and its realization following the technical specification and requirements expected. The Platone Description of Action defines this deliverable as a demonstrator. This document accompanies the code repository with a more detailed architecture description as well as some extended deployment instructions for deploying, testing and integrating the platform.

1.3 Outline of the Deliverable

The second Chapter of this document describes the first realization of the Platone Market Platform according to the specification provided in Deliverable D2.1 [3] and discusses the functionalities implemented in this first version more in detail. Chapter 3 provides a brief overview of Interfaces and Communication Mechanisms. Chapter 4 delivers a compilation of Languages, Technologies and External Tools used throughout the platform. Chapter 5 is closely linked to the software delivery and provides detailed installation, setup, and configuration instructions. Finally, Chapter 6 concludes this deliverable.

1.4 How to Read this Document

The document aims to give an overview to the Platone Market Platform prototype release. A description of the foreseen functional and non-functional requirements expected can be found in D2.1 [3].

2 Platform Architecture

2.1 Overview

The Platone Market Platform consists of a three-layer architecture:

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

The **blockchain service layer** consists of a blockchain infrastructure, based on Ethereum blockchain nodes, which enables the deployment of Smart Contracts.

The Market Platform architecture is shown in Figure 1.

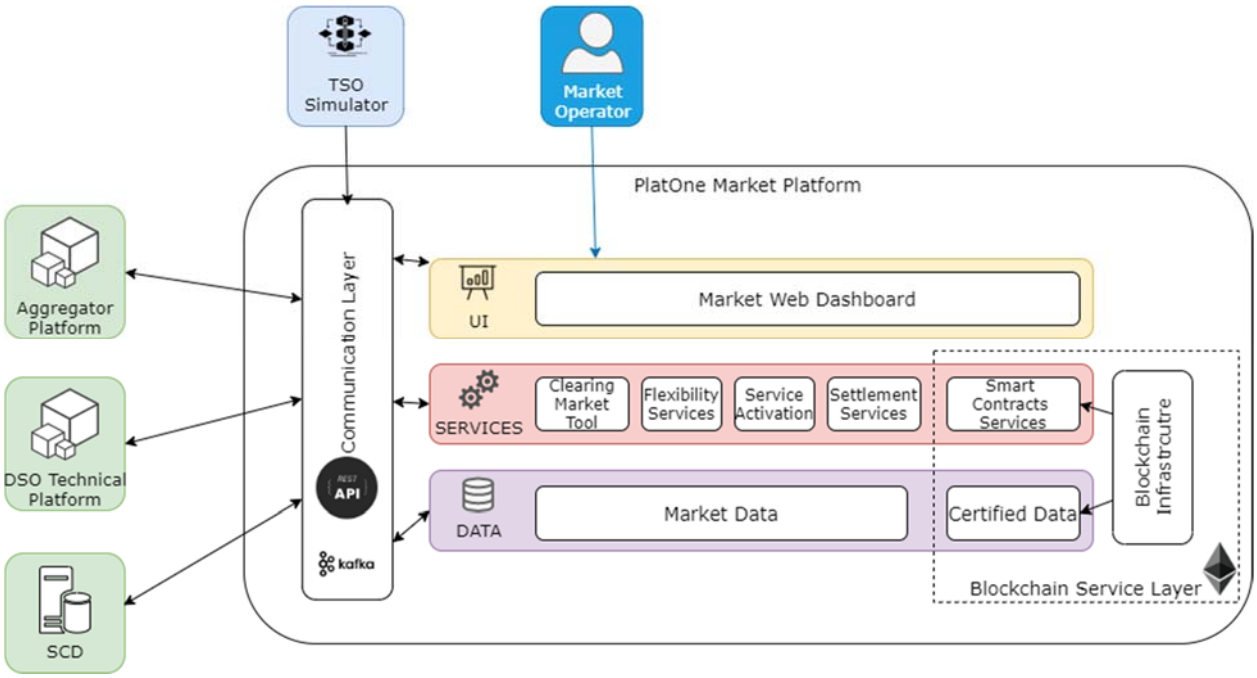

Figure 1: Market Platform Architecture

UI Layer

The UI Layer includes the User Interfaces available for all the market participants. In particular, it includes a Web Dashboard, accessible by DSOs, TSOs and Aggregators for monitoring their activities.

Services Layer

The Services Layer is the core of the Market Platform and it includes all the services that implement the functionalities offered by the Market Platform.

Data Layer

The Data Layer manages all the data necessary for the implementation of Market Platform services and it includes a NoSQL database for the storage of the Application Data. Furthermore, this layer implements the business logic for the certification of the market data on the blockchain service layer.

Communication Layer

The Market Platform architecture includes a Communication Layer, a specific component that provides two different communication mechanisms: synchronous and asynchronous. A specific architectural component dedicated to communication mechanisms, provides a greater flexibility to the Market Platform, which is able to cover different solution and integrate different external systems.

More in detail, the synchronous communication is implemented in the API Gateway via REST APIs. The API gateway is the entry point for every HTTP request that is launched by the external systems. This central component, shared by the whole Market Platform, allows some middleware functionalities to be centralised, i.e.:

- Authentication
- Logging
- Caching
- Security
- Load Balancing

The asynchronous communication is implemented in the Message Broker. A message broker (or queue manager) is a software where queues can be defined, applications may connect to the queue and transfer a message onto it.

A message can include any kind of information. For example, it could include information about a process/task that should start on another application (that could be on another server), or it could be just a simple text message. The queue-manager software stores the messages until a receiving application connects and takes a message off the queue. The receiving application then processes the message in an appropriate manner.

A message broker can act as a middleware for various services (e.g. different external systems). They can be used to reduce loads and delivery times by web application servers since tasks, which would normally take quite a bit of time to process, can be delegated to a third party whose only job is to perform them.

Message queueing allows web servers to respond to requests quickly instead of being forced to perform resource-heavy procedures on the spot. Message queueing is also good when you want to distribute a message to multiple recipients for consumption or for balancing loads between workers.

Blockchain Service Layer

The blockchain service layer is based on a blockchain infrastructure that includes 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 “tokenization” of 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 [4] 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 these policies.

TSO Simulator

The TSO simulator is an external tool that simulates possible flexibility requests coming from TSOs for solving possible congestion issues. This tool was developed by ENG only for simulation purposes and it is an external component of the Platone Market Platform.

2.2 Functionalities

The Market Platform implements a series of functionalities in the Service Layer, some of these based on blockchain technology. The services implemented in the service layer are:

- Flexibility Services
- Clearing Market Tool
- Settlement Services

Flexibility Services

The Market Platform is a “virtual” place in which Market Players can participate to the flexibility market, in different market sessions, such as day-ahead and real time markets.

Before starting a new market session, the Market Platform receives the configuration of the network, including PoD information. This information is fundamental to perform all the processes within the Market Platform.

During an active market Session, the Market Platform is able to receive requests and offers for flexibility services by the Market Participants (DSO and TSO for flexibility request, Aggregator for flexibility offers).

At the end of the market session, the Market Platform performs the economic phase of the Market Clearing, matching the DSO’s and TSO’s request with the Aggregator’s offers.

Clearing Market Tool

The clearing market tool is highly configurable, and its main goal is to find among the various offers of the aggregators, those that meet the DSO and TSO requests. All the offers that accomplish 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 use cases (e.g., in the Italian demo, DSO and price prioritisation).

At the end of the clearing phase, the Market platform produces a Market Outcome, which could be provided to the DSO platform, for any technical validation.

The results of Market Validation and Technical Validation (if needed) is the Validated Market Outcome which is shared with all the market participants.

In the Real Time (RT) Market, the collection of flexibility requests and offers is identical with respect to the Day Ahead (DA) market. During the clearing step, the Market Platform considers both the results of the RT session and those of the DA session for the following 4 hours.

Settlement Services

After the activation of the flexibilities, as agreed during the market session, the Market Platform has to create the Settlement outcomes, and communicates them to DSO, TSO and Aggregator.

In order to perform the settlement, the Market Platform must know the remuneration mechanisms for each Point of Delivery (PoD). Those information are provided by the Aggregator (a) and registered on the blockchain through specific smart contracts. Each smart contract can be associated with a single PoD or a cluster of PoDs in base of their type.

At the end of the settlement phase, DSO (or TSO if the related service was requested from them) receives all the necessary information that allows them to pay for the received flexibility service.

In the blockchain ecosystem, the Market Platform is able to “tokenize” the settlement outcomes enabling a token-based remuneration process that the Aggregator can exploit for performing the settlement of the flexibility resources under its jurisdiction, incentivising the customer engagement.

Blockchain-based Services

The services implemented using Smart Contract and Blockchain technology are:

- Tracking and controlling the registration and validation of energy data and market data;

- Publishing bid/offer actions by Market Participants;
- Settlement certification;
- Token-based End-Customer incentivisation.

2.2.1 Details of the first prototype

The first prototype of the Platone Market Platform includes a subset of all the functionalities expected in the functional requirements.

In particular:

- Only Day-Ahead Market Flexibility Services (no intra-day market);
- Clearing Market Tool based on price priority (no other optimisation mechanisms included),
- Settlement Outcomes and validation (feature completed);
- Blockchain and Smart Contract services for Settlement and Customer Incentivisation (no Market Data certification),
- Basic version of Web Dashboard for Market Participants (extended version expected for the 2nd prototype).

2.3 Data Models

The data models follow the Open API specification [5]

User

Table 1: User Model

Field	Type	Description
username	String	**Required**
password	String	**Required**
role	String	DSO, TSO, aggregator, admin

PoD

Table 2: PoD model

Field	Type	Description
podId	String	**Required,** pod identification string
aggregatorId	String	**Required,** aggregator identification string
zone	String	PoD geographic area
pomId	String	PoM identification string
pnp	Number	Active Power Withdrawn in kW
inp	Number	Active Power Feed in kW
inq	Number	Inductive Reactive Power in kVar
cnq	Number	Capacitive Reactive Power in kVar

sendTime	Date	Sending data Timestamp- format ISO-8601: YYYY-MM-DDThh:mm:ss[.mmm]TZD
validityStart	Date	Validity Date - format ISO-8601: YYYY-MM-DDThh:mm:ss[.mmm]TZD
flexibilityType	String	Type of flexibility offered [“continue”, “discrete”]
actionType	String	Type of allowed action [“activate”,”modify”,”delete”]
smartContractId	String	ID of the associated smart contract for settlement mechanism
powerCurvesDuration	Number	Curve duration in hours (**default** 24)
powerCurveInterval	Number	Curve interval in minutes (**default** 15)
powerBaselineCurves	**PowerBaselineCurvesObject**	Typical POD power baseline
maxFlexibility	**MaxFlexibilityObject**	Maximum flexibility

PowerBaselineCurvesObject

Table 3: Power Baseline Curves Object Model

Field	Type	Description
Workday	[{index,p,q}]	Typical power baseline in a workday *index*: [from 0 to 95] is the timeslot of the day *p*: active power in kW *q*: reactive power in kVar
Saturday	[{index,p,q}]	Typical power baseline on Saturday *index*: [from 0 to 95] is the timeslot of the day *p*: active power in kW *q*: reactive power in kVAr
Sunday	[{index,p,q}]	Typical power baseline on Sunday *index*: [from 0 to 95] is the timeslot of the day *p*: active power in kW *q*: reactive power in kVar

MaxFlexibilityObject

Table 4: Max Flexibility Object Model

Field	Type	Description
upperP	Number	Maximum flexibility up for active power in kW
downP	Number	Maximum flexibility down for active power in kW
upperQ	Number	Maximum flexibility up for reactive power in kVar
downQ	Number	Maximum flexibility down for reactive power in kVar

MarketSession

Table 5: Market Session Model

Field	Type	Description
marketType	String	**Required** Market Type : “dayAhead” || realTime”
Status	String	**Required** Status of the session: “created”, “active”, “closed”
Start	Date	Starting date of the session - format ISO-8601: YYYY-MM-DDThh:mm:ss[.mmm]TZD
End	Date	Closing date of the session - format ISO-8601: YYYY-MM-DDThh:mm:ss[.mmm]TZD

FlexibilityService

Table 6: Flexibility Service Model

Field	Type	Description
playerId	String	**Required** Id of the Market Player (DSO, TSO or Aggregator)
startTime	Date	Starting date of the market timeframe - format ISO-8601: YYYY-MM-DDThh:mm:ss[.mmm]TZD
duration	Number	Duration of the timeframe (in hours, e.g. 24) **default 24**
interval	Number	Duration of the interval (in minutes, e.g. 15) **default 15**
serviceType	String	Service Type: “offer” || “DSO_request” || “TSO_request”

marketType	String	**Required** Market Type : “dayAhead” || realTime”
marketSession	String	Id of the associated market session (used internally)
playerServiceId	String	**Required** Unique Id for each flexibility offer provided by the aggregator
flexibility	[**FlexibilityObject**]	**Required** Object that describes the flexibility offer/request

FlexibilityObject

Table 7: Flexibility Object Model

Field	Type	Description
pod	String	**Required** Id of the PoD
realTime	Boolean	It is true if the flexibility offer can be re-submitted in the real time market in case it is not accepted in the day-ahead market
flexibileBlock	Object	It describes the flexible block offer
power	[**PowerObject**]	**Required** It describes the power request/offer

PowerObject

Table 8: Power Object Model

Field	Type	Description
Index	String	Index of the interval in the time frame (e.g. from 0 to 95 in day-ahead market)
p	Number	Active Power offered/requested (in kW)
pPrice	Number	Price for active power flexibility (€ per kW)
q	Number	Reactive Power offered/requested (in kVar)
qPrice	Number	Price for reactive power flexibility (€ per kVar)

MarketOutcome

Table 9: Market Outcome Model

Field	Type	Description
_id	String	The Market Outcome Id
startTime	Date	Starting date of the market timeframe - format ISO-8601: YYYY-MM-DDThh:mm:ss[.mmm]TZD
duration	Number	Duration of the timeframe (in hours, e.g. 24) **default 24**
interval	Number	Duration of the interval (in minutes, e.g. 15) **default 15**
serviceType	String	Service Type: “offer” || “DSO_request” || “TSO_request”
marketType	String	**Required** Market Type : “dayAhead” || realTime”
marketSession	String	**Required** Id of the associated market session (used internally)
flexibility	[**MatchedFlexibilityObject**]	**Required** Object that describes the matched flexibility

MatchedFlexibilityObject

Table 10: Matched Flexibility Object Model

Field	Type	Description
pod	String	**Required** Id of the PoD
power	[**MatchedPowerObject**]	**Required** It describes the power request/offer

MatchedPowerObject

Table 11: Matched Power Object Model

Field	Type	Description
Index	String	Index of the interval in the time frame (e.g. from 0 to 95 in day-ahead market)
p	Number	Active Power matched (in kW)
priorityP	Number	Index of priority for the matched active power (Max priority = 1)
q	Number	Reactive Power matched (in kVar)

priorityQ	Number	Index of priority for the matched reactive power (Max priority = 1)

TechnicalOutcome

Table 12: Technical Outcome Model

Field	Type	Description
startTime	Date	Starting date of the market timeframe - format ISO-8601: YYYY-MM-DDThh:mm:ss[.mmm]TZD
duration	Number	Duration of the timeframe (in hours, e.g. 24)
interval	Number	Duration of the interval (in minutes, e.g. 15)
marketType	String	Market Type : “dayAhead” || realTime”
marketOutcome	String	**Required** Unique identifier of related Market Outcome
flexibility	[**AcceptedFlexibilityObject**]	**Required** Object that describes the accepted flexibility from DSO

AcceptedFlexibilityObject

Table 13: Accepted Flexibility Object Model

Field	Type	Description
pod	String	**Required** Id of the PoD
power	[**AcceptedPowerObject**]	**Required** It describes the accepted power

AcceptedPowerObject

Table 14: Accepted Power Object Model

Field	Type	Description
Index	String	Index of the interval in the time frame (e.g. from 0 to 95 in day-ahead market)
acceptedPValue	Number	Active Power accepted (in kW) – Null if not accepted
acceptedQValue	Number	Reactive Power accepted (in kVar) - Null if not accepted
acceptedP	String	“OK” if accepted (totally or partially), “KO” if not accepted

acceptedQ	String	“OK” if accepted (totally or partially), “KO” if not accepted

ValidatedOutcome

Table 15: Validated Outcome Model

Field	Type	Description
startTime	Date	Starting date of the market timeframe - format ISO-8601: YYYY-MM-DDThh:mm:ss[.mmm]TZD
duration	Number	Duration of the timeframe (in hours, e.g. 24)
interval	Number	Duration of the interval (in minutes, e.g. 15)
serviceType	String	Service Type: “offer” || “DSO_request” || “TSO_request”
marketType	String	Market Type : “dayAhead” || realTime”
marketOutcome	String	**Required** Unique identifier of related Market Outcome
flexibility	[**ValidatedFlexibilityObject**]	**Required** Object that describes the validated flexibility

ValidatedFlexibilityObject

Table 16: Validated Flexibility Object Model

Field	Type	Description
pod	String	Id of the PoD
power	[**ValidatedPowerObject**]	It describes the validated power

ValidatedPowerObject

Table 17: Validated Power Object Model

Field	Type	Description
Index	String	Index of the interval in the time frame (e.g. from 0 to 95 in day-ahead market)
acceptedPValue	Number	Active Power accepted (in kW)
acceptedPPrice	Number	Price accepted for active power (€ per kW)

rejectionTypeP	String	If rejected is “TEC” for technical rejection or “COM” for Market rejection
acceptedQValue	Number	Reactive Power validated (in kVar)
acceptedQPrice	Number	Price accepted for reactive power (€ per kVar)
rejectionTypeQ	String	If rejected is “TEC” for technical rejection or “COM” for Market rejection
playerServiceId	String	The reference Id of the aggregator offer for this flexibility

Measurements

Table 18: Measurements Model

Field	Type	Description
aggregatorId	String	Id of the aggregator
pod	String	**Required** Id of the PoD
dataTime	Number	**Required** Interval of the measures - format ISO-8601: YYYY-MM-DDThh:mm:ss[.mmm]TZD
anomaly	String	Anomaly Code
measures	{**energy** }	**Required** Measurements of the PoD in the interval
setPoint	**SetPointObject**	**Required** Set point requested to PoD in the interval

energy

Table 19: energy Model

Field	Type	Description
absorbedActiveEnergy	{value, quality}	In kWh
InjectedActiveEnergy	{value, quality}	In kWh
absorbedInductiveReactiveEnergy	{value, quality}	In kVarh
injectedInductiveReactiveEnergy	{value, quality}	In kVarh
absorbedCapacitiveReactiveEnergy	{value, quality}	In kVarh
injectedCapacitiveReactiveEnergy	{value, quality}	In kVarh

power

Table 20: power Model

Field	Type	Description
p	Number	Measured active power in the interval (in kW)
q	Number	Measured reactive power in the interval (in kVar)

SetPointObject

Table 21: Set Point Object Model

Field	Type	Description
marketOutcomeId	String	Reference Id of the market Outcome
activePower	Number	Requested active power in the interval (in kW)
reactivePower	Number	Requested reactive power in the interval (in kVar)

SmartContract

Table 22: Smart Contract Model

Field	Type	Description
_id	String	Id of the Smart Contract
timeFrames	[TimeFrameObject]	**Required** Different Time Frames with settlement mechanisms

TimeFrameObject

Table 23: Time Frame Object Model

Field	Type	Description
dayOfTheWeek	String	[“Mon”, “Tue”, …]
fixedPrices	[{interval, price}]	Array of prices with interval (from 0 to 95) and price (in €/kWh). In alternative to percentagePrices
percentagePrices	[{interval, price}]	Array of prices with interval (from 0 to 95) and price (% of the flexibility price). Alternative to fixedPrices

Settlement

Table 24: Settlement Model

Field	Type	Description
marketOutcomeId	String	**Required** Id of the market outcome
marketType	String	Market Type : “dayAhead” || realTime”
flexibility	[**FlexibilitySettlementObject**]	**Required** It describes the flexibility settlement for each pod

FlexibilitySettlementObject

Table 25: Flexibility Settlement Object Model

Field	Type	Description
pod	String	Id of the PoD
power	[**MeasuredPowerObject**]	Describes the measured power

MeasuredPowerObject

Table 26: Measured Power Object Model

Field	Type	Description
Index	String	Index of the interval in the time frame (e.g. from 0 to 95 in day-ahead market)
requestedP	Number	Active Power requested (in kW)
measuredP	Number	Active Power measured (in kW)
paidP	Number	Active power paid (in €)
penaltyP	Number	Active Power Penalty to be paid (in €)
requestedQ	Number	Reactive Power requested (in kVar)
measuredQ	Number	Reactive Power measured (in kVar)
paidQ	Number	Reactive power paid (in €)
penalty	Number	Reactive Power Penalty to be paid (in €)
requestPlayerId	String	Id of the DSO or TSO that requests for flexibility
offerPlayerId	String	Id of the Aggregator

3 Interfaces and Communication Mechanisms

3.1 API Interfaces

All the REST APIs exposed by the Platone Market Platform implement an authentication mechanism based on Oauth2.0 [6] over HTTPS connection.

Below a table with the list of APIs exposed. The Open API standard documentation is available in the GIT repository [7].

Table 27: Market Platform REST APIs

Name	Url	Method	Parameters	Responses
Retrieve Smart Contracts List	/smartContracts	GET	In request: aggregatorId : String	Success (200) smartContracts: Array[**Smart Contract**] Error (500) Error Message - *String*
Update PoDs Registry	/podRegistry	POST	In body: Array[**PoD**]	Success (200) Success Message: *String* Error (500) Error Message: *String*
Create Flexibility Service	/flexibilityService	POST	In body: FlexibilityService	Success (200) flexibilityServiceId: *String* Error (500) Error Message - *String*
Send Technical Outcomes	/technicalOutcome	POST	In body: **TechnicalOutcome**	Success (200)

				Success Message: *String* Error (500) Error Message: *String*
Retrieve Settlement Outcomes	/settlements	GET	In request: aggregatorId : String	Success (200) settlements: Array[Settlement] Error (500) Error Message - *String*

3.2 Message Broker

The Platone Market Platform offers in its communication layer a Message Broker based on Apache Kafka. All the connections to the Message Broker are secured through a mutual authentication based on TLS [8].

Validated Outcome

The Market Platform publishes the Validated Outcome in the Message Broker. The Validated Outcomes are filtered for each Market Participant (DSO, TSO and Aggregator(s)) and published in different Kafka Topics. Each consumer is authorized to read in a specific topic.

Table 28: Message Broker Topics

Topic	Publisher	Subscriber	Message
DSOOutcome	Market Platform	DSO Technical Platform	Validated Outcome filter by DSO
TSOOutcome	Market Platform	TSO Simulator	Validated Outcome filter by TSO
AggOutcome(Id)	Market Platform	Aggregator(id)	Validated Outcome filter by Aggregator (id)

3.3 UI Interfaces

Login

The Login section allows access to the Market Platform web dashboard for both the administrator user and the market participant users (DSO, TSO, and Aggregator). The system transmits the credentials to the server, which according to the type of the user gives access to the related section.

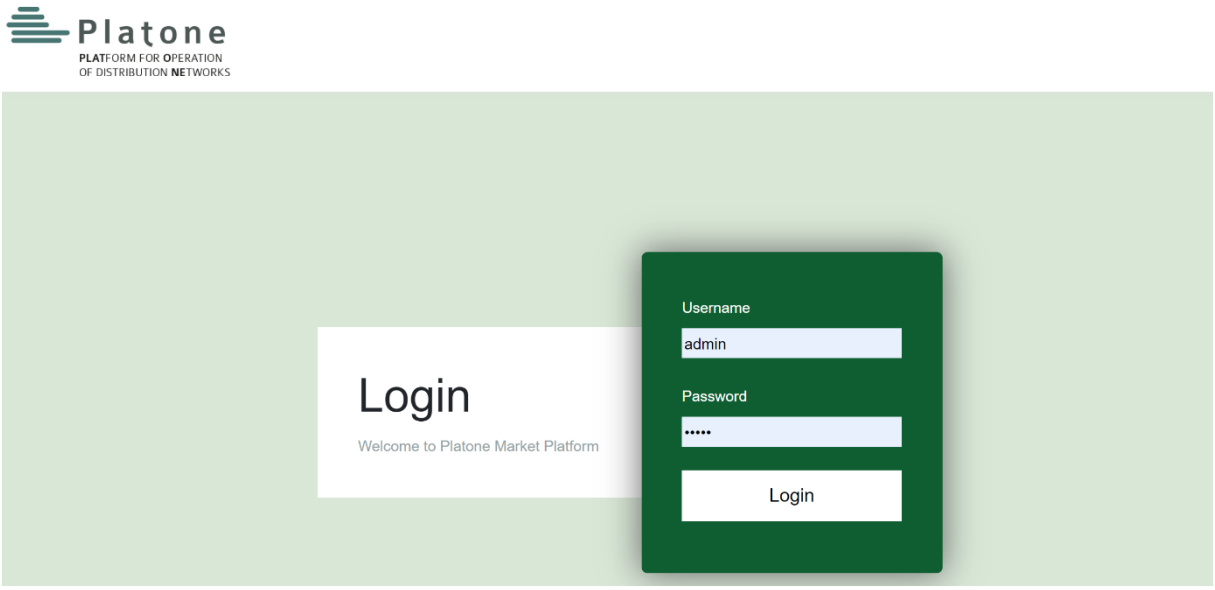

Figure 2: Login Section

Dashboard

The Market Platform web dashboard is the Home Page of the Market Platform UI. It allows to the administrator to visualise all the Market Platform activities (sessions, PoDs registry, flexibility services, market results) and to a Market Participant to visualize all the historical market data related to him: flexibility services (requests or offers) created, validated market outcomes and settlement.

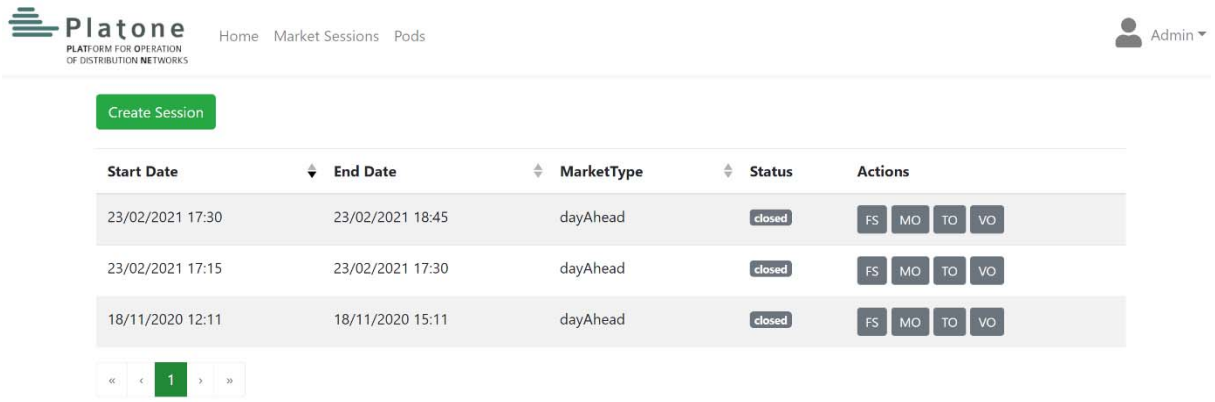

Figure 3: Market Dashboard - Market Sessions

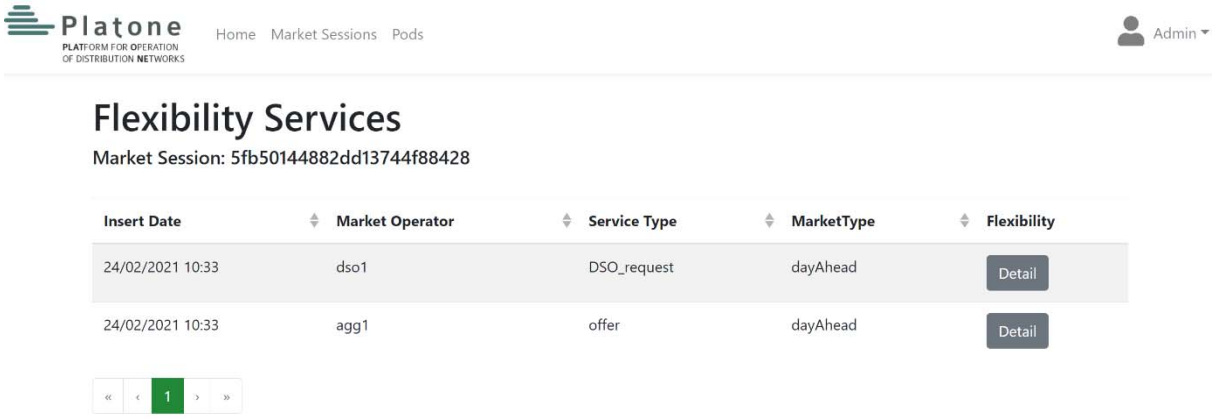

Figure 4: Market Dashboard - Flexibility Services

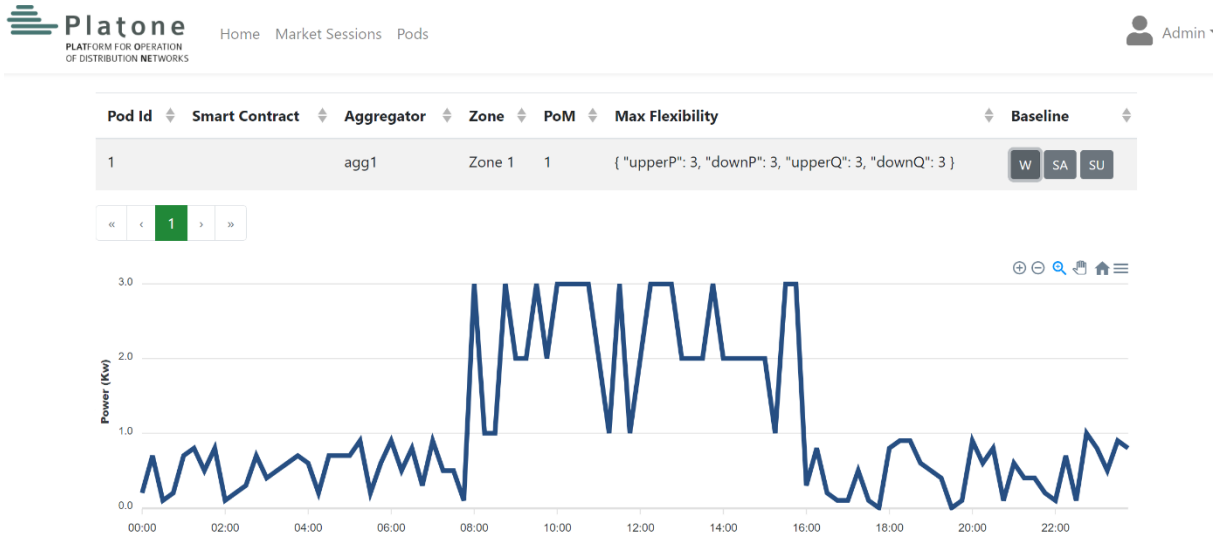

Figure 5: Market Dashboard - PoDs and Baseline

4 Languages, Technologies and External Tools

Table 29: Languages, Technologies and External Tools

Layer/Component	Languages	Technologies/Framework	External Tools
UI Layer	Javascript HTML5 CSS/SCSS	Docker Vue.js	Nginx
Service Layer	Javascript	Docker NodeJs ExpressJs	
Data Layer	Javascript	Docker NodeJs	MongoDB
Communication Layer	Javascript	Docker REST APIs NodeJs	Apache Kafka Express Gateway
Blockchain Service Layer	Solidity	Docker Truffle	Ethereum Blockchain Nodes

5 Deployment and availability

5.1 Deployment

The deployment process foresees using Docker containers. The use of Docker ensures not only an easy deployment process and total portability of the solution, but also a high level of scalability of the released applications.

Hardware

SO: Linux Host

Ram: > 4GB

Disco: > 100GB

Software

Docker > 18.06.1-ce

DB Container

```
$ docker run -d --name db <your-volume-path>:/data/db -p 27017:27017 mongo:latest
```

Kafka Container

```
$ cd app/kafka #location of docker-compose.yml
$ export HOST_IP=<your-ip-address> #ip address for kafka listening
$ docker-compose up -d
```

BackEnd Container

```
$ cd app #location of DockerFile
$ docker build -t platone-market-platform:1.0
$ docker run -p 8081:8081 -e DATABASE_URL=<your-db-url> -d platone-market-
platform:1.0
```

Web App Container

```
$ cd client #location of DockerFile
$ export API_URL=<your-api-url> #URL of BackEnd APP
$ docker build -t platone-market-platform-ui:1.0
$ docker run -p 80:80 -p 443:443 -d platone-market-platform-ui:1.0
```

5.2 Availability

The source code and the DockerFiles necessary for the deployment are available in the RWTH GIT repository [7]. ENG also provides a demo version, hosted in its cloud environment located at Pont-Saint-Martin (Italy).

Software REPO

Github -> https://git.rwth-aachen.de/acs/public/deliverables/platone

Demo Version

Web Dashboard -> https:// platone.eng.it

REST API -> http:// platone.eng.it:8080/api

Blockchain Interface -> http://platone.eng.it:8081/api/contracts

Message Broker -> http:/platone.eng.it:9042

6 Conclusion

The work done at this stage provided the first prototype of the Platone Market Platform that enables the creation of a day-ahead flexibility market, the validation of the flexibility services and the provisioning of the settlement outcomes. The blockchain technology ensures that Market Participants can participate to the market in a secure and transparent way, and a token-based incentivisation for customer engagement is enabled.

The first release of the Platone Market Platform implements a set of functionalities, as described in Ch.2.2, for integrating the Platone Market Platform with other platforms of the Platone Open Framework and for the integration within the Italian demo site.

In particular, the integration and validation in the Italian Demo site, will be a very important steps for testing the first prototype of the platform in a real environment that exploit all the functionalities provided.

A detailed description for installation and configuration of platform components is provided to ensure the usability. In addition, a demonstrative version of the entire Platone Market Platform architecture is available within ENG cloud infrastructure.

7 List of Tables

8 List of Figures

9 List of References

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

[2] Grant Agreement No. 864300 – Platone.

[3] Platone_D2.1_PlatOne Platform requirements and reference architecture (v1).

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

[5] “Open API 3.0 - Specifications,” [Online]. Available: https://swagger.io/specification/.

[6] “OAuth 2.0,” [Online]. Available: https://oauth.net/2/.

[7] “Platone GIT Repository,” [Online]. Available: https://git.rwth-aachen.de/acs/public/deliverables/platone.

[8] B. Pournader, “What is two-way TLS?,” 23 May 2018. [Online]. Available: https://benpournader.medium.com/what-is-two-way-tls-d90600e2fc06.

10 List of Abbreviations

Abbreviation	Term
API	Application Programming Interface
DA	Day Ahead
DB	Database
DSO	Distribution System Operator
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer
NSGA	Non-dominated Sorting Genetic Algorithm
PoD	Point of Delivery
PoM	Point of Measurement
REST	REpresentational State Transfer
RT	Real Time
TLS	Transport Layer Security
TSO	Transmission System Operator
UI	User Interface