Title:	Document Version:
D6.2 Storage as a service and innovative optimized solutions	3.0

Project Number:	Project Acronym:	Project Title:
H2020-731205	WiseGRID	Wide scale demonstration of Integrated Solutions for European SmartGrid

Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type*-Security*:
M18	M18 (April 2018)	R-PU

*Type: P: Prototype; R: Report; D: Demonstrator; O: Other.

**Security Class: PU: Public; PP: Restricted to other programme participants (including the Commission); RE: Restricted to a group defined by the consortium (including the Commission); CO: Confidential, only for members of the consortium (including the Commission).

Responsible:	Organisation:	Contributing WP:
Xavier Benavides	AMP	WP6

Authors (organisation):

Xavier Benavides (AMP), Ignacio Benítez (AMP), Jorge Sanjuán (AMP), Félix Ruiz (AMP), Benjamin Kraft (VS), Stefan Meir (VS), Alexandros Nikolakakis (HYP), Alexandre Lapedra (BYES), Giuseppa Caruso (ENG), Leandro Lombardo (ENG), Alberto Zambrano (ETRA), Amparo Mocholí (ITE), Javier Monreal (ITE)

Abstract:

Description of the WiseGRID Storage as a service, which will promote the distributed renewable energy together with the different possible uses of electrical energy storage, in order to increase the revenues and benefits of installing this kind of systems. Competing or even substituting the need for conventional power system due to the scalable investments and the high costs associated with erecting new power plants and distribution of energy.

The report includes the definition of the Innovative and advanced optimized storage solutions that will allow the integration of storage in the Smart Grid considering aspects such as the adaptability, use and flexibility. The related tasks are T6.2 and T6.3.

Keywords:

StaaS (Storage as a Service), VPP (Virtual Power Plant), Renewable energy, Energy Storage, Optimization, Forecast

Revision History

Revision	Date	Description	Author (Organisation)
V0.0	01.03.2018	ToC proposal	Ignacio Benítez (AMP)
V0.1	02.03.2018	ToC review	Benjamin Kraft (VS)
V0.2	05.03.2018	ToC review with all the partners	Ignacio Benítez (AMP)
V1.3	21.03.2018	First version	Ignacio Benítez (AMP)
V2.0	10.04.2018	Second version	Ignacio Benítez (AMP)
PR_v1.0	16.04.2018	First Version for Peer Review	Ignacio Benítez (AMP)
V3.0	25.04.2018	Final Version	Ignacio Benitez (AMP)

INDEX

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

The present Deliverable is a description of the WG StaaS/VPP Component. The main component objective and challenge is to facilitate transactions of the unused storage capacity of local energy storage systems in prosumers' premises, usually as a support of Renewable Energy Resources, mostly solar PV Panels, in energy markets, either wholesale (i.e. buying and selling energy daily in an auction bid in order to meet the daily energy needs of the system) or ancillary service (also called Flexibility, following the USEF – Universal Smart Energy Framework [1]).

Due to the prosumer's low amount of capacity, and the requirements of the different markets to participate, each user's capacity is aggregated from several installations and prosumers and offered as a whole by an intermediate agent, called the Aggregator. The WG StaaS/VPP offers services, therefore, to the following agents and with the following benefits, as described in Section 2.1:

- Services to prosumers:
    - Benefit from participation in ancillary services markets
    - Avoid curtailment of RES (if storage is in place)
- Services to Balance Responsible Party (BRP):
    - Day-ahead/Intraday portfolio optimization
    - Self-balancing
    - Generation optimization
- Services to DSOs:
    - Load frequency control
    - Grid capacity management
        - Deliver peak load electricity
        - Load-following power generation at short notice (DRES + batteries combined)
    - Voltage support
    - Power quality support

In order to accomplish this objective, the WG StaaS/VPP has to meet a number of requirements that were defined in a previous Work Package, in form of Primary and Secondary Use Cases [2]. Once these requirements are defined, there are also some technical barriers or difficulties to overcome, such as how to estimate individual storage capacity or flexibility, how to aggregate them all and offer them in the different markets, and how to disaggregate and send back the orders of flexibility management when an order is accepted. Another technical challenge is to deploy a system that must be able to communicate with any Storage System that may be able to participate and, in line with this specification, how to deploy such a system that may be able to scale up to hundreds of systems communicating to a central manager in real time.

To meet all these requirements, the WG StaaS/VPP has been divided in different interconnected modules. Each one of these modules serves different purposes, allowing the Component to provide all the functionalities required.

First, in order to allow all the ESS (Energy Storage Systems), and specifically BESS (Building Energy Storage Systems), to participate in the system using a common language, this language has been defined, in form of a common data model that has been implemented in the WG StaaS/VPP Component, and is described in

detail in Section 3.1. This data model describes the configuration parameters for new systems being connected; configuration parameters for working modes, addressed to frequency and voltage regulation, and operation commands and variables used to communicate with the systems and vary their behaviour from a central manager.

Having defined the common data model to be used, the following modules are implemented in the WG StaaS/VPP Component:

- Flexibility Estimation Module: this module has the objective to provide a forecast of the unused storage capacity that would be available in the following hours (given the forecasted behaviour is met, and no flexibility requests are received). This module uses as inputs two forecasts, provided by other modules: production forecast and demand forecast. The output of this module mainly consists in a bandwidth of flexibility (charge and discharge) for each time step, either in power, or in power maintained for a given time window (i.e., energy). This provision can be produced again any time that is needed, and also takes into account as an input a flexibility request received by the energy market, in this case through a module called the sales agent.
- Scheduler Module: this module is in charge of receiving the accepted offers from the sales agent and distributing them among the available systems in a proper way. To do so, it implements two control loops: a slow control loop, which is a provision of setpoints for all the storage systems, and a second, fastest loop, which evaluates the response of the storage systems, and promotes them to follow the proposed signals in order to accomplish with the demand request.
- Sales Agent Module: this module interacts with the energy markets, aggregating the flexibility signals in offers with the formats and characteristics required. In the scope of the WG StaaS/VPP Component, three different energy markets have been described: one is the wholesale market, where the WG StaaS/VPP participates offering energy to be purchased by other participants; the other one is the ancillary service for balancing purposes, where a BRP manages the offers for frequency and voltage services in the primary and secondary regulation levels mainly, and a third one for Distribution grid congestion alleviation, where the WG StaaS/VPP offers the extra capacity for this purpose, in a market managed by the DSO.
- Billing Management Module: this module provides the economic information with the results of the transactions and the service to the final users or prosumers.
- MQTT Broker for communication services. The use of the MQTT protocol and the Subscription paradigm, instead of a Peer-to-peer communication with all the energy storage systems, alleviates communication congestions and boosts up the efficiency of the Component's communication channel. The WiseGRID Interoperable Platform (WG IOP [3]) has been developed, implementing the open source multiprotocol message broker Rabbit MQ, to serve as a common interface for any storage system willing to be connected to the WG StaaS/VPP Component.
- MQTT Interfaces for the three different energy storage systems are described. These interfaces have been developed in the scope of the WG StaaS/VPP works, in order to allow communication between the storage systems and the common MQTT Broker. The development of specific HW/SW interfaces has also been described in this Deliverable (Sections 3.9, 3.10 and 3.11), for validation purposes.
- Real Time Monitor Module: this internal module gathers all the data interaction among the modules and the databases implemented in the WG StaaS/VPP Component.
- RES Monitoring Module: this module measures RES production and feeds the data into the WG StaaS/VPP, to be used for the different modules, mainly by the Production Forecast Module.
- Weather Forecast Module: this is an external module, which is described in the document for clarification purposes. It uses an external source to provide a forecast of weather conditions, such as irradiance, to be used by the Production Forecast Module.

- Production and Demand Forecast Services: These services are needed by the Flexibility estimation to produce the flexibility forecasts. Based on historical records, and on external inputs, such as weather forecast, these services provide an estimation of energy demand and production from RES for the following hours, belonging thus, to the family of Short-Term Load Forecast (STLF) tools.

All these modules and the databases needs (two have been defined: long-term, with historical records, and short-term, with operational data) have been implemented as a web application, with Javascript programming language using Node JS JavaScript runtime and Angular JS framework. This way, the solution is scalable and universal, i.e. accessible virtually from anywhere an internet connection is available. A special care has been placed, finally, on the design and the look-and-feel of the human interfaces, since they are the ones that will really demonstrate the value of the complete solution developed.

1 INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

The main document's objective is to describe the works performed in the definition, design and development of the WiseGRID StaaS/VPP (Storage as a Service/Virtual Power Plant) Component, including the necessary interactions among this component and others developed in the scope of the WiseGRID Project, which will be mainly done through the WiseGRID Interoperable Platform (WG IOP) (described in detail in Deliverable D4.2 [3]).

The present Deliverable has been conceived to be a self-container of the WG StaaS/VPP component's objective and functionalities. To this end, the document describes the whole component, starting from the data model defined and the inner logic that provides the functionalities sought, to the graphical user interface and how these functionalities are provided to the StaaS/VPP user agents, such as the Manager or Operator. These functionalities are designed to give a response to the system requirements that were defined in the WG StaaS/VPP Component Primary and Secondary Use Cases. These Use Cases can be consulted in Deliverable D2.1 [2].

1.2 SCOPE OF THE DOCUMENT

The present deliverable summarizes the main objectives and functionalities of the WiseGRID StaaS/VPP Component, and fully details the data model of the component, the different logical modules or controllers that provide the functionalities needed for its operation, and its different views. This work corresponds to the objectives of the WiseGRID Project Work Package 6 ("WP6: WiseGRID Storage-as-a-Service/VPP and optimized storage solutions") and, more specifically, to the tasks T6.2 and T6.3, with the following description, extracted from the Project's Grant Agreement:

- "**T6.2 Storage as a service (STaaS)/VPP**. This task will be based on the design and provision of the WiseGRID Storage as a service. Proper models will be identified to optimize storage services to enable different scenarios, according to the flexibility capabilities. The WiseGRID Storage as a service […] will offer services either to final consumers (households, business or industries), intermediates (ESCOs, aggregators or energy cooperatives) or to the grid operator (DSO). WiseGRID Storage as a service will promote the distributed renewable energy and will then compete or substitute for conventional power also to enhance the economic potential because of the high costs associated with erecting new power plants."
- "**T6.3 Innovative and advanced optimized storage solutions.** This task focuses on the Innovative and advanced optimized storage solutions that will allow the integration of storage in the Smart Grid considering aspects such as the adaptability (different type of technologies), expendability (large number of storage) and flexibility. […] Innovative algorithms will be modelled to enable optimal utilization of different storage systems at different levels of the distribution grid (district-level and substation level) under different constraints (location, time, response, capacity, duration, etc.). The module will perform an internal and iterative optimization of the utilization of aggregation of storage systems on the basis of their indigenous characteristics and capabilities to provide a wide range of services (such as Storage as a service defined previously) to the distribution grid (balancing, ancillary services) following the high level optimization and flexibility requirements […]."

The present document is divided in three main sections: first, there is a description of the internal architecture and the different modules that have been designed and developed in order to meet all the requirements for this system.

Then, this architecture is explained in detailed, starting from the data model that has been defined to communicate with Energy Storage Systems (ESS), and mainly with electrochemical batteries, i.e. Battery Energy

Storage Systems (BESS). These can be typically installed in the prosumers' premises along with a Renewable Energy Source (RES), providing charge and discharge capabilities through an inverter in order to optimize the electricity bill of the prosumer, in a first choice. But their capacity and production can also be offered to an aggregation agent (the Aggregator) that can sell it as an aggregated service to the electricity markets, such as the wholesale market; or an ancillary service (also called Flexibility) market, where the energy is offered to balance the equilibrium that must be constantly maintained between generation and demand, in different forms, which are gathered in the following types: energy/demand management, voltage regulation, or frequency regulation.

This is the scope of the present development and is where the experience from all the participating partners have to be put in practice, as a R&D innovation tool or component, that can deal with the mentioned objectives. There have been, mainly, two difficulties for this objective:

1. How to forecast the capacity (i.e. flexibility) that can be offered from each RES and BESS (or ESS) system.
2. How to combine the different flexibilities from the different systems in an optimal way, offering it to different electricity markets, and properly regulate the orders whenever an offer is accepted (and a response is expected from the system, that must react as a unique entity).
3. How to schedule the aggregated assets so that the service requirements are fulfilled at any time.

The present deliverable describes the solution that has been developed to overcome these specifications. Finally, an effort has been placed in how to display the functionalities developed to the user. Besides the interaction with the different electricity markets, the user interface displays also the status of the units connected to the system, and a global view of how these units may respond through the day.

Work Package 6 of the WiseGRID Project has dealt with all these challenges, being the WG StaaS/VPP Component the result, which is explained in the following pages.

1.3 STRUCTURE OF THE DOCUMENT

The Deliverable is structured as follows:

- Chapter 2 describes the functionalities of the component, and how these objectives have been translated into an architecture of different modules interconnected, each of them with its own purpose and logic, all of them serving to accomplish the global objectives of the WG StaaS/VPP Component.
- Chapter 3 includes a first section where the defined data model is described, and then sections where each of the modules developed are described, considering a brief description of the functional objectives that motivated their design, the logic and algorithms that are necessary for their operation, and the interfaces to communicate with other modules and agents in the component architecture.
- Chapter 4 describes the programming tools that have been used to implement all the modules and the internal architecture.
- Chapter 5 describes the different views and graphical user interfaces that have been developed to display the different functionalities of the WG StaaS/VPP Component.
- Chapter 6 presents some conclusions.

The document also includes references, a list of acronyms and an Annex.

2 THE WISEGRID STORAGE AS A SERVICE SOLUTION

2.1 OVERVIEW AND OBJECTIVES

WG StaaS/VPP is the WiseGRID technological solution targeting aggregators with a respective portfolio based on distributed generation and storage assets. The main goal of this solution is to help consumers and prosumers (be them households or corporate) to be aggregated and offer to the market their unused storage capacity as well as spare generation in the form of a VPP. The objective of the tool is the provision of services to different actors:

- Services to prosumers
  - Benefit from participation in energy markets
  - Avoid curtailment of RES (if storage is in place)
- Services to Balance Responsible Party
  - Day-ahead/Intraday portfolio optimization
  - Self-balancing
  - Generation optimization
- Services to DSOs
  - Load frequency control
  - Grid capacity management
    - Deliver peak load electricity
    - Load-following power generation at short notice (DRES + batteries combined)
  - Voltage support
  - Power quality support

The following Use Cases related to the WG StaaS/VPP were identified in WP2:

Table 1 – Use Cases for WG StaaS/VPP

HL-UC 4_PUC_2_Batteries management at aggregator level (grid support)
HL-UC 4_SUC_2.1_Batteries dispatch management
HL-UC 4_SUC_2.2_Black start capabilities
HL-UC 4_SUC_2.3_Power management for peak-shaving and load harmonization
HL-UC 4_SUC_2.4_Backup power for residential area
HL-UC 4_PUC_3_Ancillary services
HL-UC 4_SUC_3.1_Market scheduling
HL-UC 4_SUC_3.2_Combination of applications/services in the same storage system
HL-UC 4_SUC_3.3_Batteries automatic dispatch
HL-UC 4_PUC_4_Combination of battery storage systems
HL-UC 4_SUC_4.1_Parameter configuration of storage systems
HL-UC 4_SUC_4.2_Priority list of units running

HL-UC 4_PUC_2_Batteries management at aggregator level (grid support)
HL-UC 6_PUC_1_VPP monitoring and management
HL-UC 6_SUC_1.1_Resource metering
HL-UC 6_SUC_1.2_VPP RES forecast
HL-UC 6_SUC_1.3_VPP flexibility forecast
HL-UC 6_SUC_1.4_Strategies definition
HL-UC 6_PUC_2_VPP market participation
HL-UC 6_SUC_2.1_VPP market participation and bid calculation
HL-UC 6_SUC_2.2_VPP ancillary market participation and bid calculation
HL-UC 6_SUC_2.3_VPP unit scheduling
HL-UC 6_PUC_3_VPP Real time control
HL-UC 6_SUC_3.1_Real time flexibility calculation
HL-UC 6_SUC_3.2_VPP implementation of ancillary services
HL-UC 6_SUC_3.3_Real time decision making
HL-UC 6_PUC_4_VPP users relationship management
HL-UC 6_SUC_4.1_Manage contractual issues
HL-UC 6_SUC_4.2_Define and manage member compensation
HL-UC 6_SUC_4.3_DSM and DR mechanisms

The different Use Cases are described in deliverable D2.1 (“WiseGRID requirements, Use Cases and pilot sites analysis”) and won’t be explained in detail here. As a summary the following functionalities are attributed to the StaaS/VPP tool:

- Monitoring of assets
- Forecasting and Flexibility Estimation
- Market participation/DR campaigns
- Scheduling and dispatching
- Billing
- Contract Management

2.2 ARCHITECTURE

Based on the defined functionalities of the tool the architecture illustrated in Figure 1 was defined.

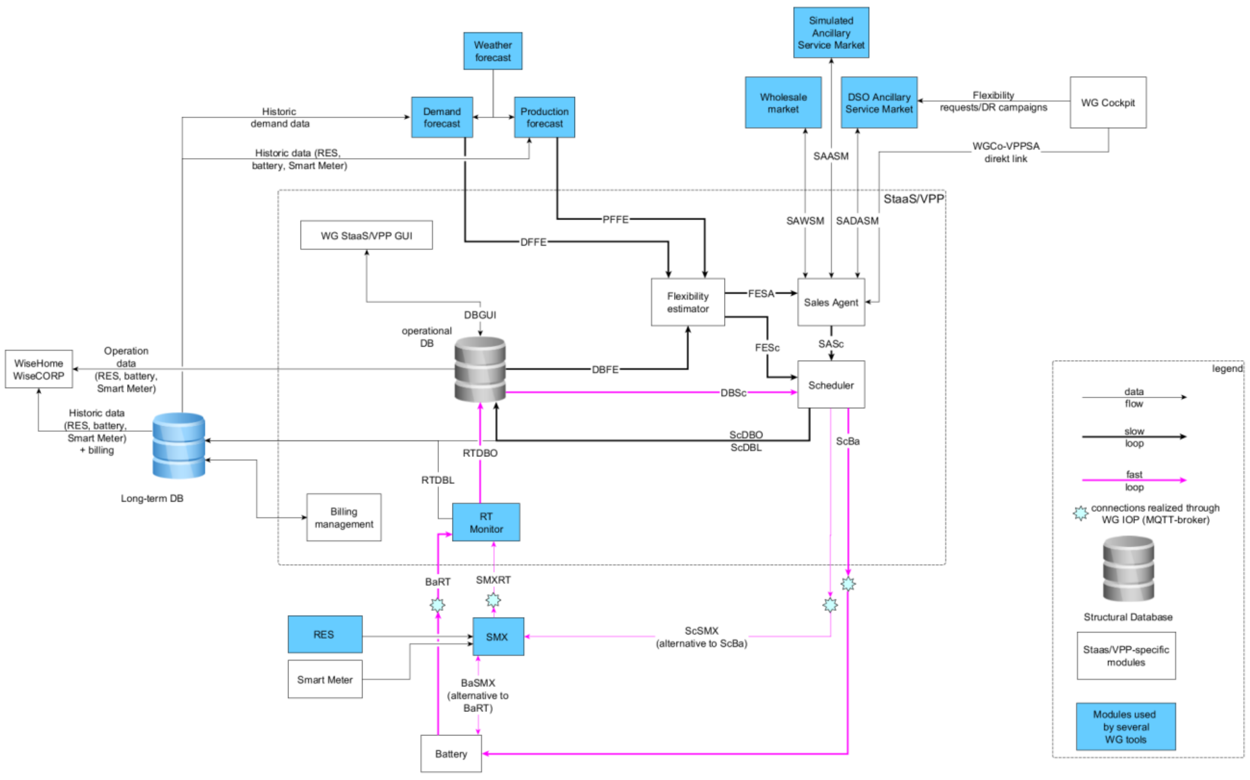


Figure 1 – STaaS/VPP dataflow overview[1]

The WG StaaS/VPP tool consists of 6 core modules with different functionalities:

- RT Monitor: In order to monitor the assets a Real-Time monitor is used. It is able to subscribe to a MQTT topic and keep the current status of each device in a database. Furthermore, the module is able to push a history of the changes into a different database (each change hold by a document). The RT Monitor receives MQTT messages directly from the batteries or from the SMX which is the gateway for RES and the Smart meters. As a second option, batteries can transmit data via the SMX as well.
- Flexibility Estimator: Based on forecast data the flexibility estimator calculates available future capacities that can be offered to energy market players and grid operators.
- Sales Agent: The Sales Agent is selling available capacities to or buying energy from the different markets, e.g. Wholesale and Ancillary service markets. Moreover, it processes the requests from the DSO tool WG Cockpit either directly or via a newly defined DSO Ancillary Service Market
- Scheduler: The awards received from the energy markets result in an overall schedule for the VPP. The scheduler is distributing the total power to the different assets considering the characteristics and the current status of each asset. For controlling purposes, the assets receive setpoint data from the scheduler.
- Billing management: By providing capacities to the energy markets the VPP and its assets generate profit. The billing module manages the profit calculation considering the received awards and assigns the payments to the different assets according to the provided capacities.

[1] Channels of data flow are labelled using the abbreviations of the tools involved, e.g., "FESc" for the data channel from Flexibility Estimator to Scheduler. Data flow through the WG IOP is depicted with light blue stars

- Graphical User Interface (GUI): Via the GUI the aggregator can monitor the overall VPP and its assets. Moreover, configuration, contracting and partly controlling is handled via the GUI.

For storing data an operational as well as long-term database will be established.

Beside the core modules of the tool, supporting modules that represent the production and demand forecast as well as the energy markets are of importance. Whereas the forecast modules provide production and demand profiles to the flexibility estimator the market modules provide market information (e.g. prices) to the Sales Agent. The Sales Agent now provides bids to the different markets. Like the RT monitor those modules can be partly used by various WiseGRID tools and thus are marked in blue.

In addition, several direct links to different WiseGRID tools are considered in the architecture. Whereas the interface to the WG Cockpit considers the deployment of the VPP assets in order to support the grid the link to WiseHome and WiseCORP serves as end user interface. In all cases the message transfer is coordinated via the IOP using MQTT.

3 DESCRIPTION OF DATA MODEL AND MODULES

The data model that has been defined for the WG StaaS/VPP Component is described first. The remaining subsections include the detailed description of all the modules that have been defined for the WG StaaS/VPP Component to operate properly, accomplishing the objectives described and ready for the Use Cases that were defined in a previous design stage (these Use Cases have been described in Deliverable D2.1 "WiseGRID requirements, Use Cases and pilot sites analysis" of the WiseGRID Project [2]).

3.1 DATA MODEL

3.1.1 Introduction

The data model presented below describes the structure and format of the communications between the ESSs and any controller interacting with them.

The variables that characterize the operation of the system are published on an MQTT Topic periodically. Therefore, any external application can have access to them in order to monitor and control the system behaviour. Two different MQTT Topics allow the external controller to configure the operation of the system: control modes and parameters.

3.1.2 Variables

Table 2 shows the System Data, i.e. the information about the ESS that will be published upon connection to the MQTT server.

Table 2 – System Data

System Data		
on MQTT-connect: publish MQTT/battery/<batteryUID>/system		
Type	**Variable**	**Units**
String	SN	String
String	FW_Version	String
Double	Rated_Ch_Power	W

Double	Rated_Dsch_Power	W
Double	Usable Capacity	Wh
Double	Rated PV Power	W

The information on Table 2 should be enough to identify each system and to know its basic information and characteristics. All the other modules and algorithms will use and combine this information in order to obtain the necessary data and parameters of the whole VPP.

On the other hand, during the normal operation of the system, the main internal variables will be published following Table 3.

Table 3 – Variables

Variables				
on MQTT-connect: battery sets LWT-message with status="disconnected" to be send in case of ungrace-ful disconnect periodically: publish MQTT/battery/<batteryUID>/status on graceful disconnect: publish MQTT/battery/<batteryUID>/status with status="disconnected" on ungraceful disconnect: broker sends LWT-message to subscribers				
Type	**Variable**	**Explanation**	**Units**	**Range**
Double	Meter_Active_Power	at the grid connection point of the household as measured by the battery controller	W	[-Pmax, Pmax]
Double	Meter_Reactive_Power	at the grid connection point of the household as measured by the battery controller	VAr	[-Qmax, Qmax]
Double	Inverter_Active_Power	(AC) active power of the battery inverter	W	[-Pmax, Pmax]
Double	Inverter_Reactive_Power	of the battery inverter	VAr	[-Qmax, Qmax]
Double	Inverter_PV_Power	of the PV inverter included in a battery system, if it is such a “hybrid” system	W	[-Pmax, Pmax]
Double	External_PV	of a stand-alone PV inverter, which is somehow measured by or in communication with the bat-tery controller	W	[-Pmax, Pmax]
Double	Inverter_Battery_Power	DC power of the battery inverter	W	[-Pmax, Pmax]
Double	Battery_SOC	usable energy presently in the battery divided by actually usable energy capacity	%	[0, 100]
Double	Battery_SOH	actually usable energy capacity divided by rated capacity	%	[0, 100]
Double	Battery_Voltage		V	[0, 500]
Double	Meter_Grid_Voltage	at the grid connection point of the household as measured by the battery controller	V	[0, 500]
Double	Meter_Grid_Frequency	at the grid connection point of the household as measured by the battery controller	Hz	[0, 500]
Array	Temperatures:			
	Batt_Cell_Max_T	maximum temperature of the battery cells	ºC	[-500, 500]
	Batt_Cell_Min_T	minimum temperature of the battery cells	ºC	[-500, 500]
	Inverter_T	inverter IC temperature	ºC	[-500, 500]

Double	Charge_Available	actual maximum available active charge power (AC) of the battery	W	[0, Pmax]
Double	Discharge_Available	actual maximum available active discharge power (AC) of the battery	W	[0, Pmax]
Int	Status	0: disconnected 1: connected 2: charge 3: discharge 4: standby 5: error 6: busy 7: islanding	---	---
Array	Alarms	Error numbers	---	---
Double	Inverter_PV_Voltage		V	[0, 1000]
Int	Working mode	as defined in sheet 'Operation'	---	---
Double	Demand	power consumed at the house	W	[-Pmax, Pmax]

3.1.3 Operation

The Operation of the system is configured by commands published at the corresponding MQTT Topic. Table 4 shows the available commands. Some of these commands require a set of parameters in order to configure the behaviour of the system.

Table 4 – Commands

Commands		
on MQTT-connect: subscribe to MQTT/battery/<batteryUID>/command **on command received: publish MQTT/battery/<batteryUID>/command/ack with ID of received command**		
Command	Name	Parameters
0	Set Mode	Working Mode
1	Add Sch	[Time (s), [Control Mode, Active, Control Parameters] (x n)] (xN)
2	Clear Sch	
3	Set Manual	[Control Mode, Active, Control Parameters] (x n)

Command 0, “Set Mode”, establishes the Working Mode, as defined in Table 5. Working mode 0 corresponds to the standard system operation as it would work in an independent installation. On the other hand, working modes 1 and 5 can be used in order to externally control the system when it is integrated with other systems in a VPP or similar. Finally, working modes 2, 3 and 4 are internally managed and should not be used by an external controller.

Table 5 – Working Modes

Working Modes		
Working mode	Name	Description
0	Standard	Normal system operation

1	Manual	Manual control
2	Alarm	System stops
3	Backup	Software error, backup loaded
4	Test	Configuration and test
5	Schedule	Scheduled manual control

Commands 1 and 3 of Table 4 require a set of Control Modes, introduced as an array. For each control mode, it is necessary to include its code, status (active or not) and parameters. Control modes are defined on Table 6, and the corresponding parameters are shown in Table 7. As it can be seen on the tables, control modes 2, 3 and 4 don't require any parameters to be specified.

Table 6 – Control Modes

Control Modes		
Control (n)	Name	Description
0	System	Manually set inverter target power
1	Load	Manually set target power at meter
2	PvsF	Frequency regulation
3	PvsV	Voltage regulation
4	QvsV	Voltage regulation

Table 7 – Control Parameters

Control Parameters			
Type	Variable	Units	Range
0-1: Power controls			
Double	P	W	[-Pmax, Pmax]
Double	Q	Var	[-Qmax, Qmax]
Double	MaxChPower	W	[0, Pmax]
Double	MinChPower	W	[0, Pmax]
Double	MaxDischPower	W	[0, Pmax]
Double	MinDischPower	W	[0, Pmax]
2-4: Regulation controls			

Command 1 will be used to establish a schedule of control modes that will be applied following a daily basis. Therefore, the programmed schedule will be repeated each day, if not changed. The "Time" parameter on command 1 will be a number between 0 and 86400 (24h), and the corresponding control configuration will be applied at this time. Any control mode not specified will remain unchanged. Each schedule point in command 1 will be added to the existing schedule.

3.1.4 Configuration

Control Modes involving Frequency or Voltage regulation require a specific configuration of how the ESS behaves in order to perform such regulation. The corresponding configuration parameters are shown in Table 8.

Table 8 – Configuration parameters

Config				
on MQTT-connect: subscribe to MQTT/battery/<batteryUID>/config				
Type	Variable	Explanation	Units	Range
PvsF (x2: over&under-freq)				
Bit	Mode	bit 0: Disabled/Enabled bit 1: Linear/Hysteresis	---	---
Double	Lock in	Frequency limit	ΔHz	[-8, 5]
Double	Lock in Delay	Time for triggering the algorithm	ms	[0, 32767]
Array	Lock out	Frequency limit	ΔHz	[-8, 5]
Double	Lock out Delay	Time for stopping the algorithm	ms	[0, 32767]
Array	Table	Curve values (max 4 points)	[ΔHz, %Pac]	[[-8, 5], [0, 100]]
PvsV (x2: over&under-volt)				
Bit	Mode	bit 0: Disabled/Enabled bit 1: Linear/Hysteresis	---	---
Double	Return time	Time for the power ramp	s	[1, 32767]
Array	Table	Curve values (max 3 points)	[%Vnom, %Pac]	[[0, 135], [0, 100]]
QvsV				
Bit	Mode	bit 0: Disabled/Enabled bit 1: Linear/Hysteresis	---	---
Double	Lock in	Active power limit	%Snom	[0, 100]
Double	Lock in Delay	Time for triggering the algorithm	s	[0, 32767]
Double	Lock out	Active power limit	%Snom	[0, 100]
Array	Table	Curve values (max 5 points)	[%Vnom, %Snom]	[[0, 135], [-100, 100]]

The behaviour of the system depends on the regulation control mode that is active at each time. When it is active, each control mode will cause the power of the system to vary depending on the grid voltage or frequency values. This variation is configured with the parameters of Table 8, resulting in the behaviour shown in Figure 2.

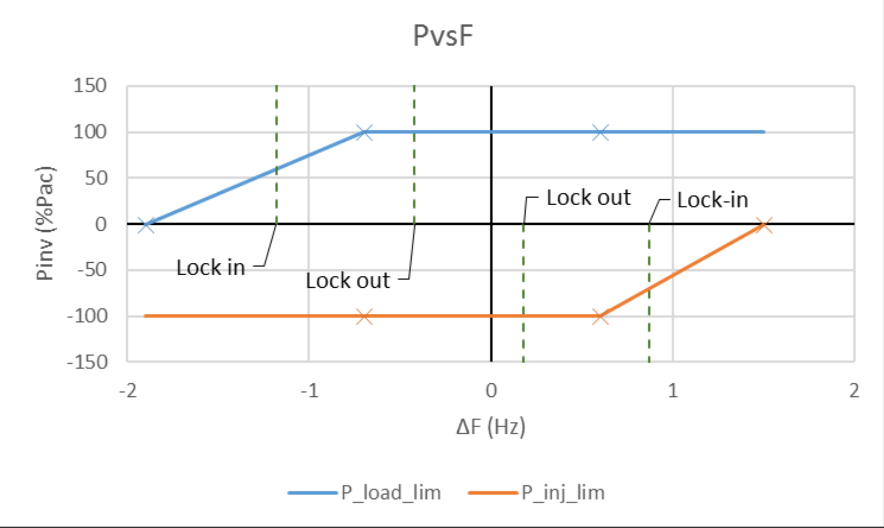
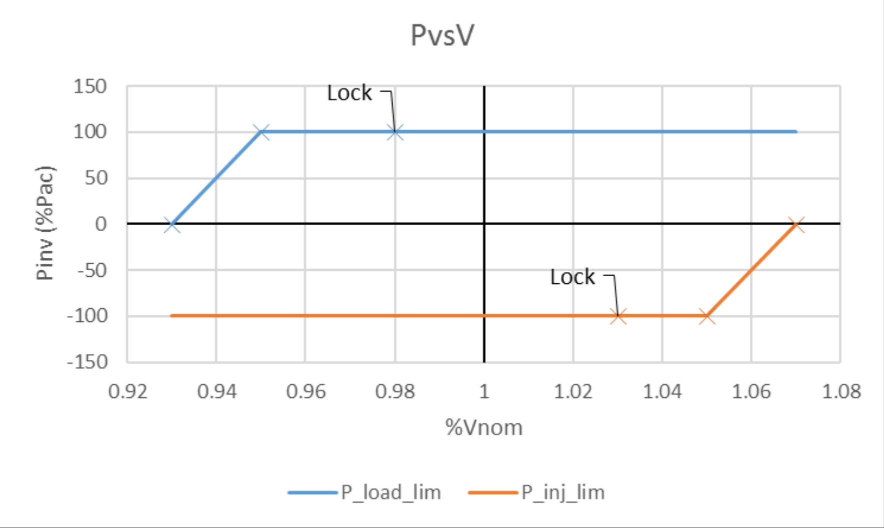
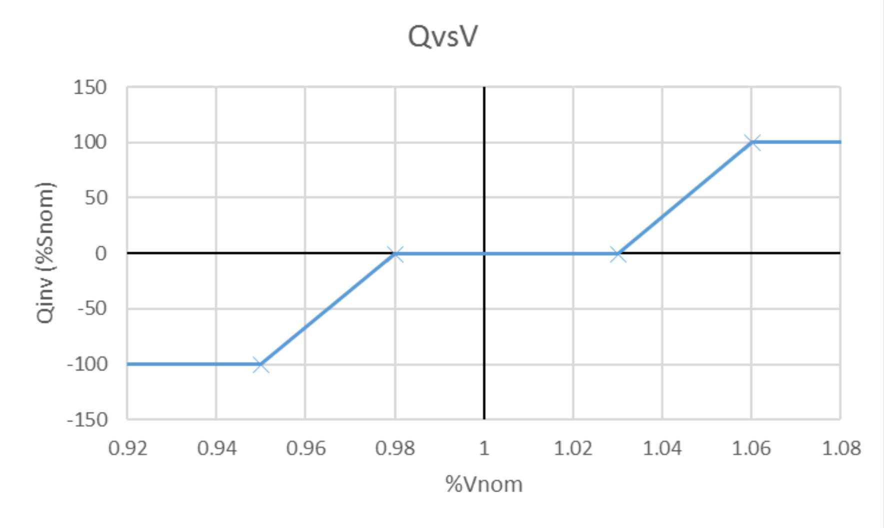
	**Pac**: inverter power at the moment the algorithm is triggered. **Lock in**: algorithm is triggered when the value stays outside this limit during more than the corresponding delay. **Lock out**: algorithm stops when the value stays inside this limit during more than the corresponding delay.
	**Pac**: inverter power at the moment the algorithm is triggered. **Lock:** (first value of the table) algorithm is triggered when the value is outside this limit and stops when the value stays inside it. **Return time:** (hysteresis mode) duration of the power ramp at the lock out.
	**Lock in Pac**: algorithm is triggered when the active power stays outside this limit during more than the corresponding delay. **Lock out Pac**: algorithm stops when the value stays inside this limit.

Figure 2 – Regulation Curves

3.2 FLEXIBILITY ESTIMATION MODULE

3.2.1 Functional objective

First, it is necessary to clarify, as accurately as possible, the concept of flexibility. This term can be defined in several ways; however, in this document, flexibility has been understood as the system's load modification capability (including charge and discharge possibilities). This definition implies that the flexibility algorithm will start with the estimation of the system behaviour in order to determine how this behaviour could be modified to fulfil any external requirements.

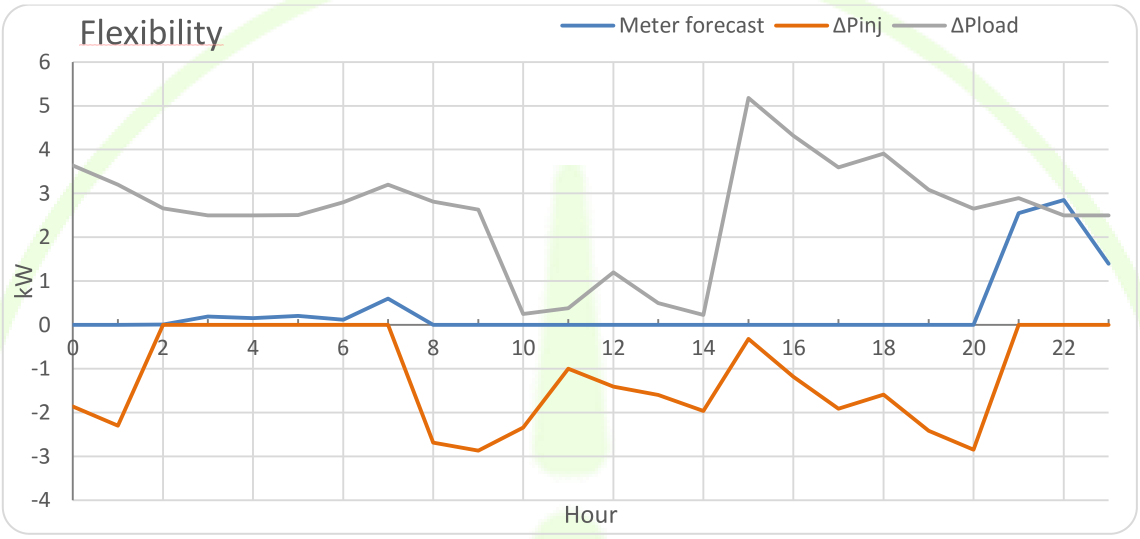


Figure 3 – Flexibility estimation graph

In a general way, the flexibility calculation is based on the inverter power flows, taking into account its limitations, and any other technical limitations. These calculations allow stablishing certain maximum variations (ΔP_{inj} and ΔP_{load} in Figure 3) reachable by either charge or discharge operations. It should be noted that these power values correspond to the possible variation applicable to the final house consumption.

With the flexibility algorithm, a first solution will be obtained focused on optimizing the self-consumption of the renewable energy source (RES) production, usually corresponding to a photovoltaic installation. Figure 4 shows an example of PV production and demand forecast for a typical house. Figure 5 presents the load forecast after taking into account the PV production and optimizing the energy consumption by means of the ESS. Considering the resulting power flows of the whole system, the flexibility variables can be obtained. With them, the Sales Agent can offer capacities and the scheduler algorithm can calculate the requirements for the system behaviour, depending on the grid and the market necessities.

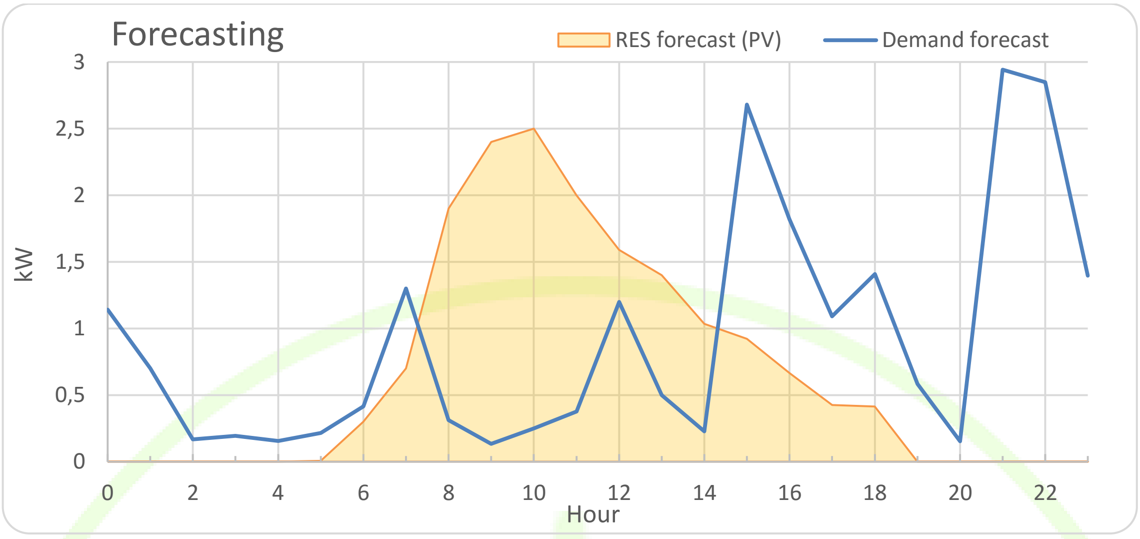


Figure 4 – Demand and generation forecast

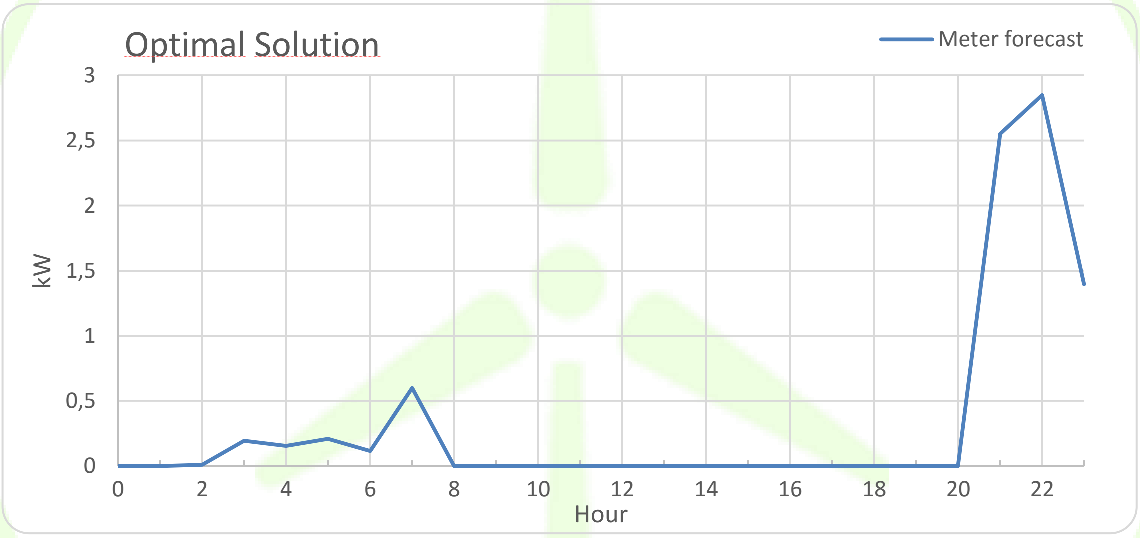


Figure 5 – First solution: Solution for increasing self-consumption

The proposed flexibility algorithm will have two main inputs: forecast and requirements. Forecast variables include the estimated demand and PV production prediction. Requirements for the system behaviour are specified by means of the power consumption objective setpoints and energy needs for fast-response peak power variations. The flexibility output of the algorithm will include the available power variation variables stated above, together with the corresponding available reactive power and the calculated evolution of the battery state of charge.

3.2.1.1 Main Variables

The main variables involved in the Flexibility Estimator algorithm calculations are listed below:

- Forecast: predictions for the system's inputs.
  - Demand Forecast: consumption prediction for the time period studied (usually 24 hours).
  - PV prediction: photovoltaic production predicted for the time period.
- Requirements: desired system behaviour, provided by the Scheduler module.
  - Power setpoint: desired power consumption/injection from/to the grid.
  - Reserved Energy: spared battery capacity for regulation purposes.

- Internal variables: calculation variables representing the behaviour of the system.
    - Battery power: power flow through the battery.
    - Inverter power: ESS resulting behaviour.
- Flexibility: capability of the ESS for variating the quantity and direction of the user energy flow, within certain limits.
    - Meter Forecast: estimated power consumption of the system.
    - Power variation: available power for modifying the system output.
    - SOC: amount of energy stored in the battery.

3.2.1.2 Algorithm Architecture

The calculation process is structured as shown in Figure 6. The Flexibility Estimator takes the forecast and requirements inputs and calculates the optimal solution for the ESS behaviour. The flexibility variables for this solution are obtained taking into account the system capabilities.

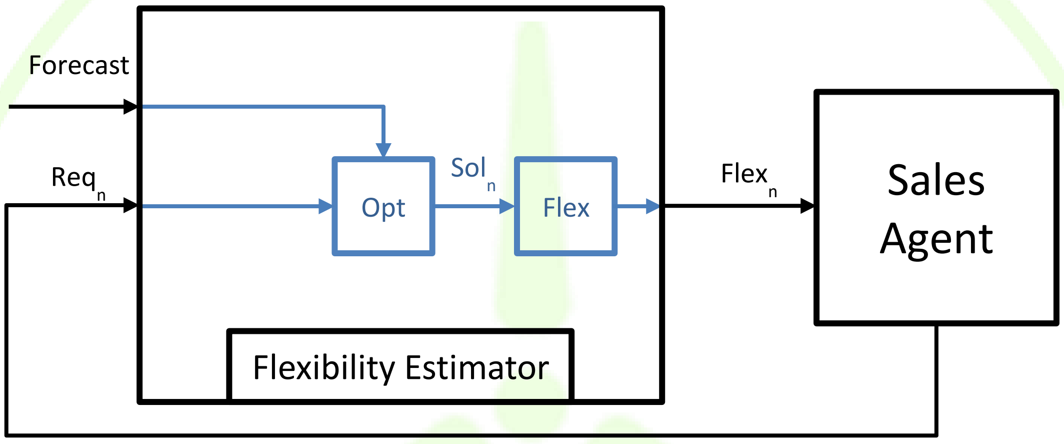


Figure 6 - Algorithm diagram flow

The calculated flexibility of the system is provided to the Sales Agent. The Sales Agent will consider the energy market needs and determine the best way to dispose of the system flexibility. The offering of capacities will result in a set of specifications for the system, expressed with the requirements variables coming from the Scheduler.

Since the flexibility depends on the specific requirements, once the Scheduler establishes them, it will be necessary to recalculate the algorithm in order to ensure the requirements are feasible and to obtain the new flexibility values. Therefore, the whole process of optimizing the system schedule will be an iterative calculation, as it is represented in Figure 6 by the feedback connection of the requirement variables. For easing and accelerating the iterative process, the battery SOC (State of Charge) evolution is included among the flexibility variables. With this information, the Scheduler can also consider the available energy when determining the requirements, so it is possible to know in advance if they can be accomplished. Still, it will be necessary to perform the iterative calculation, since the Flexibility Estimator takes into account several other variables and conditions that affect the final behaviour of the system.

3.2.2 Description and development

Input variables for the flexibility algorithm are arrays that cover the entire period under study. They will be classified in two groups, corresponding to the ones described below:

- Forecast:
    - P_{load}: Expected demand of the house (kW).
    - P_{PV} : Predicted PV production (kW).

- P_{PVunc}: Uncontrolled PV production predicted (kW). If the whole system includes an external PV inverter that cannot be controlled, the predicted PV production of this subsystem has to be treated separately.

- Requirements:
  - P_{req}: Energy Meter power setpoint (kW).
  - E_{req}: Energy reserve for ancillary services, such as frequency and voltage regulation (kWh). It can be specified in two ways:
    - $[\Delta P_{peak};\ \mathrm{dt}]$: Power peak requirement and its maximum duration [kW, h].
    - $E_{req_{in}}$: Energy reserve directly specified as an input (kWh).

Additionally, some input parameters are needed:

- Parameters:
  - SOC_0: Initial State of Charge of the battery (kWh).
  - SOC_M: Maximum State of Charge of the battery (kWh).
  - t: Timestep corresponding to the input arrays (h).
  - P_{Minv} : Nominal inverter power (kW).
  - P_{Mbat} : Maximum battery power (kW).

The output of the algorithm are the flexibility variables, which will be expressed as arrays corresponding to the input ones:

- Flexibility:
  - P_{Meter}: Expected Power consumption from the grid (kW).
  - SOC: Expected State of Charge of the battery (kWh).
  - Q_{av}: Available reactive power (kVAr).
  - P_{inj} : Available power decrement over the P_{Meter} (kW).
  - P_{load}: Available power increment over the P_{Meter} (kW).

3.2.2.1 Calculations

Each calculation step corresponds to a variable, that could be one of the flexibility variables or an internal variable of the algorithm. Therefore, they will be expressed as a definition formula followed by the corresponding restrictions (blue equations) that apply to each variable.

First, for the energy reserve, it may be possible that is stated as a combination of two variables: required energy and required peak power (and its duration). In that case, the energy will be calculated as follows:

$$E_{req} = E_{req_{in}} + \Delta P_{peak} \cdot dt \tag{1}$$

The following internal variables will be needed before finally obtaining the flexibility results:

- Internal variables:
  - $P_{loadeff}$: Effective load after considering the requirements and any uncontrolled PV source power (kW).

$$P_{load_{eff}} = P_{load} - P_{req} - P_{PV_{unc}} \tag{2}$$

  - P_{bat}: Battery power (kW).

$$P_{bat} = P_{PV} - P_{load_{eff}}$$
$$-\left(P_{M_{inv}} - P_{PV}\right) < P_{bat} < P_{m_{bat}} \tag{3}$$

  - P_{PVlim} : Actual used PV power (kW).

$$P_{PV_{\text{lim}}} = P_{PV}$$
$$0 < P_{PV_{\text{lim}}} < \mathrm{P}_{\text{load}_{\text{eff}}} + P\_bat \tag{4}$$

  - P_{inv}: Inverter active power (kW).

$$P_{inv} = P_{bat} - P_{PV_{\text{lim}}}$$
$$-P_{M_{inv}} < P_{inv} < P\_M_{bat} \tag{5}$$

  - P_{PVex} : PV power excess (kW).

$$P_{PV_{ex}} = P_{PV} - P_{load_{eff}} \qquad (6)$$
$$P_{PV_{ex}} > 0$$

- SOCresmin: Minimum charge to accomplish the energy requirements (kWh).

$$SOC_{res_{\min}} = \sum_{t}^{t_f} \left(E_{req} - P_{PV_{ex}} \Delta t\right) \qquad (7)$$
$$0 < SOC_{res_{\min}} < SOC_M$$

- SOCresmax: Maximum charge to accomplish the energy requirements (kWh).

$$SOC_{res_{\max}} = SOC_M + \sum_{t}^{t_f} \left(E_{req} + P_{load_{eff}} \Delta t\right) \qquad (8)$$
$$0 < SOC_{res_{\max}} < SOC_M$$

With these variables, it is possible to obtain the flexibility variables as follows:

$$SOC = \int_0^t P_{bat}\, dt \qquad (9)$$
$$SOC_{res_{\min}} < SOC < SOC_{res_{\max}}$$

$$P_{Meter} = P_{load} - P_{PV_{unc}} + P_{inv} \qquad (10)$$

$$Q_{av} = \sqrt{P_{M_{inv}}^2 - P_{inv}^2}$$
$$\Delta \mathrm{P_{inj}} = -\left(P_{M_{inv}} + P_{inv}\right) \; if\; SOC > 0 \qquad (11)$$
$$\Delta P_{load} = -P_{inv} + \left[P_{M_{bat}}\right] if\; SOC < SOC_M$$

3.2.2.1 Implementation

This section will present, in a detailed way, the calculation steps and the way they should be implemented. Most of the variables stated in the description of the flexibility algorithm will be expressed as arrays, containing the values corresponding to each timestep over the period being studied. Typically, calculations will be performed for a time span of 24 hours, corresponding to a natural day, with a timestep of one hour.

Most of the calculations can be directly implemented using the complete arrays, since they correspond to instantaneous power balances that don't depend on the previous steps. On the other hand, calculations involving integral or derivative operations should be implemented in a recursive step-by-step basis.

3.2.2.1.1 Symbology Notes

- Even though most of the equations may be expressed as vectorial operations for simplicity, all of them should be understood as array-component-level operations (e.g. a product won't be implemented following the rules of matrix products but as a product of each pair of components of the arrays involved, resulting in an array of the same size).
- After each calculation, some of the results should be constrained to the limits described in the previous section. In the equations, this is represented by embracing the value to constrain with square brackets, indicating the lower and upper limits. This representation is also used for the values that are restricted inside an equation, not only for its final result. Therefore, if the result from the calculation inside the brackets is greater than the upper limit, the value will be equal to that limit, and vice versa.

3.2.2.1.2 Calculation Steps

First of all, as it was seen before, the flexibility algorithm allows the possibility of specifying energy reserve requirements in two different ways. Internally, the combination of both has to be obtained in order to use it for the algorithm calculations:

$$\vec{E}_{req} = \vec{E}_{req_{in}} + \overrightarrow{\Delta P}_{peak} \cdot dt \tag{12}$$

Next follows the calculation of the straightforward internal variables:

$$\vec{P}_{load_{eff}} = \vec{P}_{load} - \vec{P}_{req} - \vec{P}_{PV_{unc}} \tag{13}$$

$$\vec{P}_{PV_{ex}} = \left[\vec{P}_{PV} - \vec{P}_{load_{eff}}\right]_{0}^{+\infty} \tag{14}$$

With them, a provisional battery power value can be obtained:

$$\vec{P}_{bat_0} = \left[\vec{P}_{PV} - \vec{P}_{load_{eff}}\right]_{-\left(P_{M_{inv}} - \vec{P}_{PV}\right)}^{P_{M_{bat}}} \tag{15}$$

The next variable to be calculated will be the state of charge evolution, but, since its limits depend on the energy reserve requirements, it is necessary to calculate them previously. In this case, the limit values depend on the expected evolution of the system behaviour from the corresponding timestep to the end of the day. This is represented in equations (7) and (8) with the limits of the sum being t and t_f. In the implementation of these equations, this means that the calculation has to be performed for each value of the array individually and starting from the final values to the first ones. The constraints will be applied for each step, so the value used for the next step will be already constrained. The resulting formulas are as follows:

$$\overrightarrow{SOC}_{res_{\min}}[n] = \left[\overrightarrow{SOC}_{res_{\min}}[n+1] + \left[\vec{E}_{req}[n]\right]_{0}^{+\infty} - \vec{P}_{PV_{ex}}[n]\Delta t\right]_{0}^{SOC_M} \tag{16}$$

$$\overrightarrow{SOC}_{res_{\max}}[n] = \left[\overrightarrow{SOC}_{res_{\max}}[n+1] + \left[\vec{E}_{req}[n]\right]_{-\infty}^{0} + \vec{P}_{load_{eff}}[n]\Delta t\right]_{0}^{SOC_M} \tag{17}$$

$$n = (N-1), \dots, 0$$
$$\overrightarrow{SOC}_{res_{\min}}[N] = 0$$
$$\overrightarrow{SOC}_{res_{\max}}[N] = SOC_M$$

being N the size of the input arrays of the flexibility algorithm.

With the battery power and the state of charge limits, it is possible to obtain the state of charge evolution along the time period. Again, the integral of this variable's definition (equation 9) will result in a step-by-step calculation. The constraints will also be applied for each step. Therefore, the resulting formula is:

$$\overrightarrow{SOC}[n] = \left[\overrightarrow{SOC}[n-1] + \vec{P}_{bat}[n]\Delta t\right]_{\overrightarrow{SOC}_{res_{\min}}[n]}^{\overrightarrow{SOC}_{res_{\max}}[n]} \tag{18}$$
$$n = 0, \dots, (N-1)$$
$$\overrightarrow{SOC}[-1] = SOC_0$$

The definitive battery power will be recalculated as the derivative of the state of charge:

$$\vec{P}_{bat}[n] = \frac{\overrightarrow{SOC}[n] - \overrightarrow{SOC}[n-1]}{\Delta t} \tag{19}$$

$$n = 0, \dots, (N-1)$$

The remaining variables calculation is straightforward from this point, applying the corresponding restrictions. The internal variables are obtained from the following formulas:

$$\vec{P}_{PV_{\lim}} = \left[\vec{P}_{PV}\right]_0^{\vec{P}_{load_{eff}}+\vec{P}_{bat}} \tag{20}$$

$$\vec{P}_{inv} = \left[\vec{P}_{bat} - \vec{P}_{PV_{\lim}}\right]_{-P_{M_{inv}}}^{P_{M_{bat}}} \tag{21}$$

Finally, the resulting flexibility variables (apart from the SOC, that has been obtained at equation (18) are calculated as follows:

$$\vec{P}_{Meter} = \vec{P}_{load} - \vec{P}_{PV_{unc}} + \vec{P}_{inv} \tag{22}$$

$$\begin{gathered}
\vec{Q}_{disp} = \sqrt{\vec{P}^2_{M_{inv}} - \vec{P}^2_{inv}} \\
\overrightarrow{\Delta P}_{inj} = \begin{cases} -\left(\vec{P}_{M_{inv}} + \vec{P}_{inv}\right) & if\ \overrightarrow{SOC} > 0 \\ 0 & if\ \overrightarrow{SOC} = 0 \end{cases} \\
\overrightarrow{\Delta P}_{load} = \begin{cases} -\vec{P}_{inv} + \vec{P}_{M_{bat}} & if\ \overrightarrow{SOC} > 0 \\ -\vec{P}_{inv} & if\ \overrightarrow{SOC} = 0 \end{cases}
\end{gathered} \tag{23}$$

Figure 7 presents a schematic representation of the calculation steps.

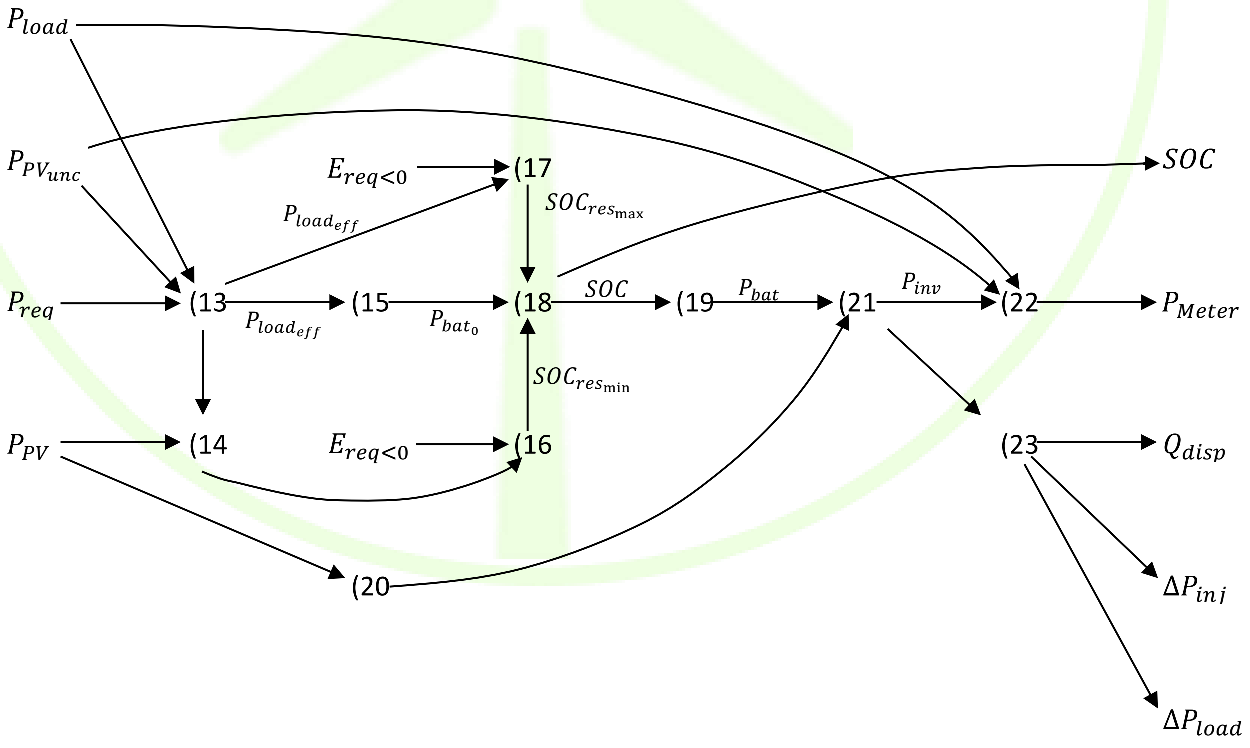


Figure 7 – Algorithm Implementation calculation scheme

3.2.3 Interfaces

The variables used for the flexibility estimation (inputs, outputs and parameters) are summarized in Table 9.

Table 9 – Summary of variables used for flexibility estimation

Variable Summary			
	Name	**Description**	**Unit**
INPUTS	P_{load}	Expected demand of the house	kW
	P_{PV}	Predicted PV production	kW
	P_{PVunc}	Uncontrolled PV production predicted. If the whole system includes an external PV inverter that cannot be controlled, the predicted PV production of this subsystem has to be treated separately	kW
	SOC_0	Initial State of Charge of the battery	kWh
	P_{req}	Energy Meter power setpoint	kW
	E_{req}	Energy reserve for regulation	kWh
PARAMETERES	t	Timestep corresponding to the input arrays	h
	SOC_M	Maximum State of Charge of the battery	kWh
	P_{Minv}	Nominal inverter power	kW
	P_{Mbat}	Maximum battery power	kW
	P_{Meter}	Expected Power consumption from the grid	kW
OUTPUTS	SOC	Expected State of Charge of the battery	kWh
	Q_{av}	Available reactive power	kVAr
	P_{inj}	Available power decrement over the PMeter	kW
	P_{load}	Available power increment over the Pmeter	kW

3.3 SCHEDULER MODULE

3.3.1 Functional objective

The Scheduler is the module in order to control the different components of the WG StaaS/VPP. It receives schedules from the Sales Agent and calculates the power setpoint for each asset.

3.3.2 Description and development

The Scheduler is designed to be run in an iterative loop in conjunction with the Flexibility Estimator, Sales Agent, the battery system internal controller and the RT monitor as shown in Figure 8. There is a slow (bold black) and a fast (bold pink) control loop which have following purposes:

Slow loop (executed every about 15 min): Flexibility Estimator calculates a new flexibility estimate for Sales Agent and Scheduler. Scheduler calculates a new schedule and feeds its back into the operational database.

Fast loop (executed every about 10 s): Scheduler sends a new set of target values to the batteries, which try as best as they can to fulfil these requests. However, in certain cases at least one battery will fail to fully fulfil the request, so in the next loop the Scheduler needs to request slightly more power from the other batteries.

In the first version of the Scheduler, the focus is exclusively on scheduling offsets to the battery-internal self-consumption-controllers. These offsets lead to extra power injection or consumption at the metering points of the storage facilities. Other modes of operation, e.g. PvsF, PvsV or QvsV for frequency and voltage support purposes are considered in a later stage as well.

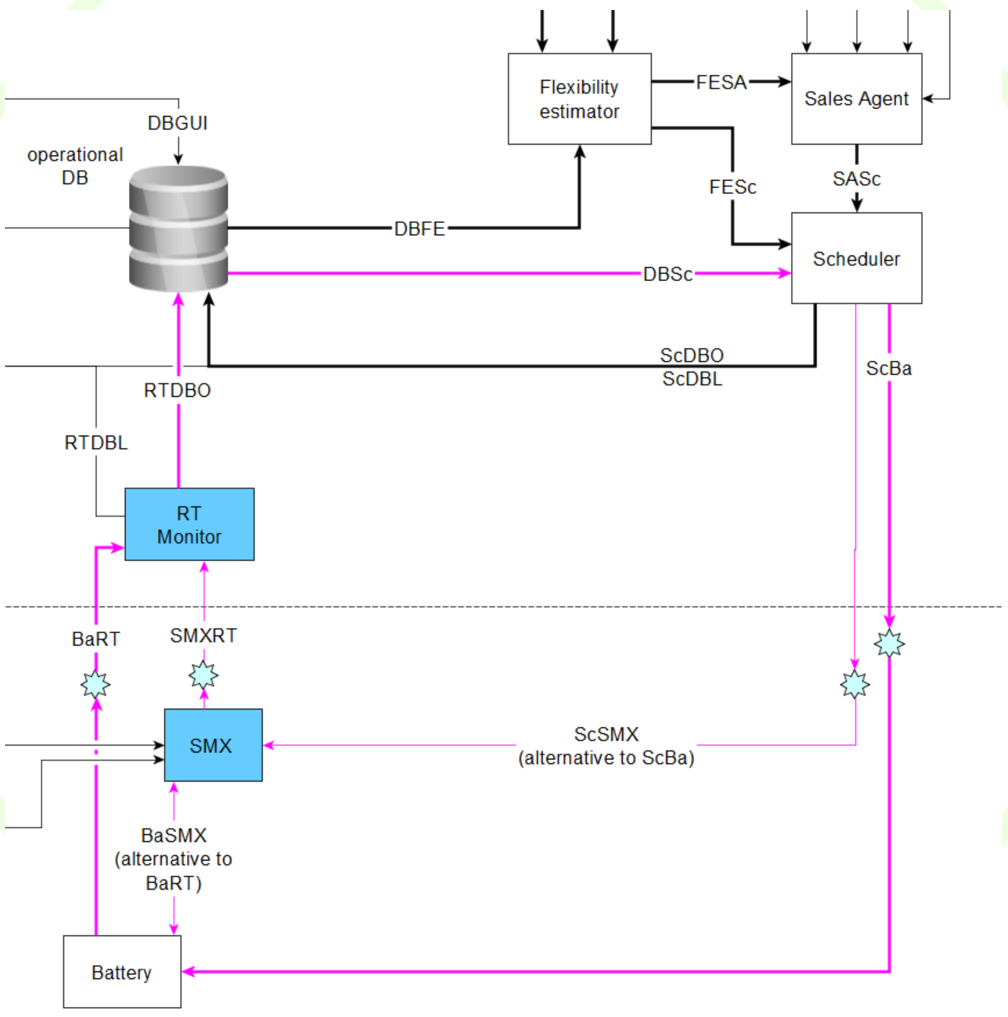


Figure 8 – Sketch of the data flow between Scheduler, Flexibility Estimator and batteries[2]

3.3.2.1 Calculation of outputs from inputs for offset power control

[2] The slow sloop is depicted as thick black lines, the fast loop as pink lines (thick pink: main data flow; thin pink: alternative data flow via the SMX). Light blue stars: communication flow via the WG IOP

The first version of the Scheduler is trying to put equal loads on all storage units, where “equal” means “an equal proportion of the current flexibility”, i.e., two storage units are considered to be equally loaded when they operate at equal amounts of their respective flexibilities.

3.3.2.1.1 Slow loop

Based on the estimated charge and discharge flexibilities of each asset and the aggregated VPP flexibilities the power setpoint for each asset in the specific 15 minutes interval is calculated by the following procedure

```
for every timestamp in channel ScDBO:
    for every storage unit in all connected storage units:
        if isInjection($P_{VPP}$):
            $P = \Delta P_{inj,su} / \Delta P_{inj,VPP} * P_{VPP}$
    else:
            $P = \Delta P_{load,su} / \Delta P_{load,VPP} * P_{VPP}$
```

where $\Delta P_{inj,su}$, $\Delta P_{load,su}$, $\Delta P_{inj,VPP}$, $\Delta P_{load,VPP}$ are the flexibilities for this storage unit for injection, load, and for the complete VPP for injection, load, respectively.

```
bool ifInjection(double PVpp):
    return (PVpp > 0)
```

3.3.2.1.2 Fast Loop

The fast loop is only executed if the current target P_{VPP} is not equal to zero. $P_{VPP} = 0$ implies that all storage units are permitted to run their internal self-consumption controllers without any offset.

Parameters

K_p : configuration parameter for proportional controller

P_{VPP}: target P_{VPP} for current timestamp as communicated by the Sales Agent

Step 1: get new setpoint

$P_{VPP,achieved} = \mathrm{sum}(Meter\_Active\_Power_{su})$: sum over all storage units, actually achieved power offset for the complete VPP, Meter_Active_Power as in MQTT data model

$P_{VPP,setpoint,new} = P_{VPP,setpoint,old} + K_p * (P_{VPP} - P_{VPP,achieved})$

$P_{VPP,setpoint,old} = P_{VPP,setpoint,new}$: stored for next loop

Step 2: distribute new setpoint among all connected storage units

for every storage unit in all connected storage units:

```
    if isInjection($P_{\mathrm{VPP,setpoint,new}}$):
        $P = \Delta P_{\mathrm{inj,su}} / \Delta P_{\mathrm{inj,VPP}} * P_{\mathrm{VPP,setpoint,new}}$
    else:
        $P = \Delta P_{\mathrm{load,su}} / \Delta P_{\mathrm{load,VPP}} * P_{\mathrm{VPP,setpoint,new}}$
```

Step 3: send new P-setpoints to all storage units

See Ch. 3.3.3.1

3.3.3 Interfaces

The Scheduler has the following inputs (labels as in Figure 8):

- **SASc**: Timeline of (future) target values (power offset, PvsF, ...) to be achieved by the VPP as sold to the markets by the Sales Agent. Data structure: One 2D-array with columns *timestamp*, P_{VPP} (first version of Scheduler) + probably some other columns, e.g., *PvsF* depending on design of Sales Agent (second version of Scheduler).
- **FESc:** Timeline of (future) flexibilities of individual storage units + of the complete VPP as determined by the flexibility estimator. One 2D-array per storage unit + one 2D-array for the complete VPP, each with columns *timestamp, SOC,* P_{Meter}, Q_{disp}, ΔP_{inj}, ΔP_{load}
- **DBSc:** Currently achieved contribution of storage units to overall control goal. One 2D-array per storage unit with columns *timestamp, Meter_Active_Power* (as in MQTT data model) (first version of Scheduler) + probably some other columns from the MQTT data model (second version of Scheduler)

The scheduler will calculate and output the following outputs (labels as in Figure 8):

- **ScDBO:** Timeline of (future) target values (power offset, PvsF, ...) of individual storage units to be stored in the operational DB for later referral by the Flexibility Estimator. 2D-array per storage units with colums *timestamp, P* (first version of Scheduler) + probably some other columns, e.g., *PvsF* depending on design of Sales Agent (second version of Scheduler).
- **ScBa / ScSMX:** Current target values of individual storage units within fast loop. One MQTT-command per storage unit per loop iteration + one or more MQTT-commands upon system startup and shutdown: working mode = 0, control mode = 1, active = true, *P* (all as in MQTT data model). See example commands in Ch. 3.3.3.1. Depending on configuration of storage unit this signal eithers goes directly to the battery (VARTA, AMP) or to the battery via the SMX (BYES).

3.3.3.1 Example commands for ScBa / ScSMX for power offset control

On startup:

```
    {
        "Id": "some id 123154684",
        "Command": 0,
        "Params": {
              "WorkingMode": 1
        }
    }
```

In every fast control loop (value for P as calculated in Ch. 3.3.2.1):

```
      {
            "Id": "another id 4546564",
            "Command": 3,
            "Params": {
                  “ControlMode” : 1,
                  “active” : true,
                  “P” : 1234
            }
      }
```

3.4 SALES AGENT MODULE

3.4.1 Functional objective

This module is the interface to the DSO and different energy markets, e.g. wholesale and TSO Ancillary service market. In order to support the local grid, it processes the requests from the DSO tool WG Cockpit either directly or via a virtual DSO Ancillary Service Market. In the energy markets available capacities are offered for trading power/energy or supporting the frequency.

3.4.2 Description and development

In order to be able to provide an offer to the DSO or to the different energy markets enough capacities have to be available. Out of this one major task of the Sales Agent is the marketing evaluation of available capacities based on the flexibility estimator. In order to do so it will be checked if enough injectable or load power is available in each of the demanded time periods. If the evaluation is positive an offer is sent, otherwise not. The investigations in WiseGRID will primarily focus on the provision of services to the DSO. However in order to generate additional profit other markets like the Wholesale and TSO Ancillary Service Market can be considered in a later stage as well. For some services the implementation of an internal balancing strategy has to be considered.

3.4.2.1 DSO Ancillary service market/WG Cockpit

In case of a congestion in the grid the DSO will analyse the situation and publish a request for adapting the active or reactive power in a certain area. WG StaaS/VPP will process the request, evaluate the feasibility of providing capacities and – if possible - send an offer to the DSO. The offer includes the available power and a price for the capacities. Afterwards the DSO evaluates all received offers from WG StaaS/VPP and other WG tools or – in a product version - energy players and selects the most appropriate one. Here the price as well as the amount of provided power might be the most important criteria. Depending on the outcome of the evaluation an order or a rejection is sent to the WG StaaS/VPP. In case of unexpected deviations, e.g. from the expected flexibilities, an offer revocation can be performed as well.

The following Figure illustrates the data exchange procedure between the DSO tool WG Cockpit and WG StaaS/VPP.

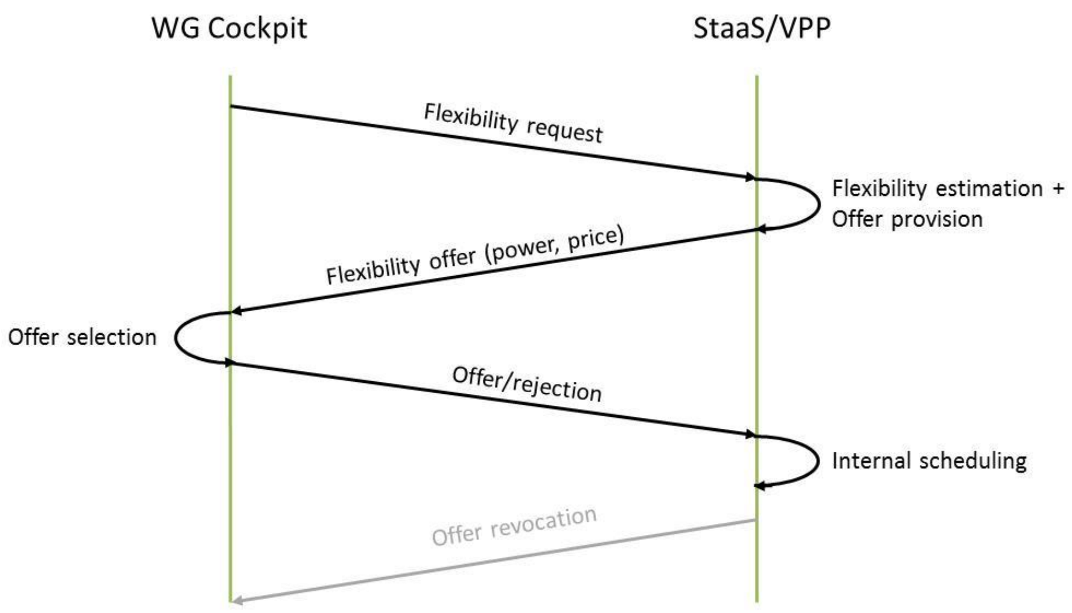


Figure 9 – Request and offer procedure between WG Cockpit and WG StaaS/VPP

Regarding the provision of offers, it has to be ensured that enough capacities are available in the demanded period. In order to evaluate this issue, the flexibility estimator is called in each of the intervals and step by step it is checked if enough flexible power is available. Afterwards the scheduler is called in order to determine the P_{req} of each single unit. Depending of the outcome a Boolean variable which states if enough capacities are available is set on true or false state. If enough free capacities are available an offer is sent to the DSO. The procedure can be described by the following formula.

for every timestamp t in the demanded period:

```
if number_of_requests_intervals > 1
      call flexibility estimator()
endif

if isInjection($P_{request\ DSO,\ t}$):
      if $\Delta P_{inj,VPP,t}$ > $P_{request\ DSO,\ t}$
            b_enough_capacities_available = true
            $P_{Vpp}$ = $P_{request\ DSO,\ t}$
      else:
            b_enough_capacities_available = false
      endif
```

```
        else
                if $\Delta P_{load,VPP,t}$ > $P_{request\ DSO,\ t}$
                        b_enough_capacities_available = true
                        $P_{Vpp}$ = $P_{request\ DSO,\ t}$
                else:
                        b_enough_capacities_available = false
                endif

        if number_of_requests_intervals > 1
                call scheduler()
        end if
end if
end for

if b_enough_capacities_available == true
        b_submit_price = true
else
        b_submit_price = false
```

b_enough_capacities_available: boolean stating if in the demanded periods enough capacities are available or not in order to meet the request

$P_{request\ DSO,\ t}$: requested power from DSO

P_{Vpp}: Total power that VPP has to provide

$\Delta P_{inj,VPP,t}$: total flexible power available for injection

$\Delta P_{load,VPP,t}$: total flexible power available for absorption

Here it is assumed that the aggregator just offers capacities if all intervals of the request can be covered. Another option would be to provide a partial. In that case *b_enough_capacities_available* must not be true.

3.4.2.2 Wholesale Market

The mechanisms behind the participation in the Wholesale Market are to a certain extend similar to the ones described before. However instead of receiving requests continuous market auctions are held. Based on the flexibility estimation WG StaaS/VPP sends a flexibility offer to the Intraday or Day-Ahead market. Depending of the bids of other energy players and the demand prices the offer is accepted or not. The underlying procedure is illustrated in the following Figure.

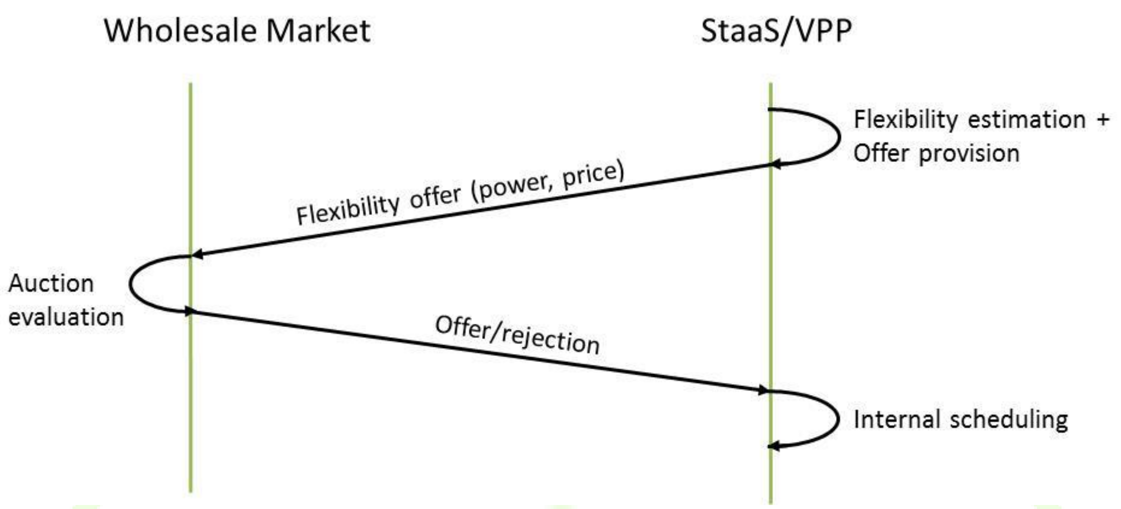


Figure 10 – Offer procedure for participating in the Wholesale market

In order to estimate if enough capacities are available for the formula presented in the chapter before can be used. Especially when high renewable production occurs low prices or even negative prices are expected in the market. For the aggregator and in the end-user this is very attractive since they get money for energy that be potentially used at a later point.

3.4.2.3 Ancillary service market

Another option in order to increase the profit for the aggregator and the end-users is the participation in the (TSO) Ancillary service market. Here between Frequency Containment Reserve (FCR) and Frequency Restoration Reserve (FRR) has to be distinguished. Comparable to the Whole Sale market tender or auction processes are often applied and energy market provide their bids. Since the periods of this service (e.g. a week) might be longer than the horizon of the forecast horizon (24 hours), it can't be fully estimated if enough capacities are available. Out of this reason it might be necessary to buy or sell energy on the Wholesale market in case too much or too less energy is available in the WG StaaS/VPP pool.

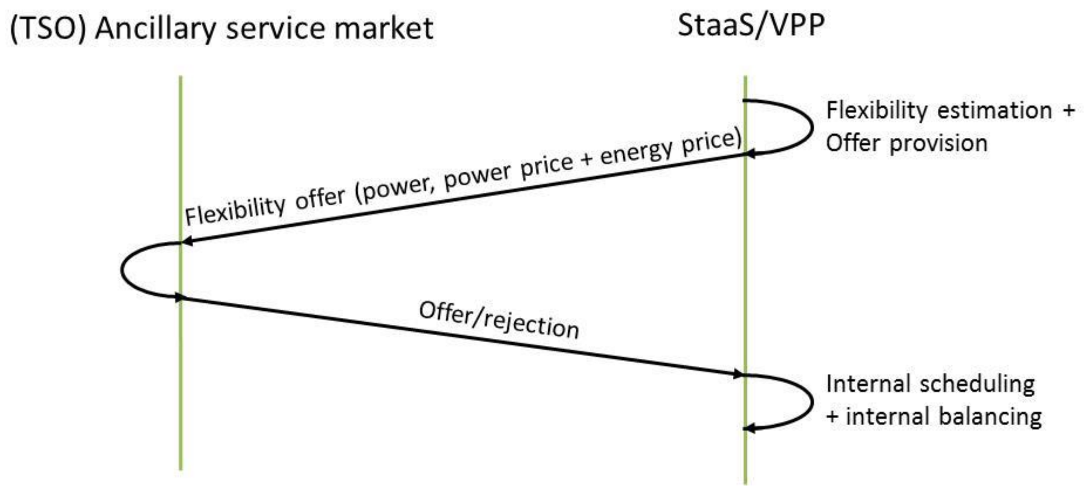

(TSO) Ancillary service market
StaaS/VPP
Flexibility estimation + Offer provision
Flexibility offer (power, power price + energy price)
Offer/rejection
Internal scheduling + internal balancing

Figure 11 - Offer procedure for participating in the Ancillary Service market

Regarding the estimation of the capacities in principle the same formula as in 3.4.2.1 can be applied for the full day. Instead of $P_{request\ DSO,\ t}$the expected necessary power for each interval based on the frequency deviations is set. However, this is a difficult task since the frequency deviations can't be forecasted and the uncertainties are high.

In case the available injected power is too low additional energy has to be bought from the Wholesale market. On the opposite if not enough load is available power has to be sold. Again the internal balancing estimation has to be performed throughout the whole interval.

for every timestamp t in the demanded period:

```
    if number_of_requests_intervals > 1
        call flexibility estimator()
    endif

    if isInjection($P_{FCR, expected}$):
        if $\Delta P_{inj,VPP,t}$ < $P_{FCR, expected,t}$
            $P_{Vpp}$ = $P_{FCR, expected,t} + P_{buy,t}$
        else:
            $P_{Vpp}$ = $P_{FCR, expected,t}$
        endif
    else
        if $\Delta P_{load,VPP,t}$ < $P_{FCR, expected,t}$
            $P_{Vpp}$ = $P_{FCR, expected,t} + P_{sell,t}$
        else:
            $P_{Vpp}$ = $P_{FCR, expected,t}$
        endif
    endif

    if number_of_requests_intervals > 1
        call scheduler()
    endif

endfor
```

$P_{FCR, expected,t}$: power in interval t based on expected frequency deviation. In that case the worst case scenario in both directions should be covered which is equal to the maximum power that has to be provided

$P_{buy,t}$: power that has to be bought from the Wholesale market in order to meet requirement of service provision

$P_{sell,t}$: power that has to be sold from the Wholesale market in order to meet requirement of service provision

Depending on the type of Ancillary service not only power but also energy prices have to be offered. Depending on the requirement of the service the operation strategy of the WG StaaS/VPP is affected. If power has to be provided in both directions it might make sense to set the average SOC of the battery storage systems to a certain level.

3.4.2.4 Bidding

For the calculation of offers in the markets the usage of the battery storage system as well as the PV and the connected operation cost have to be taken into account. In order to do so the levelized cost of storage (LCOS) that represent the cost per produced or stored kWh can be considered here. The LCOS can be calculated as follows:

$$LCOS = \frac{CAPEX + \sum_{t=1}^{t=n} \frac{A_t}{(1+i)^t}}{\sum_{t=1}^{t=n} \frac{W_{out}}{(1+i)^t}} \tag{24}$$

$CAPEX$: capital expensures/invest for stationary stoage system

A_t: yearly expenses due to maintenance and and other cost (e.g. recharging cost)

W_{out}: provided energy of the storage system

i: interest rate

n: expected lifetime of the battery, here 20 years

Since a lot of circumstances have to be considered the offers in the different markets will be manually produced in a first step in order to analyse the effect on the profit generation and the operation of the VPP.

3.4.3 Interfaces

The Sales Agent is closely linked to the flexibility estimator, the scheduler as well as the market modules respectively directly to the DSO. The dependencies can be seen in Figure 12.

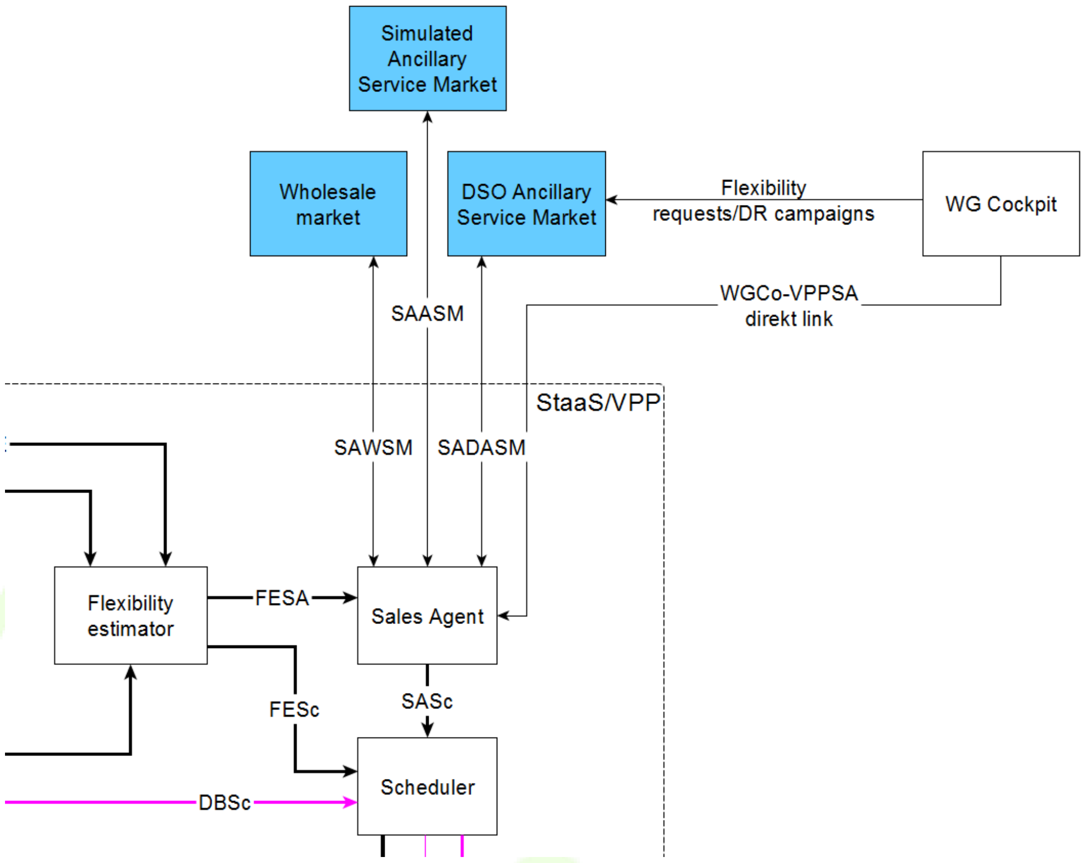


Figure 12 – Sales Agent inputs and outputs

The Sales Agent has the following inputs:

- **SAWSM**: Timeline of wholesale market prices and of received awards respectively the power to provide by the VPP. One 2D-array with columns *timestamp,* $p_{\text{Day-Ahead}}$, p_{Intraday}, P_{VPP}.
- **SAASM:** Timeline of Ancillary service market prices for awarded power distinguishing between Frequency Containment Reserve (FCR) and Frequency Restoration Reserve (FRR). One 2D-array with columns *timestamp* as well as the awarded prices for FCR p_{FCR} and FRR p_{FRR}.
- **SADASM:** One 2D-array with *timestamp,* necessary active power P_{vsV}, necessary reactive power Q_{vsV}, latitude lat_{request} and longitude $long_{\text{request}}$ of the request are send to StaaS/VPP. StaaS/VPP sends an array with price and power back to the DSO. After an offer selection the offer or a rejection is send to StaaS/VPP.
- **WGCo-VPPSA:** this is a direct link between the DSO and the StaaS/VPP assuming that no market is I place. In a 2D-array StaaS/VPP receives similar information like *timestamp*, necessary active $P_{\text{vsV,}}$ and reactive power $Q_{\text{vsV,}}$ as well as latitude lat_{request} and longitude $long_{\text{request}}$ of the request. The exchange of price information for active p_{PvsV} and reactive p_{QvsV} power are considered here as well.
- **FESA:** Timeline of (future) flexibilities of the complete VPP as determined by the flexibility estimator. One 2D-array for the complete VPP, each with columns *timestamp, SOC,* P_{Meter}, Q_{disp}, ΔP_{inj}, ΔP_{load}.

The Sales Agent has the following outputs:

- **SASc:** Timeline of (future) target values (e.g power offset, PvsF, ...) to be achieved by the VPP as sold to the markets by the Sales Agent. Data structure: One 2D-array with columns *timestamp*, P_{VPP} (first version of Scheduler) + probably some other columns, e.g., *PvsF* depending on design of Sales Agent (second version of Scheduler).
- **SAWSM:** Timeline of offers (poer and price) and demands in the Wholesale market. One 2D-array including *timestamp,* $P_{Wholesale,\mathrm{demand}}$, $P_{Wholesale,\mathrm{offer}}$, $p_{Wholesale,\mathrm{demand}}$ *and* $p_{Wholesale,\mathrm{offer}}$.
- **SAASM:** Timeline of offers to the Ancillary service market, again distinguishing between FCR and FRR market. One 2D-array including *timestamp* as well as offered power and prices for FCR $p_{\mathrm{FCR,offer}}$ and FRR $p_{\mathrm{FRR,offer}}$.
- **SADASM:** Timeline of local ancillary service market offers for specific DSO requests. One 2D-array including *timestamp*, offered active $P_{\mathrm{vsV,offered}}$ and reactive power $Q_{\mathrm{vsV},offered}$ and the linked prices for the provision $p_{\mathrm{PvsV,\ offered}}$ and $p_{\mathrm{QvsV,\ offered}}$.

3.5 BILLING MANAGEMENT MODULE

3.5.1 Functional objective

The billing management module aims at fitting the different business models that can be defined for the WG Staas / VPP application, to bill all the actors intervening on the business process.

This module is in interface with the long term data base to get all the input it would need to generate all the bills for the actor. On the same way, the other module will go on this long term data base to look for the bill it needs.

On the following sub-sections, the term of bill is the charge asked from the different actor to the WG Staas / VPP application but it is also the potential revenue given from the application to the other actor.

3.5.2 Description and development

The communications to the module are only done through the long term data base but the information are shared with more actors. The application supports the aggregator to sell unused battery capacity to different markets (e.g. ancillary service market, DSO ancillary service market and wholesale market). The battery owner is either directly connected to the application or connected through the WiseCORP and WiseHOME application. The bill is then directly given to the owner or provided through the WiseCORP / WiseHOME application.

The different actors can be summarized in the following diagram from Figure 13.

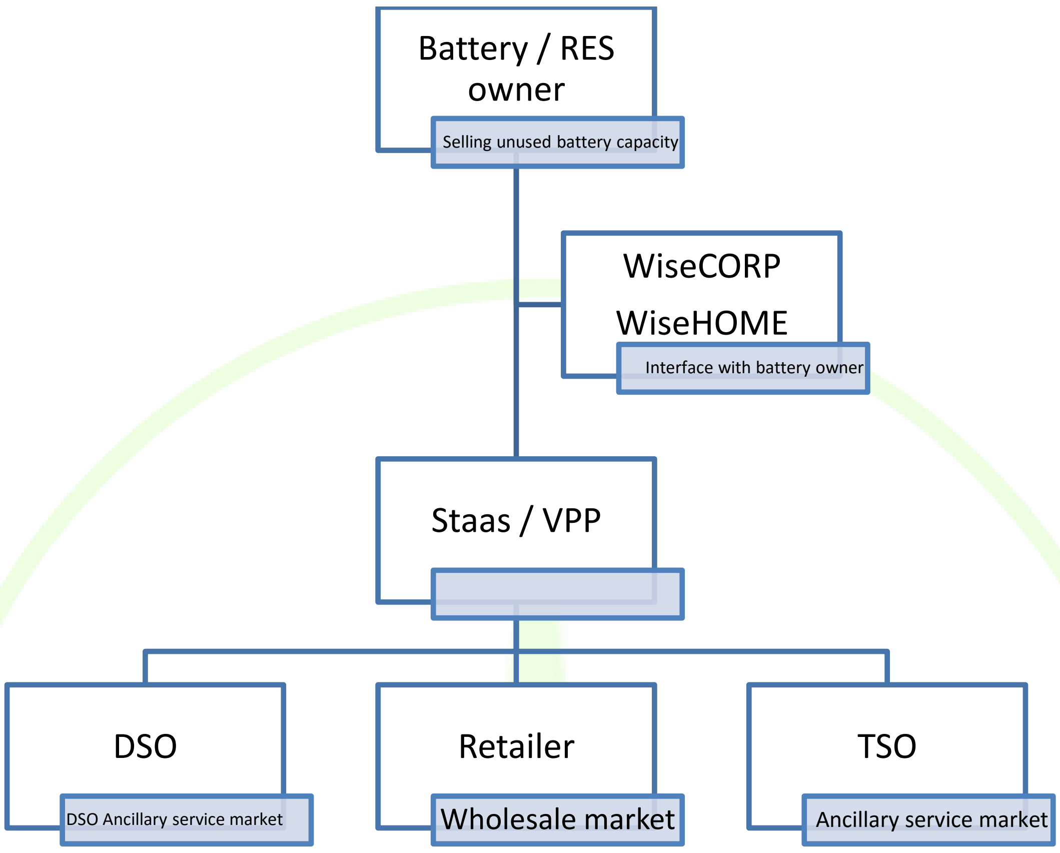


Figure 13 – Actors on WG StaaS/VPP market

Different kind of services can be sold depending on the actor. The ancillary service can be:

For the DSO:

- Reactive power compensation and voltage control

For the TSO:

- Frequency regulation

For the retailer in the Wholesale market, the service is a flexibility selling.

Since each service has its own way of billing, it is necessary to store the kind of services done on the long term data base. A specific bill is then generated depending on that service and is based on different variables of the system.

3.5.3 Interfaces

For each sale done, the client ID has been noted on the long term data base and can be used by the billing management system. It has to be noted that each actor is considered as a client. The bill is generated for the specific service on a certain amount of time, defined by the business model. For the report, a monthly bill is considered for each service, but this period can be adapted on the application to make the tool adaptable on every business models possibilities.

In term of billing management, on a generic way, the ancillary services are defined depending on:

- The power available: the power available is making value to the TSO, as a matter of fact, the bill depends on the power the application is able to give
- The availability duration on the considered period
- The energy absorbed
- The energy injected

It has to be noted that the previous terms are not compulsory on the bill but can be taken into account to fit a particular business model. Each ancillary service is a bit adapted depending on what is sold. The same structure is applied for the frequency regulation than for the voltage regulation. In one case active power is sold and on the other case, reactive power is sold.

It is clear that the business model is not fixed at any time and it is really important to find a way to adapt the billing management. The user should be able to personalize any generated bill. The proposal can be to apply a coefficient on each parameter that the user would be able to monitor and configure. That way, the application gives the possibility to each user to adapt any business model with the application.

$$Price = Coef_{power} \times P_{available} \times time_{available} + Coef_{Eabs} \times E_{Absorbed} + coef_{Einj} \times E_{injected} \quad \textbf{(25)}$$

Each coefficient is controlled by the user to apply different price depending on the situation. The bill is then shared with other WiseGRID application through the long term data base or directly to the user if the WiseGRID application is not used.

In our case, three applications are concerned:

- WG Cockpit for DSO / TSO
- WiseCORP for battery / RES owner
- WiseHOME for battery / RES owner

Each application would have the functionality to manage the bill generated.

It has to be noted that this kind of bill is also done for battery / owner compensation and depending on the contract, WG Staas / VPP owner is able to control the bill edited through the application.

The same kind of thought can be adapted for the wholesale market which depend on:

- The energy absorbed
- The energy injected

On a certain time, there are different prices per kWh depending on the moment of the energy trade. The bill is applying the addition of all the transaction done during the time period concerned by:

$$\sum E_{absorbed} \times price_{kWh_{absorbed}} + E_{injected} \times price_{kWh_{injected}} \quad (26)$$

The different price of the energy is given by the wholesale market. On a general way, for the ancillary service and for the flexibility service, all the coefficients of price are given to the billing management system by the sale agent through the long term database.

3.6 MQTT BROKER FOR COMMUNICATION SERVICES

In order to implement communication with the field devices (batteries and smart meters), and having into account the latest objective of having real-time observability of the devices, the MQTT protocol has been selected. This protocol is broadly adopted in the field of IoT devices and is therefore suited to allow monitoring of different kind of devices in a maintainable and scalable manner. This is favoured by the following characteristics:

- The protocol is suited for low-constrained devices, therefore can be applied to devices such as batteries and smart meters, the main type of devices monitored by WG StaaS/VPP
- The protocol allows different predefined Quality -Of-Service levels (*at most once*, *at least once*, *exactly once*), making it possible to adapt to different network constraints
- The protocol enforces the use of publish/subscribe communication mechanism, which allows decoupling the information producers and consumers, and introduces asynchronous communication among them

The general approach is that batteries and SMX (smart meters) regularly publish real-time status to the MQTT broker implemented within the WG IOP broker. On the WG StaaS/VPP side, the RT monitor is configured to track the changes in the status of the controlled devices and store the necessary history. The WG StaaS/VPP tool design also takes benefit of the existing connection of batteries with the WG IOP via MQTT to implement also the necessary control (via the Scheduler module).

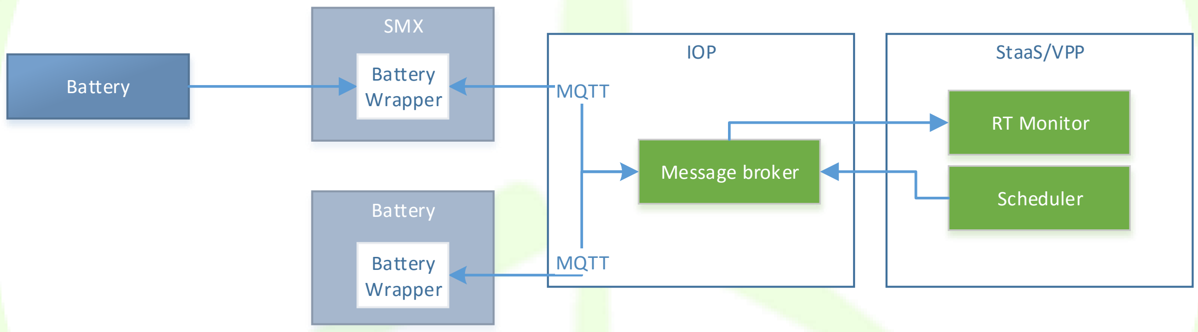


Figure 14 – Overview of communication of field devices with WG StaaS/VPP via WG IOP

A common data model has been designed to be followed by the battery wrappers integrating batteries from different vendors (VARTA, AMPERE and BYES batteries will be integrated in the project). This data model includes all necessary messages to be able to track the status of the batteries, and control the operation mode and parameters of those. The objective is that all Battery Wrappers communicate with the WiseGRID IOP and the different applications using this data model, while the actual translation to the vendor-specific protocols is implemented within the different Battery Wrappers and hidden from the other elements of the WiseGRID architecture. This way, any kind of battery can be operated regardless its actual vendor. The complete specification of this data model can be found in D4.2 [3].

3.7 MQTT INTERFACES

The interface from the battery to the WG StaaS/VPP is either done through the SMX PLC or through a direct connection between the battery and the WiseGRID Interoperable Platform (WG IOP):

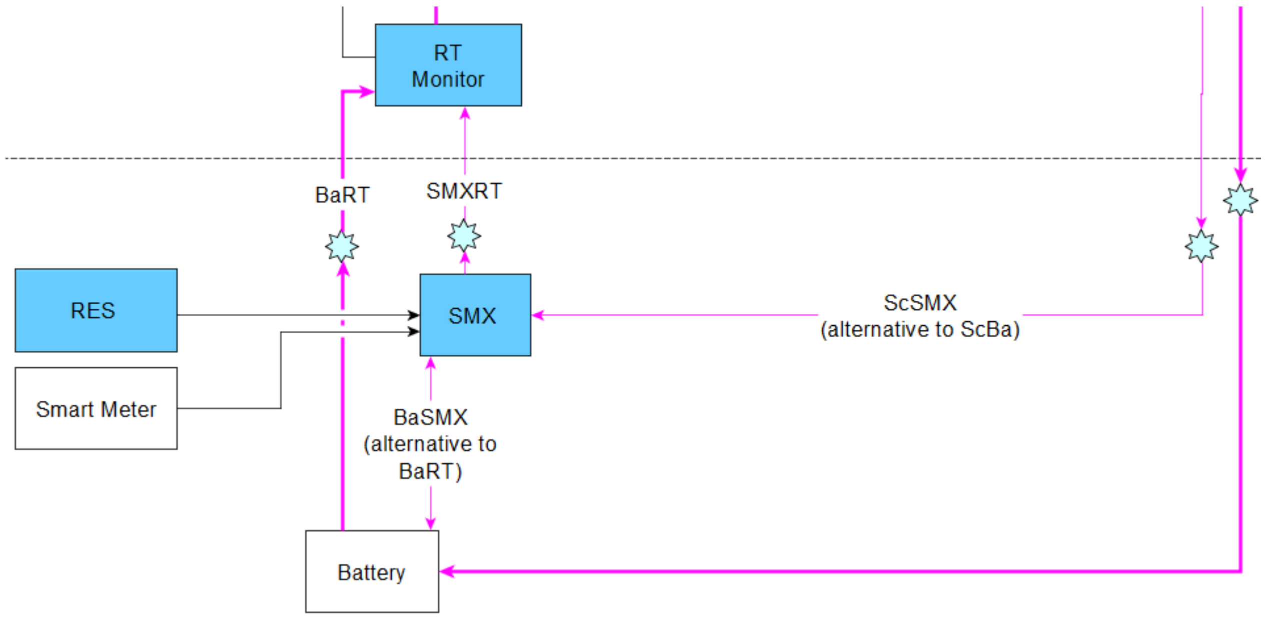


Figure 15 – Sketch of the interface between batteries and WG STaaS/VPP[3]

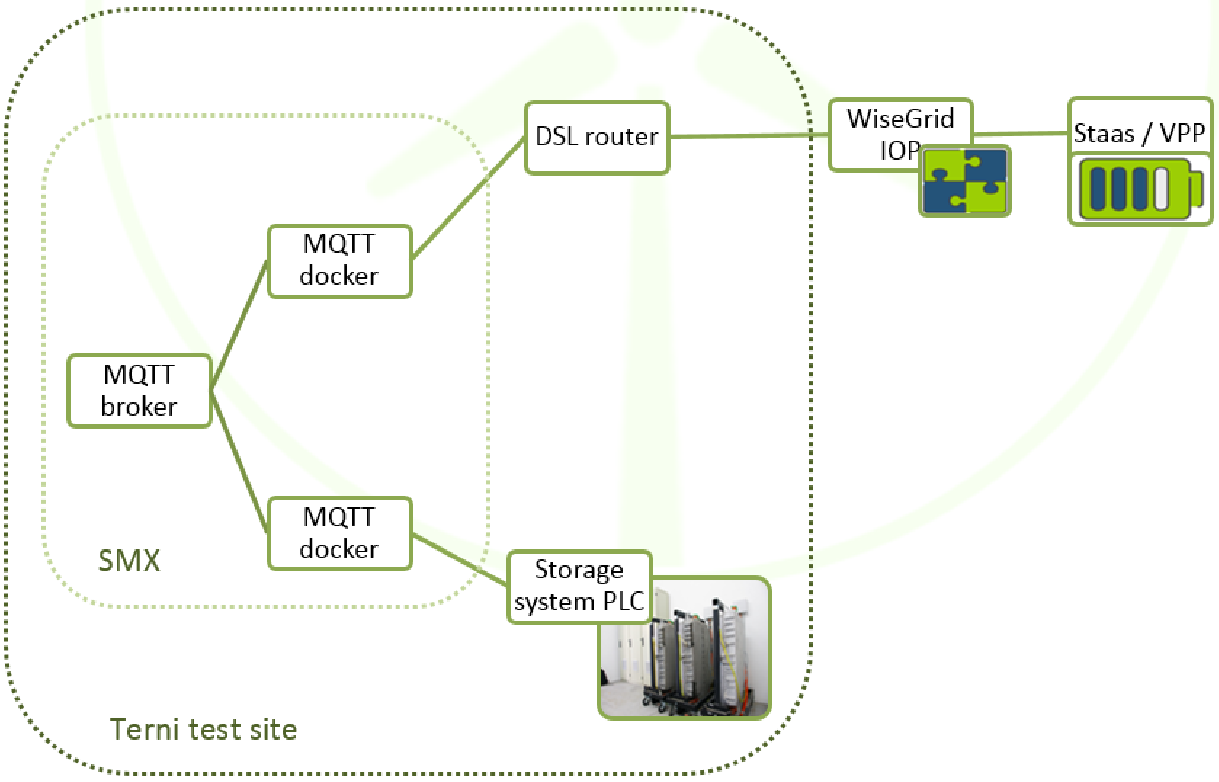


Figure 16 – Interface between WiseGRID and storage system through the SMX

[3] The interface is directly via the WG IOP's MQTT broker (depicted as light blue stars) or via the SMX and via the IOP from there

The aim of this interface with the WiseGRID application is to have a common data model for the battery storage system without regarding the supplier of the system. The MQTT messages are described in more detail in D4.2 Deliverable [3]. However, an excerpt can be found in this document, in Annex A.

3.8 REAL TIME MONITOR MODULE

3.8.1 Functional objective

The *RT monitor* is a horizontal module that will handle the data ingestion for most of the applications of the project. It has been designed in order to fulfill the requirements for data ingestion accordingly to the requirements and the architecture of communications proposed for the applications of the WiseGRID project.

3.8.2 Description and development

The *RT monitor* features the following characteristics:

Support for ESB and IOP communications

The module supports the main communication protocols proposed for the applications of the project: MQTT and AMQP. Use of these communication protocols show several advantages, as described in detail in D4.2 [3]. The RT monitor will therefore support the communication mechanisms deemed appropriate to fulfill the requirements of the applications of the project.

Maintenance of databases

The module automatically maintains the necessary data within the two different MongoDB databases proposed per application: the operational database and the long-term (big data) database.

- Operational DB keeps one document per device, stating last known status. Its main purpose is to hold a picture of the current status of the devices controlled by the application.
- Long-term DB keeps a history of changes per device, each change hold by a document. Its main purpose is serve as data source for KPI calculation and analysis.

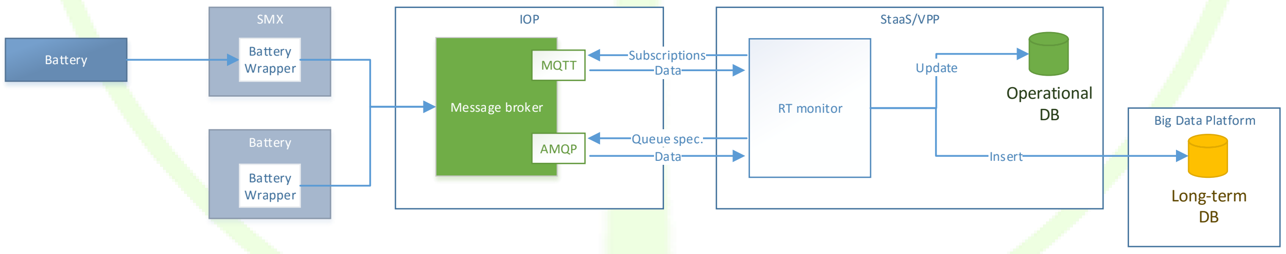


Figure 17 – Overview of the data flows for monitoring purposes

3.8.3 Interfaces

One of the main objectives behind the design of the *RT monitor* module is to allow its reusability among different applications of the project. It implements therefore interfaces to the main technologies used in the applications:

- Interface to MQTT broker: the *RT monitor* can subscribe to the necessary topics of a MQTT message broker (such as the one implemented within the WiseGRID IOP [3]), in order to track the status of those field devices using this IoT protocol to regularly publish their status.
- Interface to AMQP broker: the *RT monitor* can also subscribe to AMQP queues, the second main protocol considered in the architecture of the different WiseGRID applications. This protocol is used whenever the information to be transmitted to an application needs to be resilient and persistent against possible failures of the application.
- Interface to MongoDB databases: MongoDB has been selected as the proper database technology for both operational and long-term (big data) databases of the different applications of the project. The main purpose of the RT monitor is therefore to write the tracked information into these data-bases, making it available to other application modules.

3.9 COMMUNICATION WITH ENERGY STORAGE MODULE SYSTEM 1

3.9.1 Functional objective

The starting point for the definition of the communication with the energy storage module is the system developed in Terni for the ELSA project [4], which needs to be adapted for the WG Staas/VPP data model. In order to have a uniform communication to the battery storage system, it has been asked to develop the MQTT interface to communicate between the PLC and an SMX on which a MQTT broker is running. Basically, we integrated an MQTT client to the Beckhoff and tested it, using the SMX as an MQTT broker, and a computer as another MQTT client.

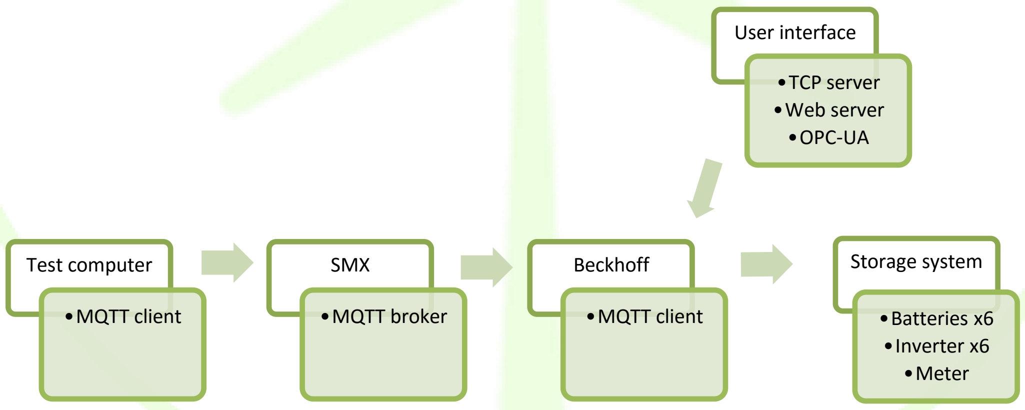


Figure 18 – Terni Storage scheme

3.9.2 Description and development

3.9.2.1 Brief Terni system description

The Terni storage system is composed of 6 Renault Kangoo batteries, 6 ABB ESI - S inverters, 1 Socomec Countis e43 energy/electric meter. All these equipments are connected to a Beckhoff C6920 (and a safety PLC CX5010). The batteries, inverters, and meter communicate to each other through the main PLC (C6920) using multiple protocols (inter alia: CAN, Modbus RTU) and communicate outside equipment through Modbus TCP, a webserver, an OPC-UA server. The MQTT interface is added to fit the WiseGRID requirements.

3.9.2.2 MQTT implementation on the Beckhoff PLC C6920

The PLC is programmed with TwinCAT 3 software (language IEC61131-3 Structured Text for Beckhoff). To help the interface implementation, libraries and license are made available. The MQTT library is used with the associated license called *TF6701-0260*. The *Tc3_IoTBase* library contains function blocks (FB), methods and structures in order to connect to a broker, publish messages on topics and subscribe to topics with different qualities of service.

In addition to this library, a JSON library is needed to encode and decode the JSON message, as well as two other programs which includes MQTT and JSON functionalities. All this structure has been set up on a test bench to implement the communication.

The publish function is called every second, meaning that every second, 30 messages are published (as there are 5 topics published per battery and 6 batteries).

3.9.2.3 Communication tests done

Before injecting the project into the PLC present on Terni Site, we have done some communication tests in order to validate the programming. To do so, we used the SMX (IP address: 172.31.41.10) as an MQTT broker, the Beckhoff PLC (IP address: 172.31.41.2) as an MQTT client and a computer (IP address: 172.31.41.1) as an MQTT client.

The “test bench” is displayed below, which shows a local network on which an SMX, a PLC and a computer are connected.

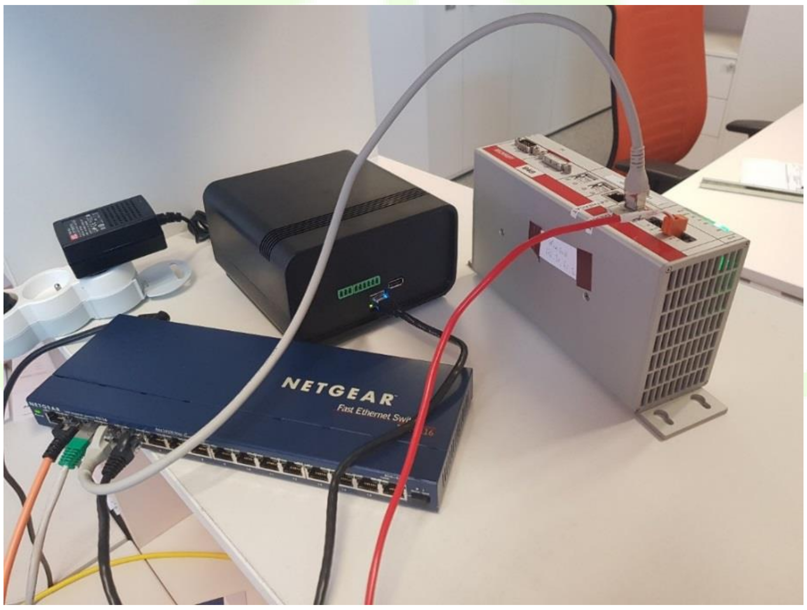
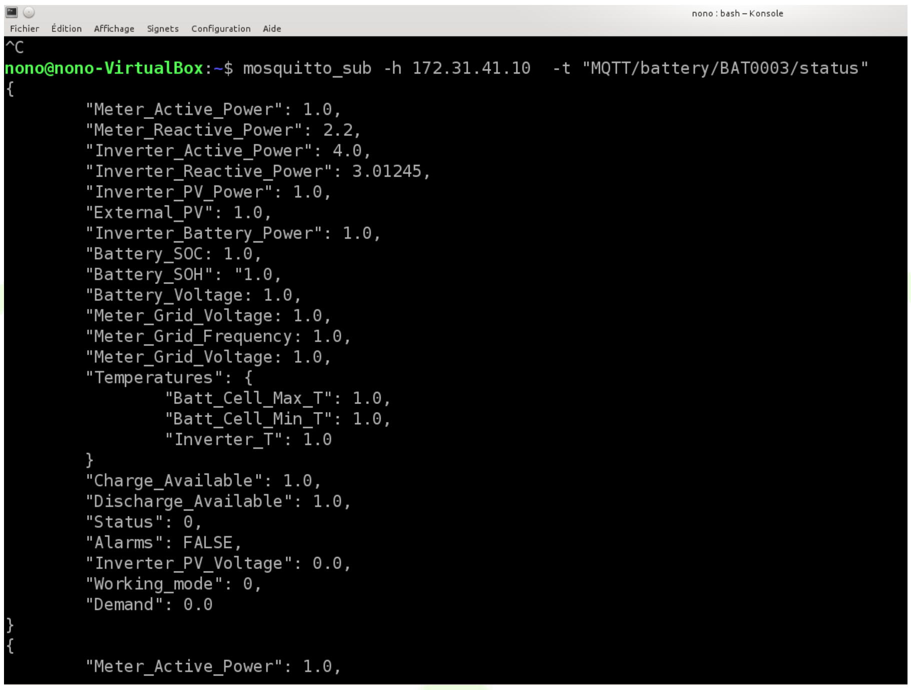


Figure 19 – Local network on which an SMX, a PLC and a computer are connected[4]

[4] The computer is not in the picture

3.9.2.4 Computer subscribing, PLC publishing (on the SMX) test

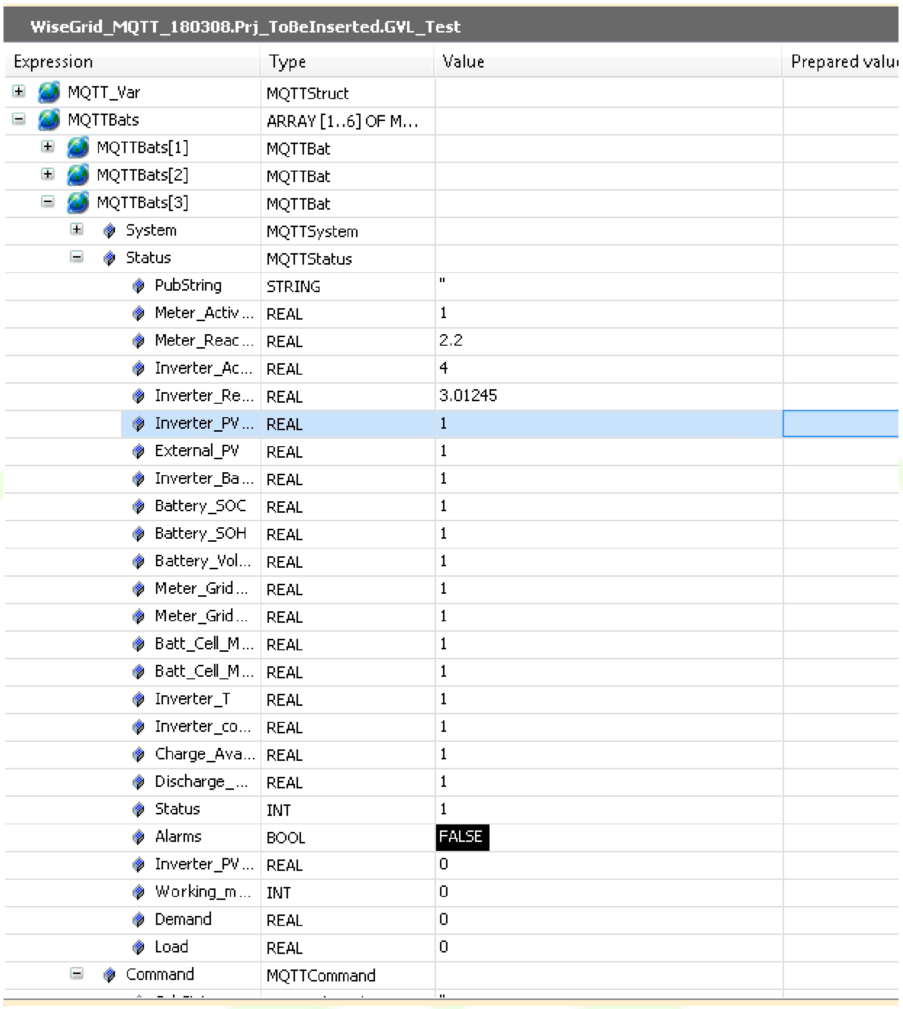
Below there is a screenshot of a subscribe example (battery#3 status): a computer subscribes to the topic "MQTT/battery/BAT0003/status" and it shows the battery#3 status:



```
^C
nono@nono-VirtualBox:~$ mosquitto_sub -h 172.31.41.10  -t "MQTT/battery/BAT0003/status"
{
        "Meter_Active_Power": 1.0,
        "Meter_Reactive_Power": 2.2,
        "Inverter_Active_Power": 4.0,
        "Inverter_Reactive_Power": 3.01245,
        "Inverter_PV_Power": 1.0,
        "External_PV": 1.0,
        "Inverter_Battery_Power": 1.0,
        "Battery_SOC: 1.0,
        "Battery_SOH": "1.0,
        "Battery_Voltage: 1.0,
        "Meter_Grid_Voltage: 1.0,
        "Meter_Grid_Frequency: 1.0,
        "Meter_Grid_Voltage: 1.0,
        "Temperatures": {
                "Batt_Cell_Max_T": 1.0,
                "Batt_Cell_Min_T": 1.0,
                "Inverter_T": 1.0
        }
        "Charge_Available": 1.0,
        "Discharge_Available": 1.0,
        "Status": 0,
        "Alarms": FALSE,
        "Inverter_PV_Voltage": 0.0,
        "Working_mode": 0,
        "Demand": 0.0
}
{
        "Meter_Active_Power": 1.0,
```

Figure 20 – SMX MQTT publishing

On the PLC, test variables that can be changed manually have been added, as can be seen below:



WiseGrid_MQTT_180308.Prj_ToBeInserted.GVL_Test

Expression	Type	Value	Prepared valu
MQTT_Var	MQTTStruct		
MQTTBats	ARRAY [1..6] OF M...		
MQTTBats[1]	MQTTBat		
MQTTBats[2]	MQTTBat		
MQTTBats[3]	MQTTBat		
System	MQTTSystem		
Status	MQTTStatus		
PubString	STRING	''	
Meter_Activ...	REAL	1	
Meter_Reac...	REAL	2.2	
Inverter_Ac...	REAL	4	
Inverter_Re...	REAL	3.01245	
Inverter_PV...	REAL	1	
External_PV	REAL	1	
Inverter_Ba...	REAL	1	
Battery_SOC	REAL	1	
Battery_SOH	REAL	1	
Battery_Vol...	REAL	1	
Meter_Grid...	REAL	1	
Meter_Grid...	REAL	1	
Batt_Cell_M...	REAL	1	
Batt_Cell_M...	REAL	1	
Inverter_T	REAL	1	
Inverter_co...	REAL	1	
Charge_Ava...	REAL	1	
Discharge_...	REAL	1	
Status	INT	1	
Alarms	BOOL	FALSE	
Inverter_PV...	REAL	0	
Working_m...	INT	0	
Demand	REAL	0	
Load	REAL	0	
Command	MQTTCommand		

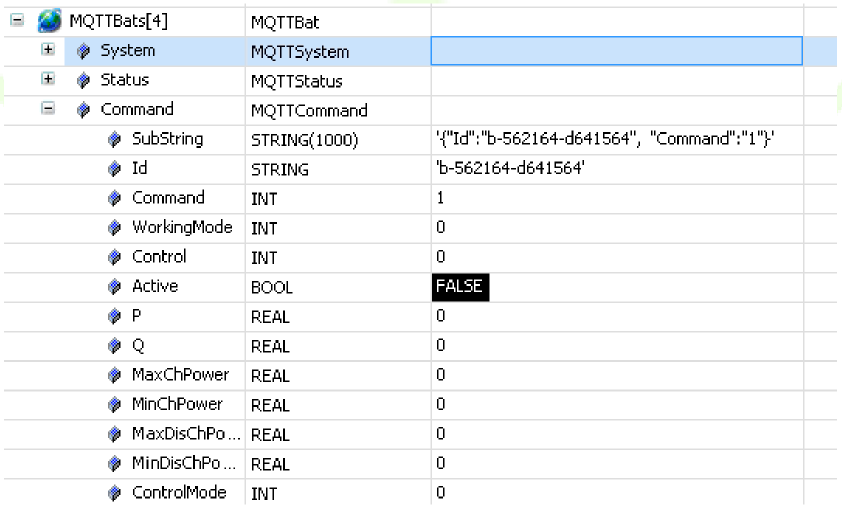
Figure 21 – variables on PLC

3.9.2.5 PLC subscribing, Computer publishing (on the SMX) test

On the computer, the following command line is written:

```
mosquitto_pub -h 172.31.41.10 -t "MQTT/battery/BAT0004/command" -m "{\"Id\":\"b-562164-d641564\", \"Command\":\"1\"}"
```

In the PLC, the reading is:



MQTTBats[4]	MQTTBat	
System	MQTTSystem	
Status	MQTTStatus	
Command	MQTTCommand	
SubString	STRING(1000)	'{"Id":"b-562164-d641564", "Command":"1"}'
Id	STRING	'b-562164-d641564'
Command	INT	1
WorkingMode	INT	0
Control	INT	0
Active	BOOL	FALSE
P	REAL	0
Q	REAL	0
MaxChPower	REAL	0
MinChPower	REAL	0
MaxDisChPo...	REAL	0
MinDisChPo...	REAL	0
ControlMode	INT	0

Figure 22 – Readings from PLC

This demonstrates that an MQTT message can be received on the PLC and decode it thanks to JSON PLC library.

3.10 COMMUNICATION WITH ENERGY STORAGE MODULE SYSTEM 2

3.10.1 Functional objective

VARTA Storage battery software must be extended to enable communication with an external MQTT broker and to send data and receive commands from the MQTT broker. Therefore, a “VS MQTT Client” must be implemented as part of VS’s energy management system (EMS) included in all VS battery systems.

As is the case for all other energy storage modules (AMP, BYES) to be connected to the WG Staas/VPP, the VS MQTT Client will make use of the common data model (see Section 3.1) and MQTT dialect (see Section 3.7).

3.10.2 Description and development

3.10.2.1 VARTA Storage’s energy management system

The technology stack of VS’s energy management system (EMS) is shown in Table 10 to give a brief overview.

Table 10 – VS' EMS technology stack

Operation system	GNU/Linux Debian 9 Stretch
Init system	systemd
CPU architecture	armhf
Compiler	GNU C++ 6.3
Programming languages	C++14, Python 3
Libraries	Qt 5.9 LTS

VS’s EMS internal structure based on this technology stack is sketched in Figure 23. All components are individual applications or daemons managed as services of the “systemd init” system. Due to this highly modular structure, many components of the Linux based system did not need to be changed. In particular, this accounts for the core EMS application and the long-term database.

A short-term database already exists, but it had to be reworked substantially to enable a remote controller, i.e., WG Staas/VPP to set target values and schedules from outside the core system. WiseGRID introduces some new control schemes for batteries – in particular the Scheduler mode, as described in Section 3.3 –, which previously were not considered for VARTA’s battery systems and which therefore had to be implemented for WiseGRID.

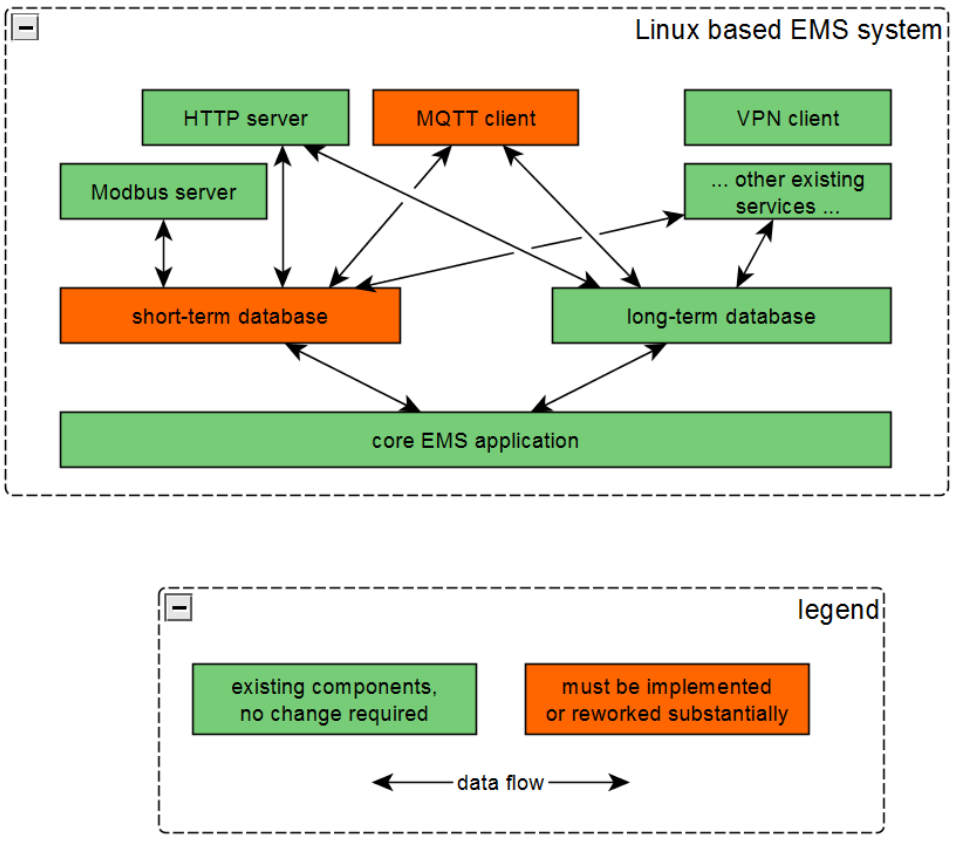


Figure 23 – sketch of VS' EMS internal structure[5]

3.10.2.2 Implementation of the MQTT client

As all other components, the MQTT client is an independent application running on top of the Linux operation system and management by the "systemd init" system. It is implemented as a C++ application using the Paho MQTT library for C++ [5].

Messages from the battery system to the WG StaaS/VPP (MQTT topic "status") are composed based on the values currently present in the short- and long-term databases. Messages to the battery system (MQTT topics "configuration" and "command") are written into JSON files, which are then interpreted by the core EMS application. The latter already has a possibility to read some commands from JSON files, but these files are not directly compatible to the JSON data model used within WiseGRID. Therefore, the MQTT client also has to parse the JSON from incoming MQTT messages and rewrite those messages to make them compatible to the inputs expected by the core EMS application.

3.10.2.3 Communication tests

As of the time of this writing, communications tests between VARTA's new MQTT client and the WG IOP and WG Staas/VPP are still ongoing and the results cannot be reported yet.

[5] Existing components are in green and components that need to be implemented for WiseGRID or substantially reworked are in red

3.11 COMMUNICATION WITH ENERGY STORAGE MODULE SYSTEM 3

3.11.1 Functional objective

Ampere ESS software must be adapted in order to allow an external controller to interact and determine the way the system behaves. The first step on this adaptation will be to include the communication interfaces required for publishing the internal variables and receiving external control and configuration commands.

The MQTT interface implemented will communicate directly with the WG IOP, as described in D4.2 [3].

3.11.2 Description and development

3.11.2.1 Ampere ESS

Ampere systems are based on a central controller which communicates with all the components of the system: batteries, inverter, and energy meter. This controller also provides the internet connection through Ethernet or WIFI for the system.

The EMS software runs on this controller and is responsible of the control and energy management functions of the ESS. Moreover, it also implements the necessary functions for establishing the external communications with the net and writing and reading messages through these communication channels. The MQTT interface is added to the available communication channels.

3.11.2.2 MQTT implementation

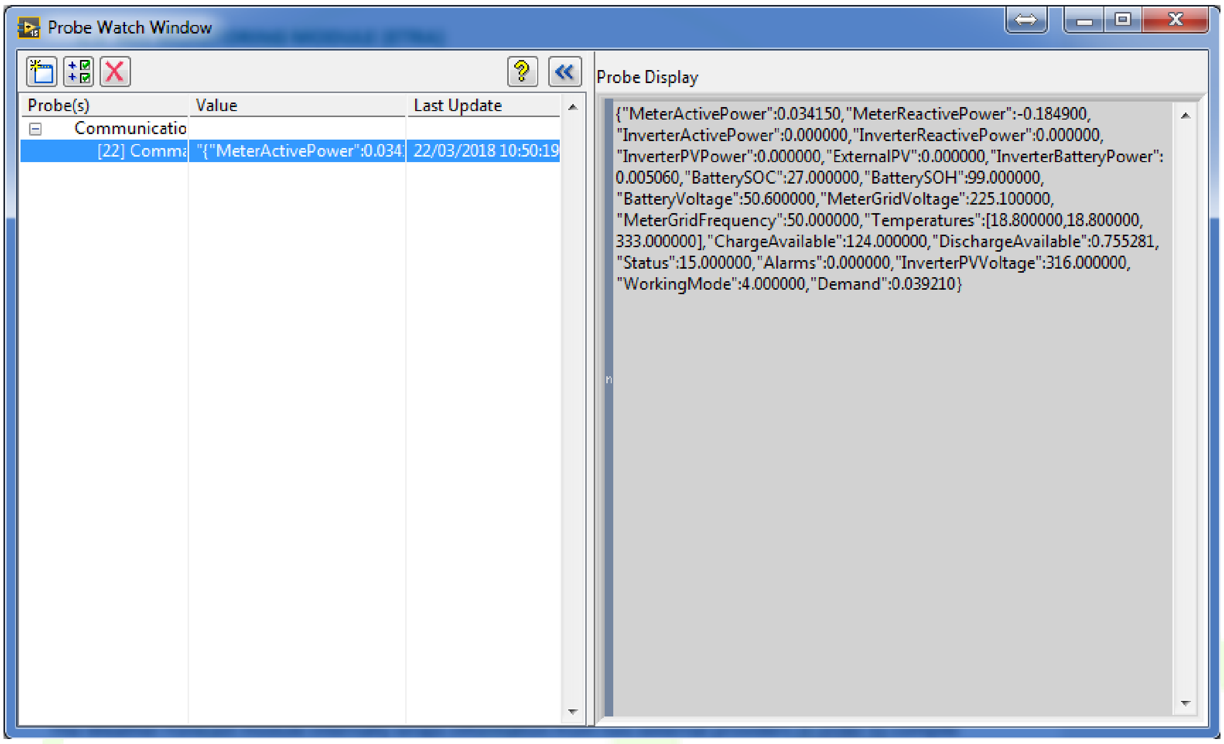
Ampere EMS is programmed with LabVIEW, which allows parallel and real-time computing. All the communication channels are implemented on a specific process that keeps reading the TCP ports and sending any data packet from the system. Any message received will be passed on to the corresponding sub-process that will perform the required actions. The messages that the different subprocesses generate to be sent to the internet are written in an internal queue that the communication channels process is constantly checking.

Inside this architecture, the MQTT interface is implemented as a new communication channel. The LabVIEW MQTT library includes the main functions needed to create an MQTT client, subscribe to and read from the corresponding topics and publish any kind of data. Messages are parsed to JSON format before publishing them.

The publish messages are issued in a different process, responsible of collecting the internal variables and processing them to conform the JSON data to be published. The publishing frequency can be configured in order to meet the requirements of any specific application.

3.11.2.3 Communication tests

Communications have been tested using MQTTfx client application on a computer, and LabVIEW probes on the EMS. Figure 24 shows the status message published by the EMS that is received by any device subscribed to the corresponding topic, as seen on Figure 25. Messages published on a topic at which the EMS is subscribed (Figure 26) are correctly received at the EMS (Figure 27).

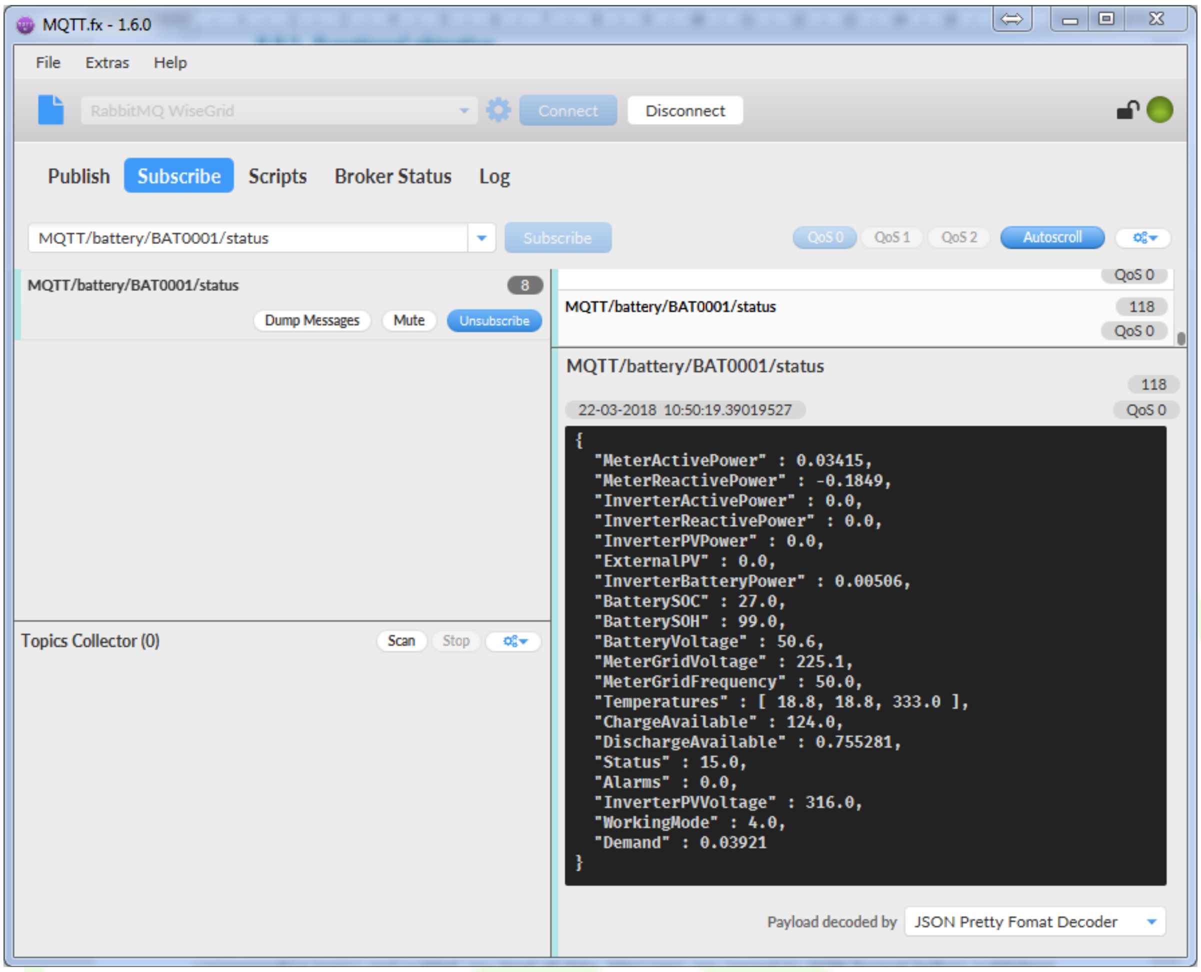

Probe Watch Window

Probe(s)	Value	Last Update
Communicatio		
[22] Comma	"{"MeterActivePower":0.034	22/03/2018 10:50:19

Probe Display

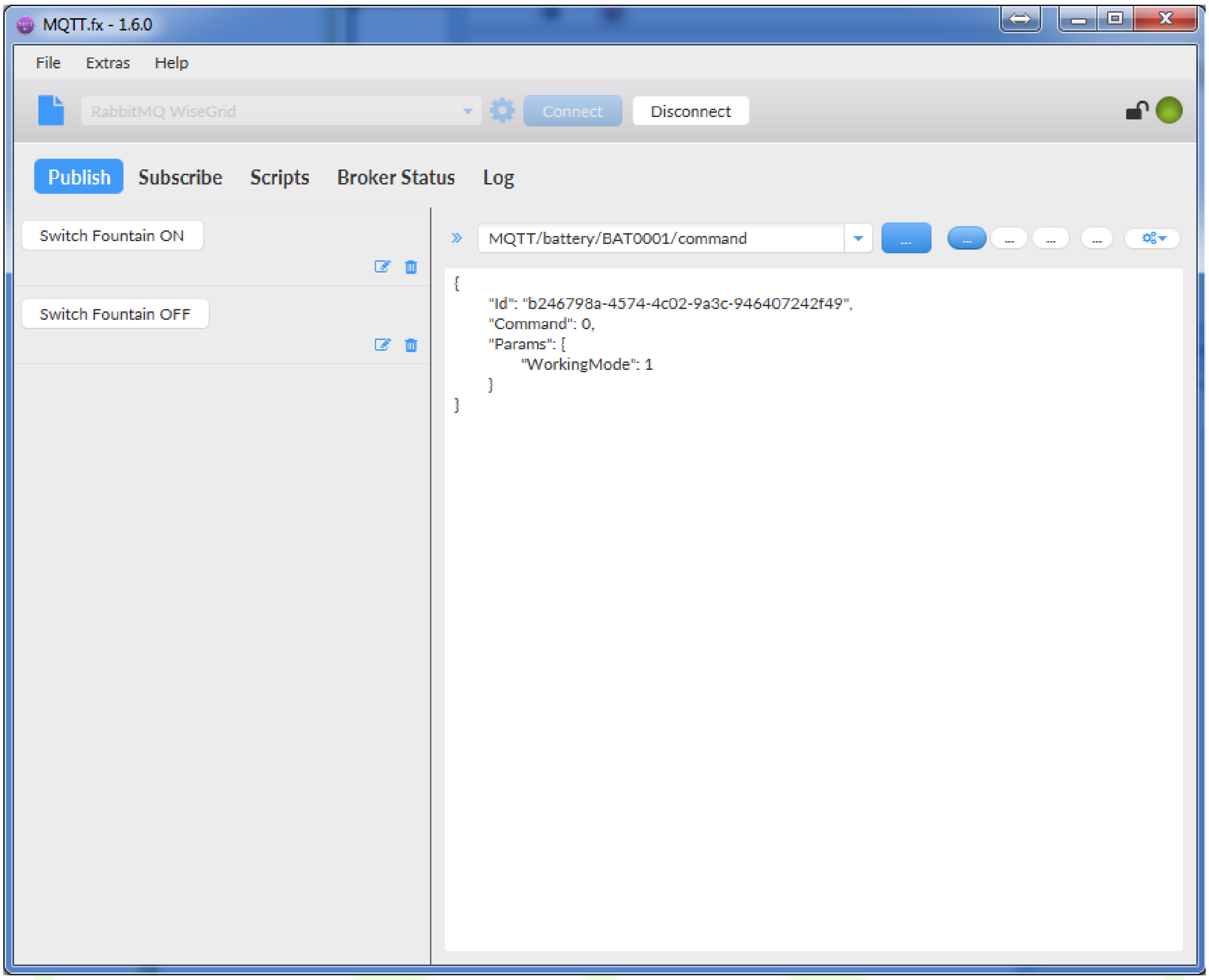
```
{"MeterActivePower":0.034150,"MeterReactivePower":-0.184900,
"InverterActivePower":0.000000,"InverterReactivePower":0.000000,
"InverterPVPower":0.000000,"ExternalPV":0.000000,"InverterBatteryPower":
0.005060,"BatterySOC":27.000000,"BatterySOH":99.000000,
"BatteryVoltage":50.600000,"MeterGridVoltage":225.100000,
"MeterGridFrequency":50.000000,"Temperatures":[18.800000,18.800000,
333.000000],"ChargeAvailable":124.000000,"DischargeAvailable":0.755281,
"Status":15.000000,"Alarms":0.000000,"InverterPVVoltage":316.000000,
"WorkingMode":4.000000,"Demand":0.039210}
```

Figure 24 – Message published by the EMS



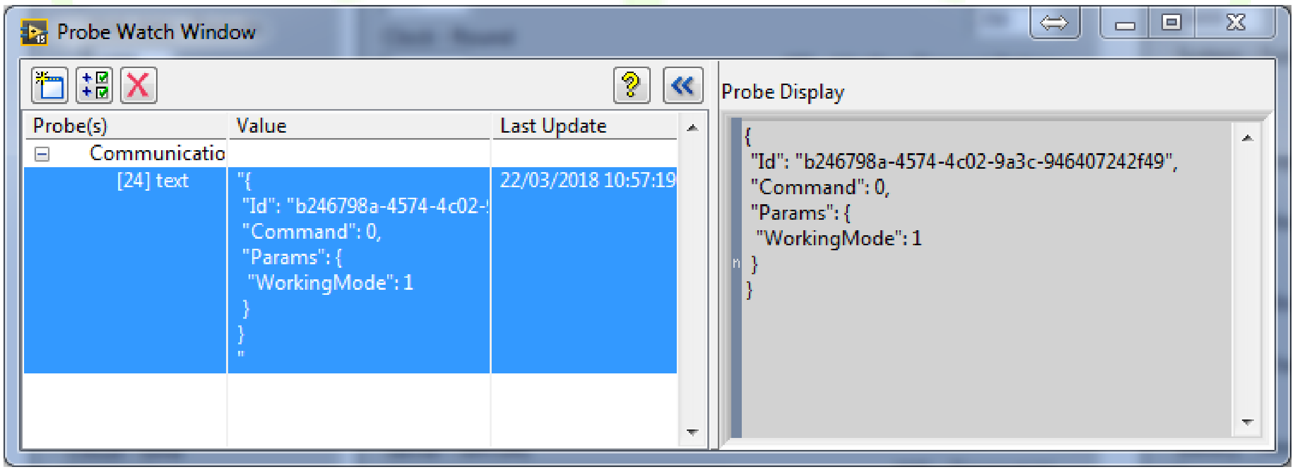
```
{
  "MeterActivePower" : 0.03415,
  "MeterReactivePower" : -0.1849,
  "InverterActivePower" : 0.0,
  "InverterReactivePower" : 0.0,
  "InverterPVPower" : 0.0,
  "ExternalPV" : 0.0,
  "InverterBatteryPower" : 0.00506,
  "BatterySOC" : 27.0,
  "BatterySOH" : 99.0,
  "BatteryVoltage" : 50.6,
  "MeterGridVoltage" : 225.1,
  "MeterGridFrequency" : 50.0,
  "Temperatures" : [ 18.8, 18.8, 333.0 ],
  "ChargeAvailable" : 124.0,
  "DischargeAvailable" : 0.755281,
  "Status" : 15.0,
  "Alarms" : 0.0,
  "InverterPVVoltage" : 316.0,
  "WorkingMode" : 4.0,
  "Demand" : 0.03921
}
```

Figure 25 – Message received by the computer



```
{
    "Id": "b246798a-4574-4c02-9a3c-946407242f49",
    "Command": 0,
    "Params": {
        "WorkingMode": 1
    }
}
```

Figure 26 – Message published by the computer



```
{
 "Id": "b246798a-4574-4c02-9a3c-946407242f49",
 "Command": 0,
 "Params": {
  "WorkingMode": 1
 }
}
```

Figure 27 – Message received by the EMS

3.12 RES MONITORING MODULE

3.12.1 Functional objective

The objective of this module is to publish real-time measurements of production of RES under the control of the WG StaaS/VPP application, to be further processed by the corresponding modules of the WG StaaS/VPP application.

3.12.2 Description and development

In order to retrieve production data, production units to be integrated need to be monitored using smart meters. SMX devices are a result of the former project NobelGRID [6] that can interface with a large set of smart meters in order to provide them with a new set of features, one of the most important being the capability to provide real-time data with short update periods using IoT communication mechanisms (e.g. MQTT protocol over Internet). This is the core idea behind the Unbundled Smart Meter concept.

SMX devices will therefore be used to implement the wrapper in order to, on one side, interface with the corresponding smart meters in charge of read production data and, on the other side, publish information to the WiseGRID IOP, which will deliver it appropriately to the WG StaaS/VPP application.

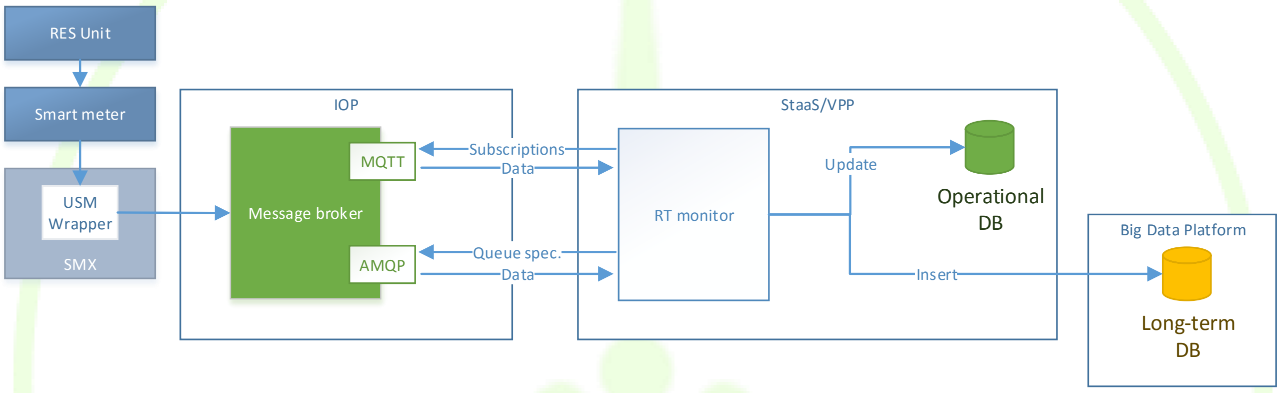


Figure 28 – Overview of the monitoring process of RES Units

3.12.3 Interfaces

Production data, as well as all energy demand and production related data provided by the USM Wrappers, is provided in the form of CIM MeterReading messages and published to the WiseGRID IOP using MQTT protocol. Being one of the main Common Data Models used in the project, the structure of those messages is described in detail in D4.2 [3]. The *Real-time Monitor* module within the WG StaaS/VPP application is configured to subscribe to these data flows and store the production data provided by the RES Units under control of the WG StaaS/VPP.

Within both databases, this data is stored in a collection named *devices_smartmeters*. While the operational database keeps just one document per smart meter, with the last known status of it, the long-term DB holds the whole history of measurements provided by each device.

```
{
    "_id" : "BBB5979",
    "Meter" : {
        "mRID" : "BBB5979",
        "reason" : "",
        "remark" : "",
        "value" : "ok"
```

```
},
"topic" : "BBB5979/SMX/CIM",
"Readings" : [
    {
        "ReadingType" : "0.0.0.0.0.1.4.0.0.0.0.0.0.0.128.0.5.0",
        "timeStamp" : "2018-04-04T06:32:18.000Z",
        "value" : 0.436,
        "ReadingQualities" : {
            "comment" : "Instantaneous current L1"
        }
    },
    {
        "ReadingType" : "0.0.0.0.0.1.54.0.0.0.0.0.0.0.128.0.29.0",
        "timeStamp" : "2018-04-04T06:32:18.000Z",
        "value" : 223.74,
        "ReadingQualities" : {
            "comment" : "Instantaneous voltage L1"
        }
    },
    {
        "ReadingType" : "0.0.0.12.0.1.36.0.0.0.0.0.0.0.128.0.153.0",
        "timeStamp" : "2018-04-04T06:32:19.000Z",
        "value" : 0.835,
        "ReadingQualities" : {
            "comment" : "L1 Instantaneous Power Factor"
        }
    },
    {
        "ReadingType" : "0.26.0.1.1.1.12.0.0.0.0.0.0.0.224.3.72.0",
        "timeStamp" : "2018-04-04T06:32:18.000Z",
        "value" : 15.54,
        "ReadingQualities" : {
            "comment" : "Active Energy + Total"
        }
    },
    {
        "ReadingType" : "0.0.0.12.0.1.15.0.0.0.0.0.0.0.224.0.33.0",
        "timeStamp" : "2018-04-04T06:32:19.000Z",
            "value" : 49.97,
            "ReadingQualities" : {
                "comment" : "Instantaneous net Frequency any phase"
            }
        },
        {
            "ReadingType" : "0.26.0.1.19.1.12.0.0.0.0.0.0.0.224.3.72.0",
            "timeStamp" : "2018-04-04T06:32:18.000Z",
            "value" : 0,
            "ReadingQualities" : {
                "comment" : "Active Energy - Total"
            }
        },
        {
            "ReadingType" : "0.0.0.12.1.1.37.0.0.0.0.0.0.0.128.0.38.0",
            "timeStamp" : "2018-04-04T06:32:18.000Z",
            "value" : 0.081,
            "ReadingQualities" : {
                "comment" : "L1 Instantaneous Active Power"
            }
        }
    ]
}
```

3.13 WEATHER FORECAST MODULE

3.13.1 Functional objective

The Weather Forecast provider is a horizontal module of the project with the purpose of providing both actual and forecast of weather-related information to any application requiring it.

3.13.2 Description and development

The Weather Forecast module provides the information required by the Demand and Production forecast modules in order to crosscheck the demand and forecast historic information with weather-related parameters, thus allowing those modules to learn the relationship between these elements in order to provide more accurate demand and production forecasts. The Weather Forecast module needs therefore to fulfil the requirements of the demand and production forecast modules.

The Weather Forecast module internally wraps information from two external providers in order to compile all required information:

- OpenWeatherMap [7] is used to access generic weather information from any location in the world, as this information is relevant to the demand and production forecast algorithms

- SOLCAST [8] is used to access specific solar irradiance information from any location in the world, as this information is required by the solar production forecast algorithm

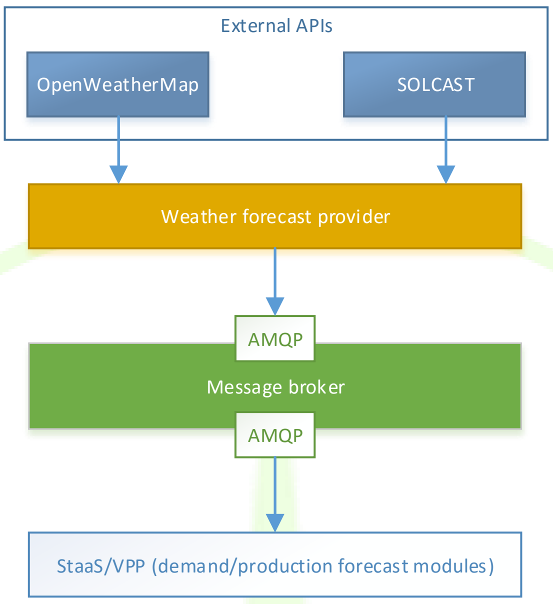

External APIs
OpenWeatherMap
SOLCAST
Weather forecast provider
AMQP
Message broker
AMQP
StaaS/VPP (demand/production forecast modules)

Figure 29 – Integration of Weather Information sources within the Weather Forecast Provider

A summary of the information provided by the Weather Forecast is included in the following tables.

Table 11 – Weather Forecast service. Current weather information

Current weather information	
Temperature	Temperature in ºK
Pressure	Bars
Humidity	%
Wind speed	Km/h
Wind direction	Degrees
Cloud coverage	%
Direct normal irradiance	W/m2
Zenith	Solar zenith angle (degrees). Zero means directly upwards/overhead). Varies from 0 to 180. A value of 90 means the sun is at the horizon.
Azimuth	Solar azimuth angle (degrees). Zero means true north. Varies from -180 to 180. Positive is anticlockwise (west). A value of -90 means the sun is in the east.

Table 12 – Weather forecast service. Forecast weather information

Forecasted weather information

Temperature	Temperature in ºK
Pressure	Bars
Humidity	%
Wind speed	Km/h
Wind direction	Degrees
Cloud coverage	%
Direct normal irradiance	W/m2
Air temperature	Temperature in ºC
Zenith	Solar zenith angle (degrees). Zero means directly upwards/overhead). Varies from 0 to 180. A value of 90 means the sun is at the horizon.
Azimuth	Solar azimuth angle (degrees). Zero means true north. Varies from -180 to 180. Positive is anticlockwise (west). A value of -90 means the sun is in the east.

3.13.3 Interfaces

The Weather Forecast module is designed to be integrated within the IOP message broker platform [3] or within the internal Enterprise Service Bus of any of the applications of the project. In order to fulfil those integration requirements, the possible queries to the Weather Forecast Module have been designed using the RPC pattern with AMQP protocol. This interface is formally described in the next paragraphs.

Weather forecast

Provides weather forecast in the given location for the next 10 days, with a granularity of 3 hours.

AMQP queue: "weatherforecastprovider"

Message properties

- reply_to: name of the queue where response will be delivered
- correlation_id: free text for query/response correlation (RPC pattern https://www.rabbitmq.com/tutorials/tutorial-six-python.html)

Message payload examples

```
{
      "query":"forecast",
      "city":"Valencia,ESP"
}
```

Or

```
{
      "query":"forecast",
      "lat":39.46975,
      "lon":-0.37739
}
```

Or

```
{
        "query":"forecast",
        "cityId":"6362115"
}
```

City codes available here: http://bulk.openweathermap.org/sample/city.list.json.gz

Response example:

```
[
{
        "timestamp" : "2018-01-01T00:00:00.000Z",
        "temp" : 279.47,
        "pressure" : 977.52,
        "humidity" : 100,
        "clouds" : 92,
        "wind" : {
                "speed" : 2.67,
                "deg" : 16.506
        },
        "dhi" : 283,
        "azimuth" : 72,
        "zenith" : 37
},
…
]
```

Current weather

Provides information about the current weather in the given location

AMQP queue: "weatherforecastprovider"

Message properties

- reply_to: name of the queue where response will be delivered
- correlation_id: free text for query/response correlation (RPC pattern https://www.rabbitmq.com/tutorials/tutorial-six-python.html)

Message payload examples

```
{
        "query":"weather",
        "city":"Valencia,ESP"
}
```

Or

```
{
        "query":"weather",
        "lat":39.46975,
        "lon":-0.37739
}
```

Or

```
{
        "query":"weather",
        "cityId":"6362115"
}
```

City codes available here: *http://bulk.openweathermap.org/sample/city.list.json.gz*

Response example:

```
{
        "temp" : 280.15,
        "pressure" : 1007,
        "humidity" : 87,
        "clouds" : 75,
        "wind" : {
                "speed" : 5.1,
                "deg" : 350
        },
        "dhi" : 283,
        "azimuth" : 72,
        "zenith" : 37
}
```

3.14 PRODUCTION FORECAST SERVICE

3.14.1 Functional objective

The main aim of this service is to provide a production forecast that will depend on the requirements of the client, thus different clients (or either the same client) could ask to this service asking for different forecasting scopes. In addition, this forecast service must be capable of doing a production forecast taking in consideration a mix of technologies: photovoltaic and wind power generation.

Summarizing, the production forecast will contribute to WG StaaS/VPP tool:

- Improving the energy market transactions due to a more flexible module that allows obtaining different production forecasting profiles results depending on the scope (or area) selected by the WG StaaS/VPP development.
- Improving the quality and the quantity of the services that the the WG StaaS/VPP offers to the end-users due to more detailed production forecasting (more personalized services and more favourable economic transactions).
- Improved planning for storage use. In fact, the use of production forecasting profiles with different scope areas at the same time will allow sharing storage resources among different infrastructures placed relatively closed.

In addition, the production forecast service will provide to the WG StaaS/VPP forecasting profiles for its areas of interest, that do not have to be coincident with geographical areas. Thus, the WG StaaS/VPP will be able to perform a better management of its assets and resources using this forecasting division.

3.14.2 Description and development

The production forecasting algorithm has been implemented under CNEA (Cascaded Neuro-Evolutionary Algorithm) forecasting model premises that are based on neural networks algorithms. The CNEA forecasting model consists on some SVM (Support Vector Machine) in cascade with an optimal selection of input models, number of neurons and evolutive training process (see Figure 30). The iterative CNEA algorithm uses a SVM to forecast the power of the first hour (h+1); this result is used as an input of the next SVM to forecast the power for the next hour (h+2) and the algorithm repeats this process until the last forecasting slot is achieved. Thus, it is required 24 SVM executed in cascade for doing a forecasting of 24 values [9].

The proper way to obtain a good forecasting result in a short term power forecasting is to use past days with similar power profiles. Therefore, if the historical information available in the long term data base is not enough the error of the forecasting result will increase. To reduce this error, in the first step of the CNEA model, the most influential variables are chosen from the data base in an optimal way.

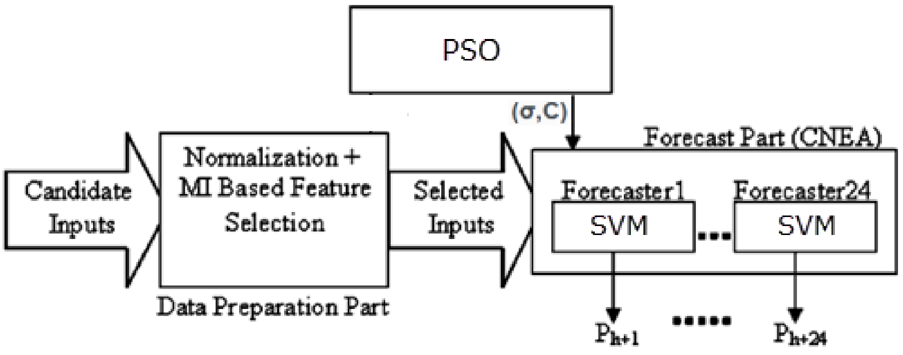


Figure 30 – CNEA network structure [9]

The algorithm has also a high number of internal parameters that will be automatically optimized to obtain the best prediction, such as: number of past days for training, number of past hours for prediction and variables related with internal supervised learning and optimization algorithms.

Furthermore in the Long Term database should be available information about historical data and calendars described below. In addition, it is mandatory to have the following information in the long term data base:

- The aggregation of CUPS (concerning to the productors) associated to the same aggregation identifier.
- The energy resource type of each CUPS; the possible renewable resource considered in the scope of the project are photovoltaic and wind power. Thus, a production forecast request can consider a mix

of different renewable resources.

Finally, production forecast service request weather forecast to the service developed in the scope of the project. Specifically, the detailed information required from the weather forecast to the production forecast service is detailed below.

Historical data will correspond with the historical data of the renewable generation profile for each supply point. The information that will be included in this input will be:

- Time stamp (type: timestamp). Recorded day and time concerning the historical information.
- Value (type: int16). Recorded power value at the specific timestamp in watts [W].

The calendar input will be a two dimensional matrix with the detail of working days, holidays and weekends for a whole year. Therefore, as many calendars as regions will be necessary.

Information contained in this matrix will be:

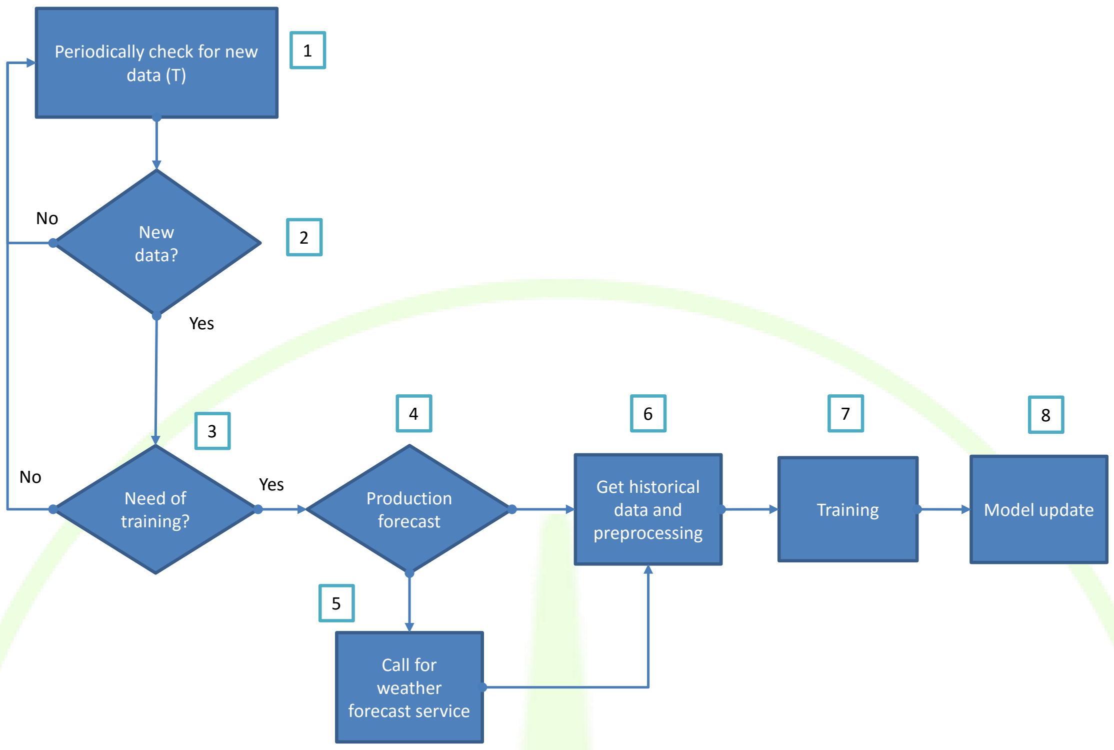
- Day (type: day [dd-mm-yyyy]). Day of the year
- Type of day (type: int8). It could have the values (1) working days; (2) holyday

The forecasts will need, as an input, information concerning the maximum, minimum and average temperature of a day for the next days to be included in the forecasting window. Thus, the information contained in this matrix will be:

- Day and hour (type: day [dd-mm-yyyy hh:mm]). Day of the year and hour.
- Wind speed (type: int8). Meters per second.
- Wind direction: SSO, NNO, etc.
- Average temperature forecasted in Celsius degrees (type: int8)
- Maximum temperature forecasted in Celsius degrees (type: int8)
- Minimum temperature forecasted in Celsius degrees (type: int8)
- Solar radiation (type: int8). W/m^2
- zenith angle
- azimuth angle

The characteristics of the historical data will affect the forecast output requirements. As an example, the forecasting algorithm will not be able to obtain a forecast with fifteen minutes detail if the historical data does not have at least the same time granularity.

The forecasting algorithms will have two different workflows: one for training the models and the other one to attend the forecasting requests.
The workflow for training the forecasting models runs once a day (at the end of the day) and follows these steps:

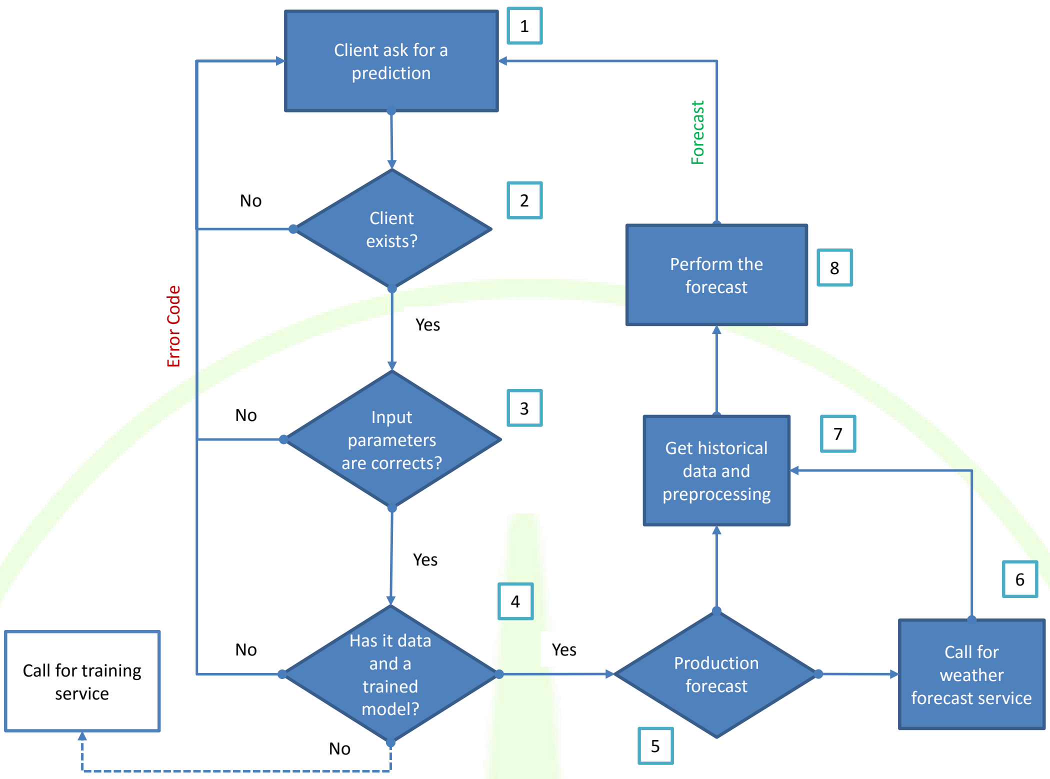


Periodically check for new data (T)
1
New data?
2
No
Yes
3
Need of training?
No
Yes
4
Production forecast
5
Call for weather forecast service
6
Get historical data and preprocessing
7
Training
8
Model update

Figure 31 – Production forecast - Training workflow schema

1. This service works periodically, checking for new historical data for each client.
2. If new data is available on Long Term DB next step will be reached, otherwise return to step 1.
3. Not always is needed to train forecasting models. Depending on the restrictions could not be necessary to re-train the model.
4. In the request concerning production forecast, it will be needed a weather forecast request thus, the client location should be able into the database to perform these calls.
5. In production model to be trained, a call for the weather historical and forecast data will be needed and information collected from this service will depend on the energy resource (wind or solar energy).
6. Historical production and weather data will be obtained through a Long Term DB call.
7. With all collected data the training of the model is performed.
8. Finally, old models are deleted and the new one remains to perform new forecasts (see forecast request workflow).

The workflow to attend the forecast requests follows these steps:



Client ask for a prediction
1
Forecast
No
Client exists?
2
Perform the forecast
8
Error Code
Yes
No
Input parameters are corrects?
3
Get historical data and preprocessing
7
Yes
6
4
Call for training service
No
Has it data and a trained model?
Yes
Production forecast
Call for weather forecast service
No
5

Figure 32 – Production Forecast - Forecast workflow schema

1. A client asks for a forecast by invoking a RPC function and adding some input parameters.
2. The identifier of that client is checked to know if exists.
3. The input parameters are checked to validate coherence.
4. It is checked if a model is trained and historical data is available.
5. In the request concerning production forecast, it will be needed a weather forecast request thus, the client location should be available in the database to perform these calls. In particular, the required information will depend on the energy resource (solar or wind energy).
6. Historical production data will be obtained through a Long Term DB call.
7. Forecast is performed and returned to the client.

3.14.3 Interfaces

Provides load demand forecast for a client/tenant who ask for a prediction.

AMQP queue: “forecasting_production”

Message properties

- reply_to: name of the queue where response will be delivered
- correlation_id: free text for query/response correlation (RPC pattern https://www.rabbitmq.com/tutorials/tutorial-six-python.html)

- Payload[6]:
    - client_id: client identifier in the WiseGRID database.
    - Horizon: number of days client wants to predict, starting from current day. From 1 to 7.
    - Period: Time period between forecast samples. 15 min / 60 min. Default 60 minutes.

Response properties

- errCode: Error code regarding possible exceptions.
- Forecast: Desired prediction formed by key value pair ("Timestamp" : Value)
    - Timestamp: UNIX time seconds (UTC).
    - Value: predicted value for the specified timestamp.
- Units: Value units.

Table 13 – Production forecast API, error codes

Error Code	Meaning
0	No error
1	Invalid client identifier
2	No enough historical data to perform forecasts
3	Untrained forecast model
4	Unfound forecast model
5	Invalid parameter horizon
6	Invalid parameter period
7	Unauthorized access to forecast model
255	Unhandled error

Message payload examples

```
{     "Client_id": 1,
      "Horizon": 1,
      "Period": 60,


}
```

Response example:

```
{
      "errCode":0,
      "forecast":
```

[6] Note that for a production forecast, a weather forecast for a specific location is needed. Clients do not need to specify its location since this information should be available into the database.

```
        {
              "1507154400":22.544,
              "1507158000":21.438,
              "1507161600":12.242,
              "1507165200":12.116,
              "1507168800":10.985,
              "1507172400":12.235,
              "1507176000":9.152,
              "1507179600":58.837,
              "1507183200":65.365,
              "1507186800":22.05,
              "1507190400":38.03,
              "1507194000":8.861,
              "1507197600":1.071,
              "1507201200":15.919,
              "1507204800":20.187,
              "1507208400":16.721,
              "1507212000":9.775,
              "1507215600":2.027,
              "1507219200":4.288,
              "1507222800":3.249,
              "1507226400":6.186,
              "1507230000":2.068,
              "1507233600":3.909,
              "1507237200":2.478
        },
        "units":"kW"
}
```

3.15 DEMAND FORECAST SERVICE

3.15.1 Functional objective

The main aim of this service is to provide a demand forecast that will depend on the requirements of the client, thus different clients (or either the same client) could ask to this service requesting for different forecasting scopes. As is the case with the generation forecast service, this service will contribute to WG StaaS/VPP tool in the following:

- Improving the energy market transactions due to a more flexible module that allows obtaining different demand forecasting profiles results depending on the scope (or area) selected by the WG StaaS/VPP development.

- Improving the quality and the quantity of the services that the the WG StaaS/VPP offers to the end-users due to more detailed demand forecast (more personalized services and more favourable economic transactions).
- Improved planning for storage use. In fact, the use of demand forecasting profiles with different scope areas at the same time will allow sharing storage resources among different infrastructures placed relatively closed.

Finally, the demand forecast service will provide to the WG StaaS/VPP forecasting profiles for its areas of interest, that do not have to be coincident with geographical areas. Thus, the WG StaaS/VPP will be able to perform a better management of its assets and resources using this forecasting division.

3.15.2 Description and development

The demand forecasting algorithm has been implemented under the same base design used in the production forecasting algorithm thus, the demand forecasting algorithm has been implemented under CNEA (Cascaded Neuro-Evolutionary Algorithm) forecasting model premises that are based on neural networks algorithms that were described in more detail in the previous Section.

In this case the input information required in the Long Term database should be available information about historical data and calendars described below. In addition, it is mandatory to have next information in the long term data base:

- The aggregation of CUPS (Supply Point Universal Code concerning consumers) associated to the same aggregation identifier.

Historical data will correspond with the historical data of the load profile for each supply point and the historical weather records. The information that will be included concerning the historical demand profile input will be:

- Time stamp (type: timestamp). Recorded day and time concerning the historical information.
- Value (type: int16). Recorded power value at the specific timestamp in watts [W].

The calendar input will be a two dimensional matrix with the detail of working days, holidays and weekends for a whole year. Therefore, as many calendars as regions will be necessary. Information contained in this matrix will be:

- Day (type: day [dd-mm-yyyy]). Day of the year
- Type of day (type: int8). It could have the values (1) working days; (2) holyday

Finally, demand forecast service requests weather forecast to the service developed in the scope of the project. Specifically, the detailed information required from the weather forecast to the demand forecast service is concerning the maximum, minimum and average temperature of a day for the next days to be included in the forecasting window. Thus, the information contained in this matrix will be:

- Day and hour (type: day [dd-mm-yyyy hh:mm]). Day of the year and hour.
- Average temperature forecasted in Celsius degrees (type: int8)
- Maximum temperature forecasted in Celsius degrees (type: int8)
- Minimum temperature forecasted in Celsius degrees (type: int8)

The characteristics of the historical data will affect the forecast output requirements. As an example, the forecasting algorithm will be not able to obtain a forecast with fifteen minutes detail if the historical data does not have at least the same time granularity.

In the same way that the production forecast service, the demand forecast algorithms will have two different workflows: one for training the models and the other one to attend the forecasting requests.

The workflow for training the forecasting models runs once a day (at the end of the day) and follows next steps:

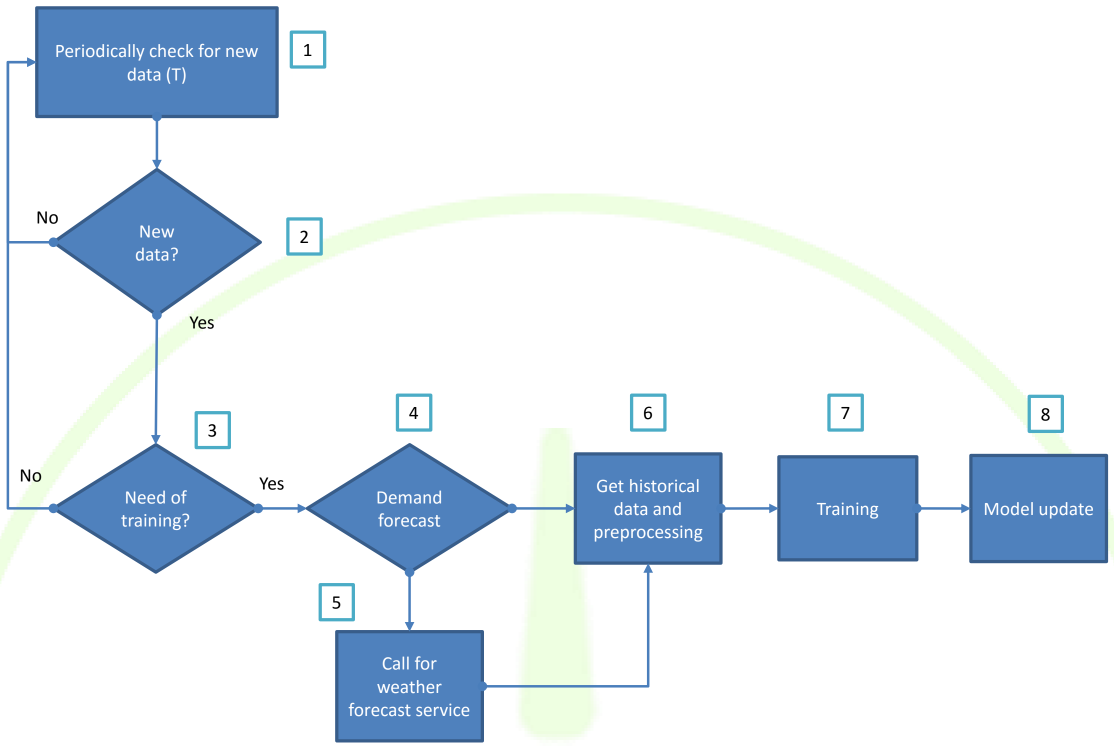

Periodically check for new data (T)
1
New data?
2
No
Yes
3
Need of training?
No
Yes
4
Demand forecast
5
Call for weather forecast service
6
Get historical data and preprocessing
7
Training
8
Model update

Figure 33 – Demand Forecast - Training workflow schema

1. This service works periodically, checking for new historical data for each client.
2. If new data is available on Long Term DB next step will be reached, otherwise return to step 1.
3. Not always is needed to train forecasting models. Depending on the restrictions could be not necessary re-train the model.
4. In the case of demand forecast, historical information and weather forecast is required in the training process.
5. For weather forecast request, the client location should be able into the database to perform these calls.
6. Historical demand data will be obtained through a Long Term DB call.
7. With all collected data the training of the model is performed.
8. Finally, old models are deleted and the new one remains to perform new forecasts.

The workflow to attend the demand forecast requests follows next steps:

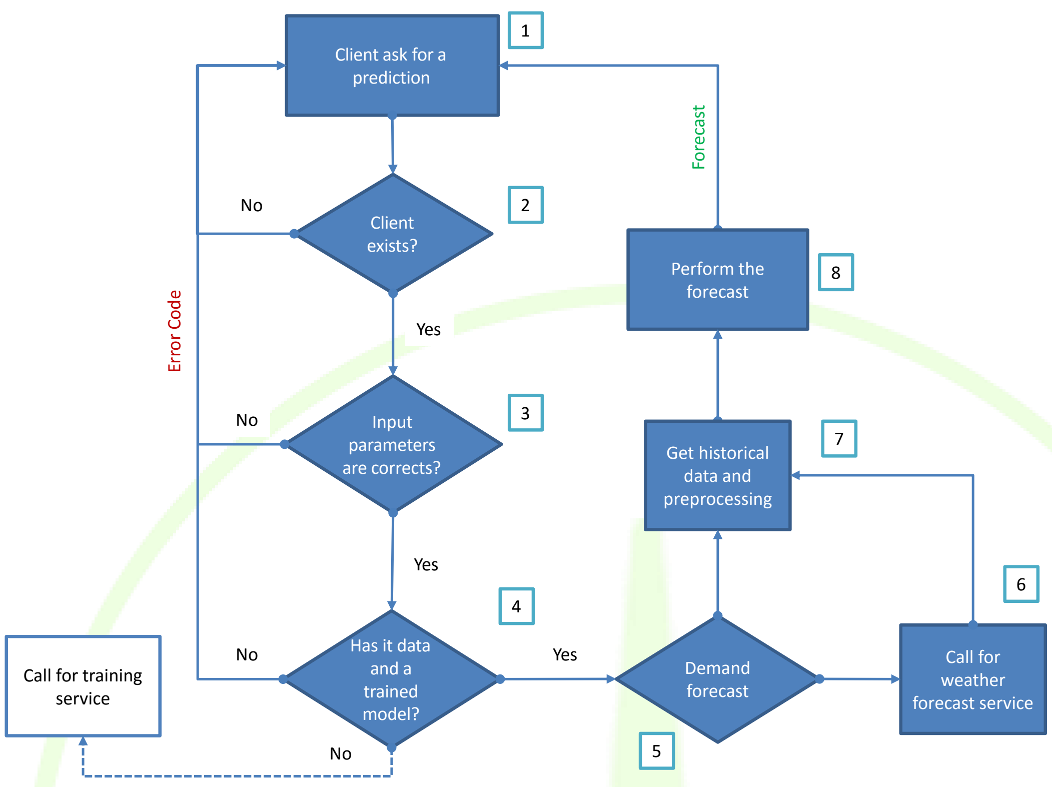

Client ask for a prediction
1
Client exists?
2
No
Yes
Input parameters are corrects?
3
No
Yes
Has it data and a trained model?
4
No
Yes
No
Call for training service
Error Code
Forecast
Perform the forecast
8
Get historical data and preprocessing
7
Demand forecast
5
Call for weather forecast service
6

Figure 34 – Demand Forecast - Forecast workflow schema

1. A client asks for a forecast by invoking a RPC function and adding some input parameters.
2. The identifier of that client is checked to know if exists.
3. The input parameters are checked to validate coherence.
4. It is checked if a model is trained and historical data is available.
5. Historical temperatures and load profile and weather forecast are required to perform demand forecast.
6. It will perform a weather forecast call per each area. That is why the client location should be able into the database to perform these calls.
7. Historical temperatures and load profile data will be obtained through a Long Term DB call.
8. Forecast is performed and returned to the client.

3.15.3 Interfaces

The demand and production forecast provider is a wrapper of the forecast algorithms developed by ITE for an easy integration with the WG StaaS/VPP tool.

The forecasting module developed in the WISEGRID project can be seen as a function that forecasts the next desired values taking into consideration historical data and additional data such as a calendar (with working days, holidays and weekends), exogenous variables (weather) and some client parameters.

Figure 35 contains the structure defined to integrate demand forecast and renewable forecast services into the WG StaaS/VPP developments. The forecast server will communicate with clients through the WG IOP, specifically, by means of the RabbitMQ services in the central control platform. Thus, the forecast server will include an AMQP client with two main objectives: route client request to the RPC forecasting functions and call for the RPC function which obtains the weather forecast.

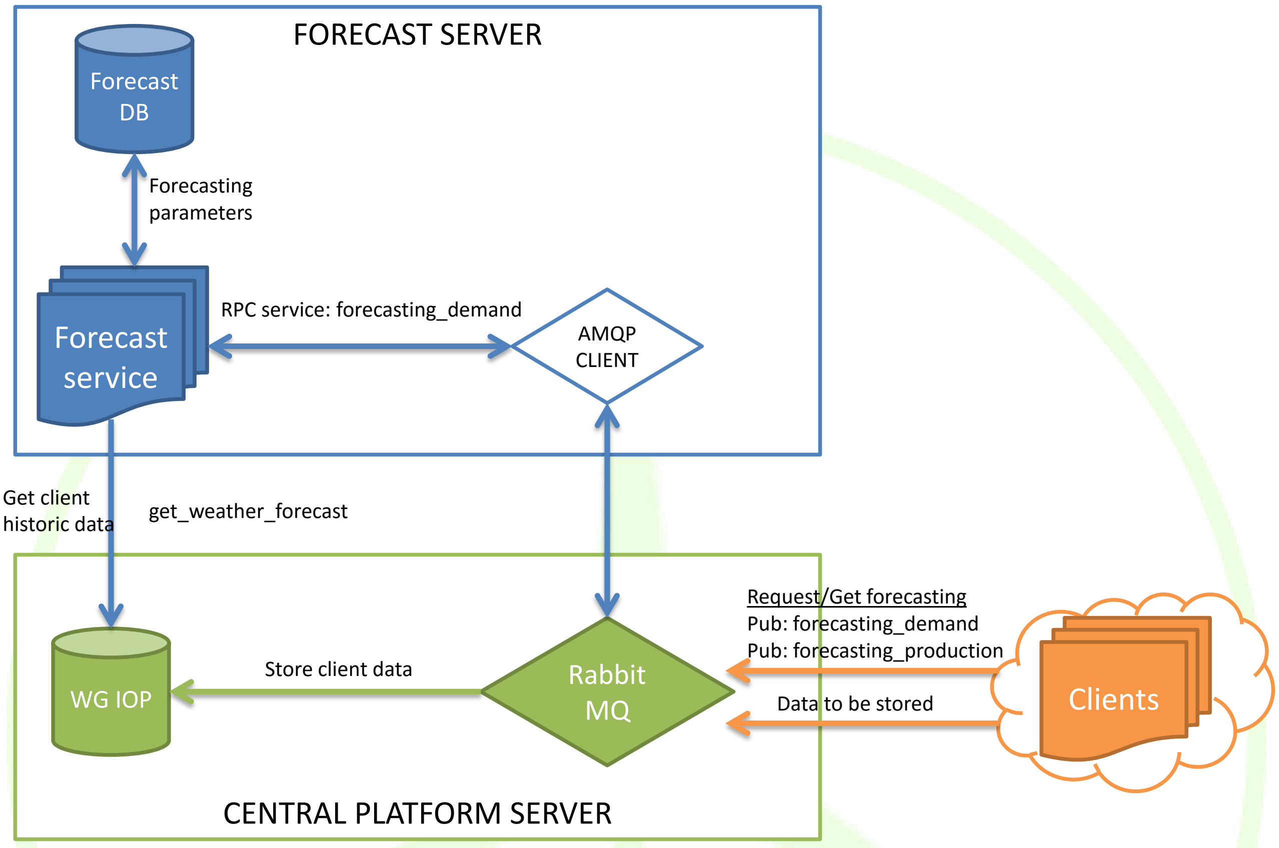


Figure 35 – Forecast server structure integration schema

Based on this architecture, the implemented developments will support the next general procedure. Note that are clients who ask on demand for a prediction.

1. The StaaS/VPP tool (called client or tenant) requests the forecasting profile calling a Remote Procedure Call function with some required parameters (more details in following sections).
2. The forecast service will receive the forecasting request, by means the AMQP client, and will perform the forecasting based on historical data (long term data base) and weather forecast if proceed.
3. Once the forecasting service has completed the forecast request, taking into consideration parameters included in the request message, the forecast service will return as a response the results to that specific client.

The forecast database is in charge of storing every model and parameter directly needed for the training of the algorithm model and for the forecast.

Provides load demand forecast for a client/tenant who ask for a prediction.

AMQP queue: “forecasting_demand”

Message properties

- reply_to: name of the queue where response will be delivered

- correlation_id: free text for query/response correlation (RPC pattern https://www.rabbitmq.com/tutorials/tutorial-six-python.html)
- Payload:
  - client_id: client identifier in the WiseGRID database.
  - Horizon: number of days client wants to predict, starting from current day. From 1 to 7.
  - Period: Time period between forecast samples. 15 min / 60 min. Default 60 minutes.

Response properties

- errCode: Error code regarding possible exceptions.
- Forecast: Desired prediction formed by key value pair ("Timestamp" : Value)
  - Timestamp: UNIX time seconds (UTC).
  - Value: predicted value for the specified timestamp.
- Units: Value units.

Table 14 – Demand forecast API, error codes

Error Code	Meaning
0	No error
1	Invalid client identifier
2	No enough historical data to perform forecasts
3	Untrained forecast model
4	Unfound forecast model
5	Invalid parameter horizon
6	Invalid parameter period
7	Unauthorized access to forecast model
255	Unhandled error

Message payload examples

```
{     "Client_id": 1,
      "Horizon": 1,
      "Period": 60
}
```

Response example:

```
{
      "errCode":0,
      "forecast":
      {
            "1507154400":22.544,
          "1507158000":21.438,
          "1507161600":12.242,
          "1507165200":12.116,
          "1507168800":10.985,
          "1507172400":12.235,
          "1507176000":9.152,
          "1507179600":58.837,
          "1507183200":65.365,
          "1507186800":22.05,
          "1507190400":38.03,
          "1507194000":8.861,
          "1507197600":1.071,
          "1507201200":15.919,
          "1507204800":20.187,
          "1507208400":16.721,
          "1507212000":9.775,
          "1507215600":2.027,
          "1507219200":4.288,
          "1507222800":3.249,
          "1507226400":6.186,
          "1507230000":2.068,
          "1507233600":3.909,
          "1507237200":2.478
     },
     "units":"kW"
}
```

4 IMPLEMENTATION

The web-app of the WG StaaS/VPP tool allows the logged-in user to benefit from the different functions that each available functionality defines. In order to harness its architecture, it has been developed using the MEAN.JS full-stack JavaScript solution, this choice has been made to build fast, robust, and maintainable production web applications using MongoDB, Express, AngularJS, and adopting Node.js as engine. The web-app is exposed on the network thanks to Ngnix reverse proxy Different modules have been used to develop both the client and server parts of the WG Staas/VPP tool, below is a detail of these and for what purpose and then a brief description of the most important ones.

Modules adopted to develop the client side software:

- “Bootstrap v. 3.3.7” used to draw the front-end web-app interface.
- “Angular-ui-router v. latest” used for the page routing.

- “Ngstorage v. latest” used to manage the session and its data.
- “Angularjs-datepicker v.2.1.23” used to manage the popup of the date.
- “Angular-chart.js v. latest” to create and draw the charts inside the tool.
- “Ngmap v. latest” together whit google maps API (src="http://maps.google.com/maps/api/js?key=AIzaSyCtAWW-Sv99CiDFq5i4cYgE_0UBAuQBwXg") to create and manage the maps of the Map functionality of the tool.

Modules adopted to develop the server side software:

- “Amqplib v.0.5.2” to communicate with the WG IOP RabbitMQ istance.
- “Async v.2.6.0” to perform synchronous promise call.
- “Express-jwt” and “jsonwebtoken” for manage the authentication token.
- “Mongoose v. latest” to facilitate the interaction with mongoDB plugin.
- “Validator v. latest” to perform fields validation like email validation.

Mean [10]
MEAN.JS is a full-stack JavaScript solution that helps you build fast, robust, and maintainable production web applications using MongoDB, Express, AngularJS, and Node.js.

Angular-ui-router [11]
UI-Router is the defacto standard for routing in AngularJS. Influenced by the core angular router $route and the Ember Router, UI-Router has become the standard choice for routing non-trivial apps in AngularJS (1.x)

Express [12]
Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It provides a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy and a thin layer of fundamental web application features, without obscuring Node.js features.

Angular [13]
AngularJS is a structural framework for dynamic web apps. AngularJS is the MVC framework through which the DRMF cockpit web application side has been developed. Thanks to the fact that with AngularJS applications are defined with modules that can depend from one to the others, very powerful templates directly in RESCO tool HTML part have been defined with new attributes or tags and expressions, encapsulating the behavior of the RESCO tool in presenter. Moreover, thanks to the use of dependency injection provided by AngularJS, the test JavaScript code was structured very easily. Finally, AngularsJS allowed to use HTML as template language as well as to extend HTML's syntax to express RESCO tool application's components clearly and succinctly. Angular's data binding and dependency injection provided the development team with the necessary features to make the implementation faster and easier.

Node [14]
Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Nginx [15]

NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption.
NGINX is one of a handful of servers written to address the C10K problem. Unlike traditional servers, NGINX doesn't rely on threads to handle requests. Instead it uses a much more scalable event-driven (asynchronous) architecture. This architecture uses small, but more importantly, predictable amounts of memory under load. Even if you don't expect to handle thousands of simultaneous requests, you can still benefit from NGINX's high-performance and small memory footprint. NGINX scales in all directions: from the smallest VPS all the way up to large clusters of servers

Bootstrap CSS framework [16]
Bootstrap is a free and open-source front-end web framework for designing websites and web applications. It contains HTML- and CSS-based design templates for typography: forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.
Bootstrap is modular and consists essentially of a series of *Less stylesheets*[1] that implement the various components of the toolkit. A stylesheet called "Bootstrap less" includes the components stylesheets. Developers can adapt the Bootstrap file itself, selecting the components they wish to use in their projects.
The Less stylesheet language allowed the development team to use variables, functions and operators, nested selectors, as well as mixing for the implementation of the WG StaasVPP Web-app.

Mogoose [17]
MongoDB object modelling for Node.js® .

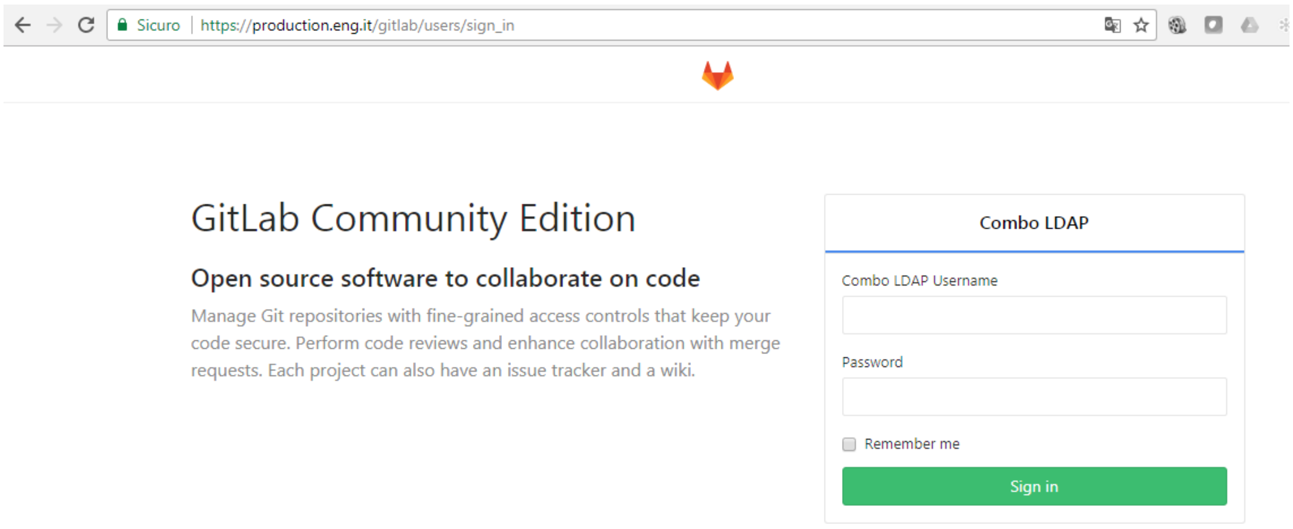
4.1 DEPLOYMENT

Along with the development view of the system component, a special focus is delivered on the deployment view of the WG StaasVPP tool. The minimum hardware and software requirements for the deployment of the full application are presented.

For hardware requirements, A 4-core (or superior) CPU at 2GHz, 8GB of RAM, 100 GB hard disk and a 64-bit CentOS 6.6 (core linux intsys-trentour 2.6.32-431.29.2.el6.x86_64) or higher is the minimum of requirement. There are 2 ways of deployment the services in local servers: either plain installation in the server machine (if this fulfils the minimum of hardware & software requirements) or setting a Virtual Machine for managing separately the different applications. This approach enables us to deploy the WG Staas/VPP tool either as a host service or as an application running in a cloud service provider (with the appropriate customizations).
The aforementioned deployment analysis defines the list of prerequisites for the demonstration of the WG StaasVPP tool at the different pilot sites (though as presented above we are fully flexible at any possible deployment). The current deployment of the WG Staas/VPP tool is available in a cloud server, managed by the ENGINEERING DATA CENTER of Pont-Saint-Martin (AO) as the leader of this task. The web-ui is currently accessible at the following url: http://wisegrid.esf.eng.it/staasvpp/login. This URL is managed by a Ngnix reverse proxy . The WEB-UI access is protected by username and password authentication.

The software developed and its versioning is saved on a Git repository offered by Engineering to all project partners. It is possible to get the last version of the RESCO tool linking on https://production.eng.it/gitlab/WiseGrid/StaasVpp by accessing with user provided user credentials.

The following Figure (Figure 36) shows the login page of the GitLab repository that hosts the source code of the WG Staas/VPP tool.


Sicuro | https://production.eng.it/gitlab/users/sign_in

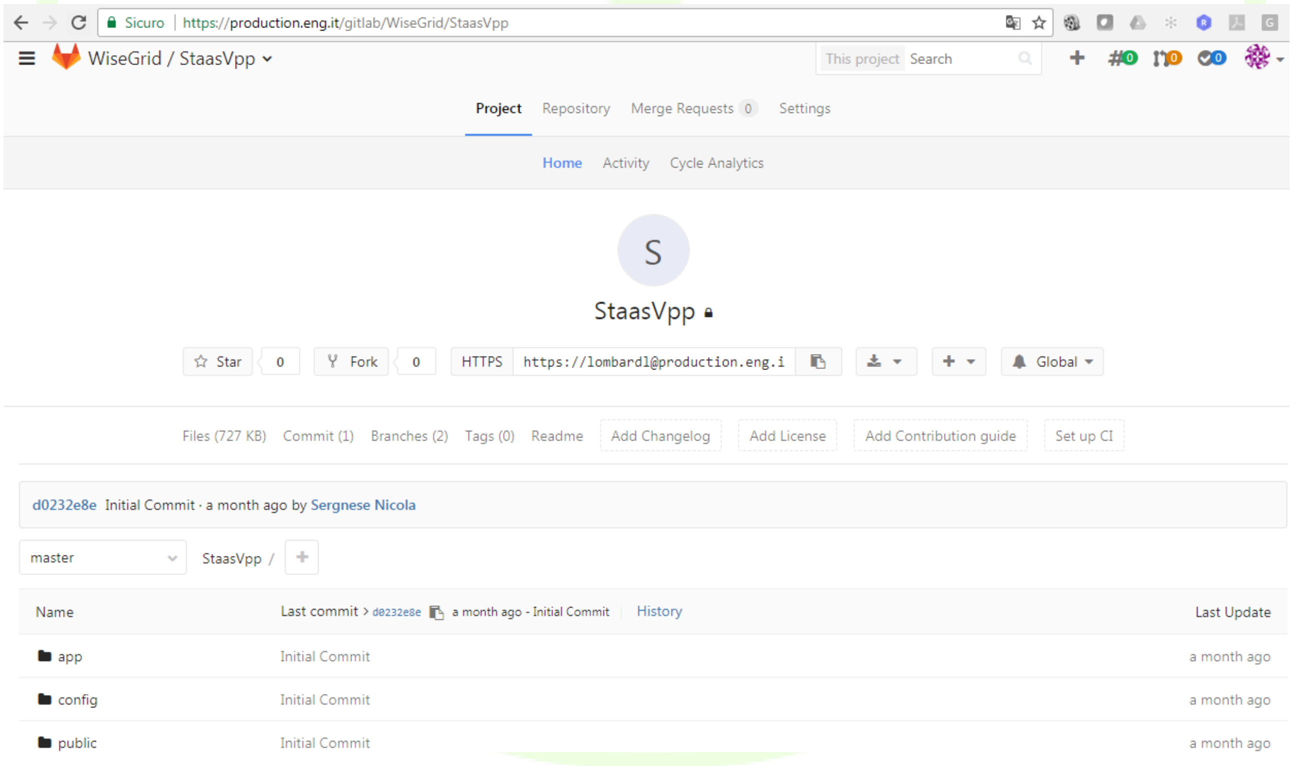
GitLab Community Edition

Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

Figure 36 – GitLab Login page

Figure 37 shows the repository tenant that hosts the source code of the WG Staas/VPP tool and its related source directory main structure.


Sicuro | https://production.eng.it/gitlab/WiseGrid/StaasVpp

WiseGrid / StaasVpp
This project Search
Project Repository Merge Requests 0 Settings
Home Activity Cycle Analytics
StaasVpp
Star 0 Fork 0 HTTPS https://lombardl@production.eng.i Global
Files (727 KB) Commit (1) Branches (2) Tags (0) Readme Add Changelog Add License Add Contribution guide Set up CI

d0232e8e Initial Commit · a month ago by Sergnese Nicola

master StaasVpp /

Name	Last commit > d0232e8e a month ago - Initial Commit History	Last Update
app	Initial Commit	a month ago
config	Initial Commit	a month ago
public	Initial Commit	a month ago

Figure 37 – GitLab Software Repository page

5 DESCRIPTION OF GRAPHICAL USER INTERFACE

5.1 LOGIN

The WG Staas/VPP tool is a web application that can be browsed from any browser. The tool is reachable at the following URL:

- http://wisegrid.esf.eng.it/staasvpp/login

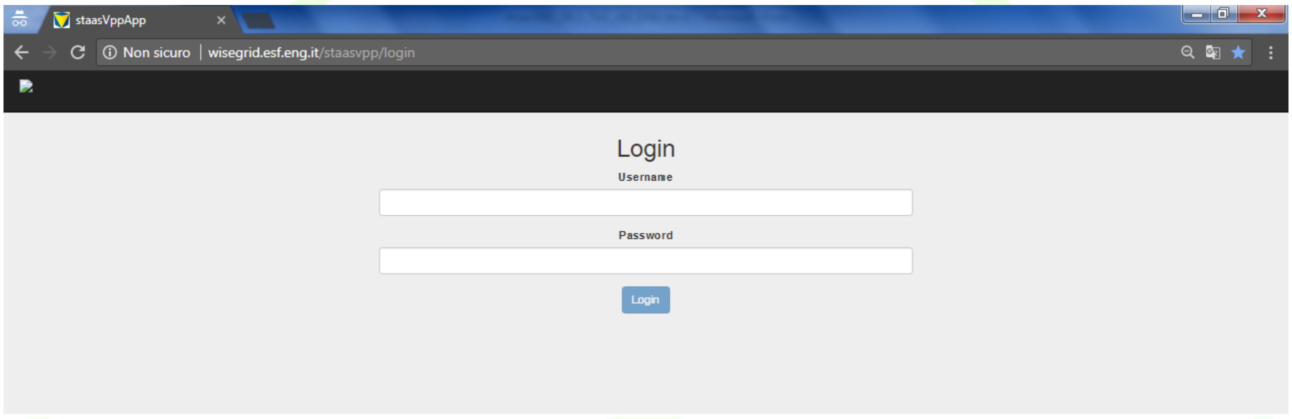


Figure 38 – Login page

Currently the login credentials are provided by the system administrator that create it after a specific user request. As a first step, user needs to login into the WG Staas/VPP, filling "*Username*" and "*Password*" fields and clicking the "*Login"* button. If the login credentials are correct, the logged user can have access to the functionalities provided by the tool.

5.2 DASHBOARD

The first feature that is offered after logging in is the user Dashboard, through this dashboard the logged user has an immediate view through graphs of the production status of prosumers relative to energy produced by batteries and RES and a view of Incomes in euro. These data are obtained by performing queries to the big data platform where they are stored.

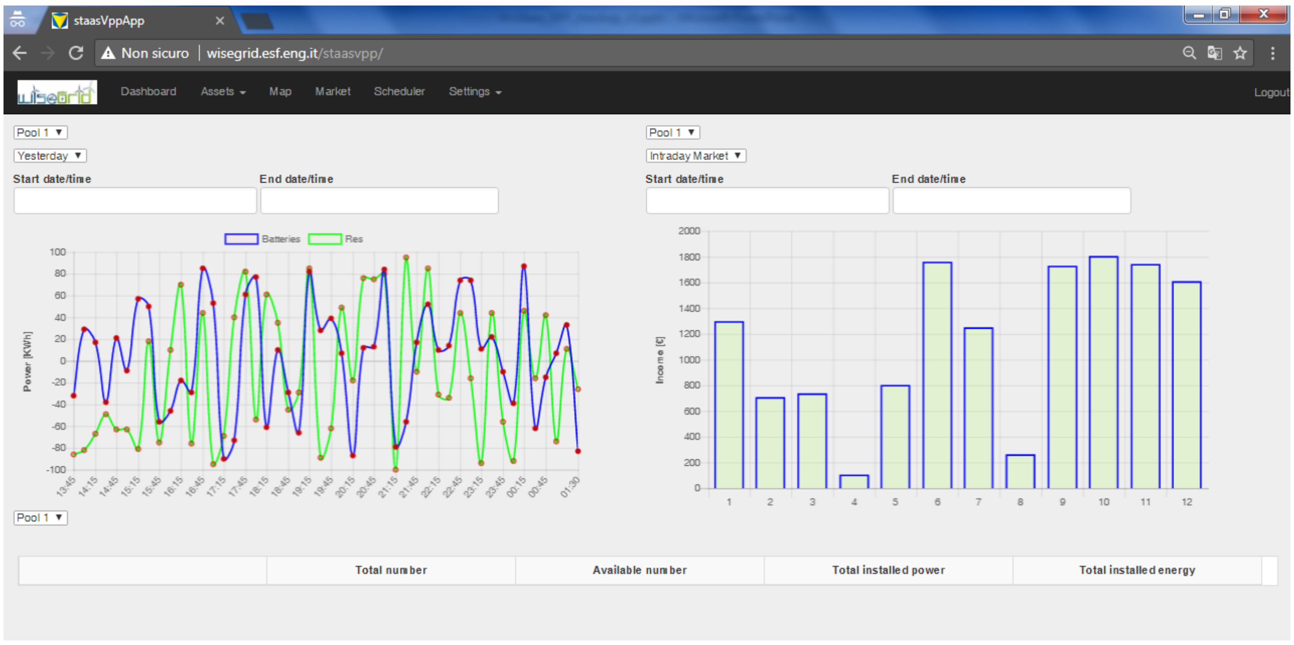


Figure 39 – Dashboard

Two different graphs give the user of the tool the opportunity to visualize this data by applying some search filters.

The Batteries and RES graph can be filtered by:

- Pool: different assets of the VPP can be allocated to a pool, e.g. for geographical clustering. In Wise-GRID it makes sense to create pools for the different pilot sites.
- Start Date and End Date of data illustration. For an easier navigation a selection box is included with which one can easily choose relevant time periods (e.g. yesterday)

It is also possible ide one the two curves by clicking on the Batteries or RES legend on the upper side of the graph to obtain a more readable graph. To have a precise view of a value at a specific point on the graph curve, the WG Staas/VPP user can click on it and the detail of the selected point will be shown.

The Incomes graph can be filtered by:

- Pool
- Market type (DSO Ancillary Service, Wholesale Market, TSO Ancillary Service)
- Start Date and End Date related to Incomes period

It is also possible to hide one the two curves by clicking on the Batteries or RES legend on the upper side of the graph to obtain a more readable graph. To have a precise view of a value at a specific point on the graph curve or histogram, the WG Staas/VPP user can click on it and the detail of the selected point will be shown as showed in the next images.

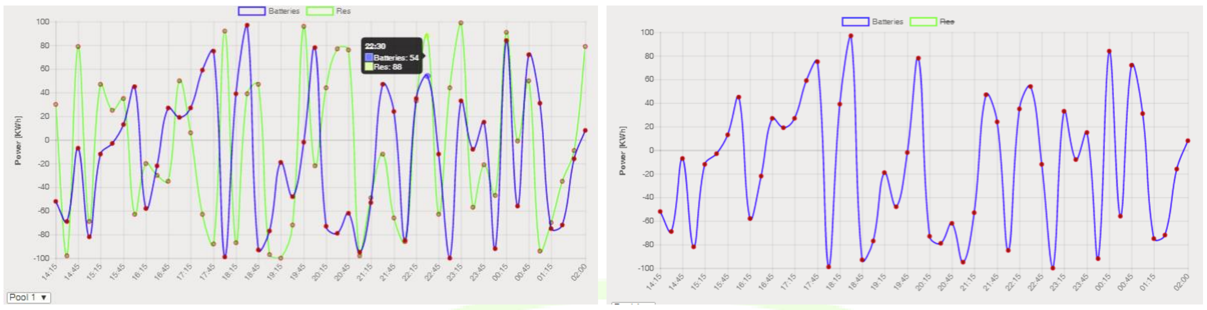


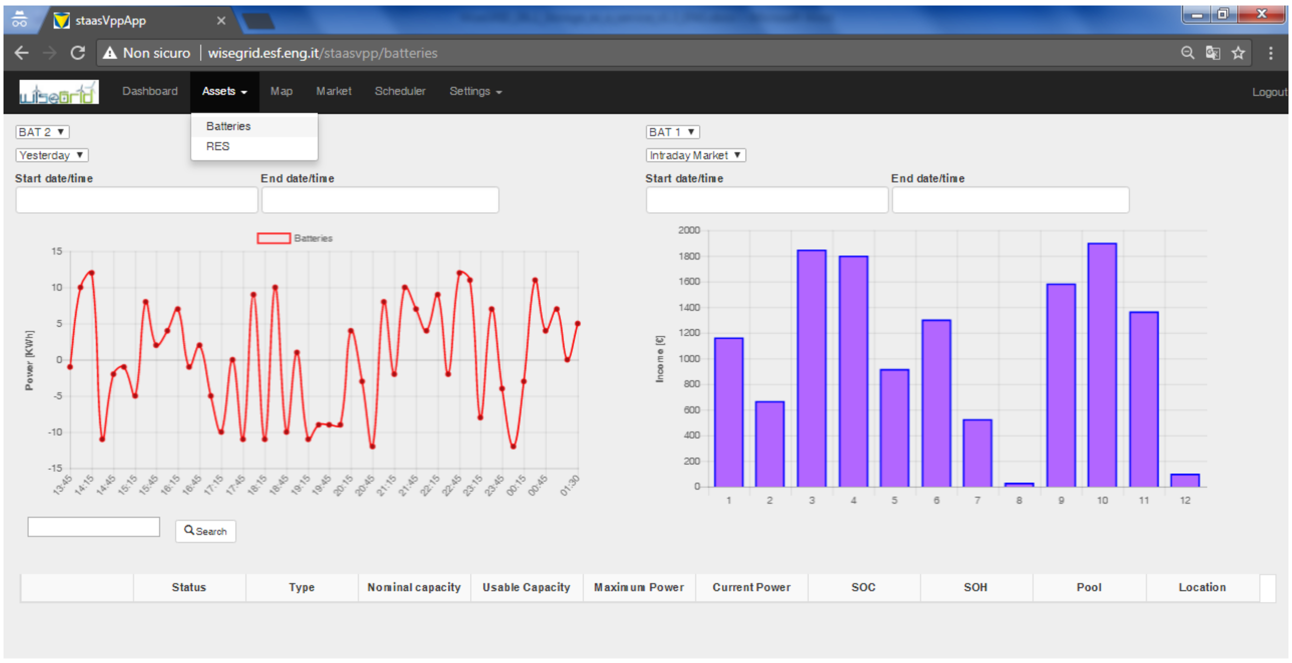
Figure 40 – Graphs detail

The lower side of the Dashboard give to the user an overview of each selected pool by listing in a tabs such information like:

- Total number: This the total number of batteries and RES units that are connected to the WG StaaS/VPP
- Number available: number of batteries and RES units that are currently available
- Total installed nominal power: This is the aggregated total power of all batteries and RES
- Total installed energy: This is just valid for storage systems. It represents the sum of all installed battery capacities.
- Current power: This is aggregated power of all batteries resp. RES

5.3 ASSETS BATTERIES/RES

The Assets functionalities available by the “Assets” menu is divided into two sub-menu, one for the Batteries asset and one for the RES asset. This choice was made to separate the two typology of assets available for the WG Staas/VPP tool and to have two light user interface instead of only one with too many information. The two functionalities are very similar for the graphs design, however the recap table at the bottom side of the page shows different information related to the different nature of the two kind of assets. The main difference with the dashboard (it seems very similar) is that in these two pages the data showed are not aggregated for “pool” but are for each singular asset and the summary table shows information by type of asset.



	Status	Type	Nominal capacity	Usable Capacity	Maximum Power	Current Power	SOC	SOH	Pool	Location

Figure 41 – Asset – Batteries

The Batteries graph can be filtered by:

- Battery ID
- Start Date and End Date of batteries measurement

To have a precise view of a value at a specific point on the graph curve, the WG Staas/VPP user can click on it and the detail of the selected point will be shown.

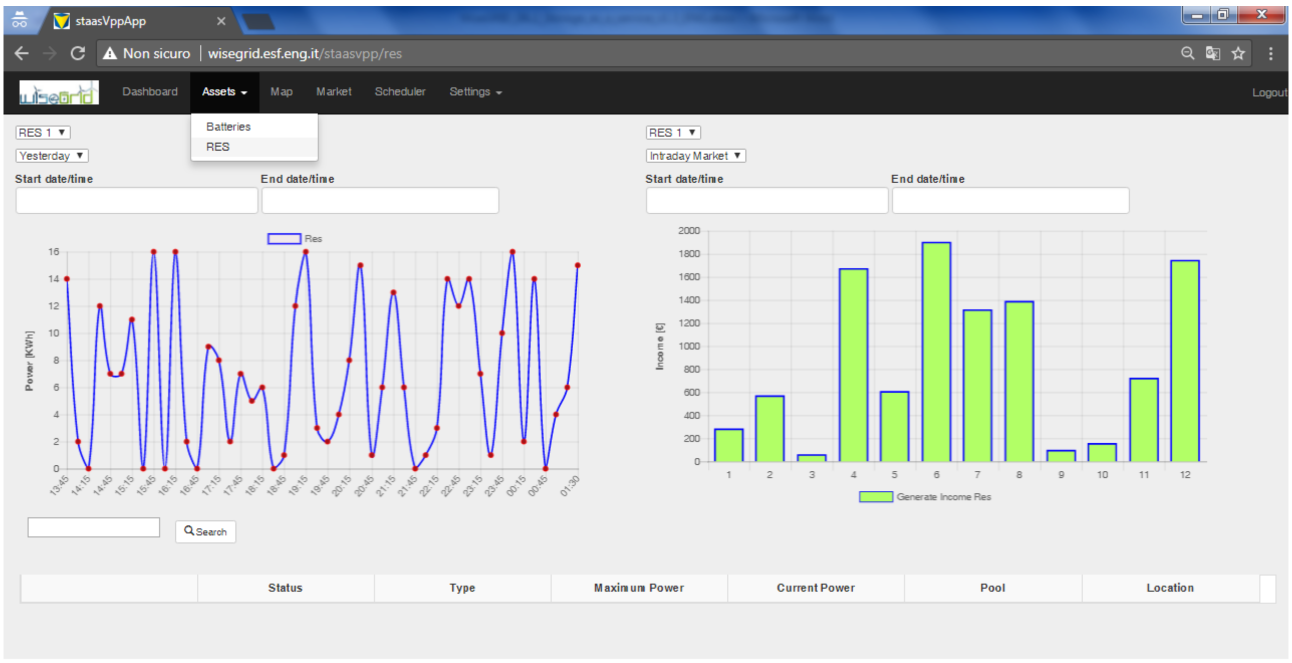
The Battery Incomes graph can be filtered by:

- Battery ID
- Market typology
- Start Date and End Date related to Incomes period

The lower side of the Dashboard give to the user an overview of each selected Battery by listing in a tabs such information like:

- Status
- Type
- Nominal capacity
- Usable capacity
- Maximum power
- Current power
- SOC
- SOH

- Pool
- Location



	Status	Type	Maximum Power	Current Power	Pool	Location

Figure 42 – Assets – RES

The RES graph can be filtered by:

- RES ID
- Start Date and End Date of RES measurement

To have a precise view of a value at a specific point on the graph curve, the WG Staas/VPP user can click on it and the detail of the selected point will be shown.

The RES Incomes graph can be filtered by:

- Battery ID
- Market typology
- Start Date and End Date related to Incomes period

The lower side of the Dashboard give to the user an overview of each selected Battery by listing in a tabs such information like:

- Status
- Type
- Maximum power
- Current power
- Pool

- Location

5.4 MAP

The “Map” functionality available by the Map menu allow the logged user to have a quIck view of the points on the map where the Battery and RES installation are located. The user can navigate through the map, zoom in on the image for more details about the place. The map can be filtered to see all the Assets supply points or alternatively the Batteries or the RES supply points.

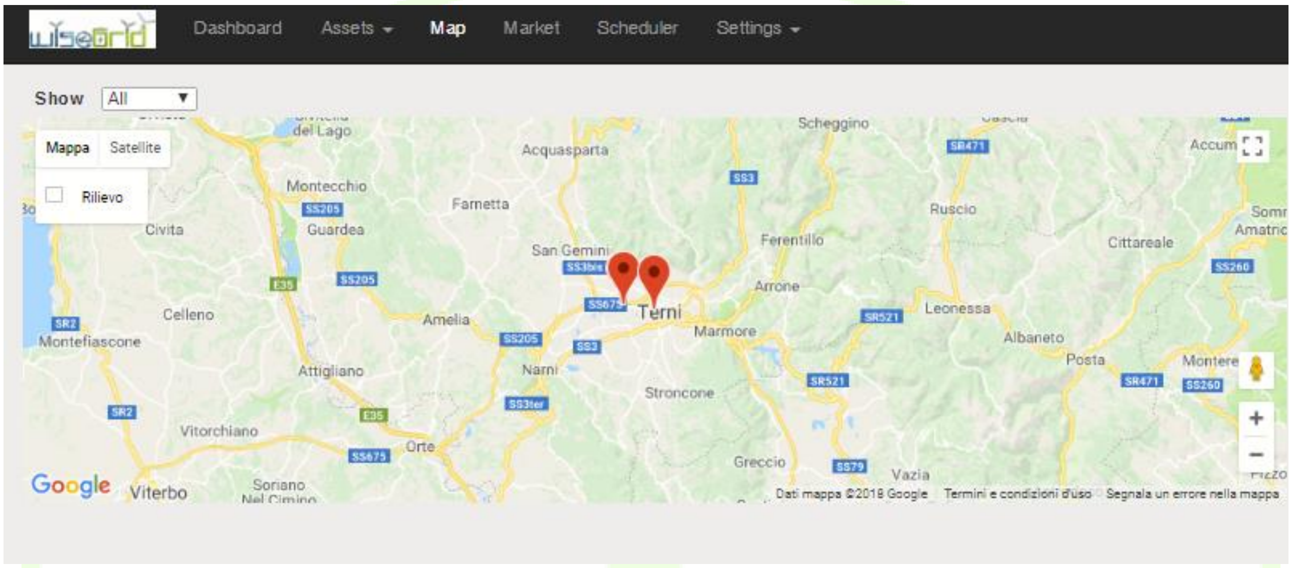


Figure 43 – Map

It is also possible to view detailed information for each individual supply point identified in the map by clicking on it. A pop-up will show some detailed information like for example the name of the owner of the supply point as showed in the next image.

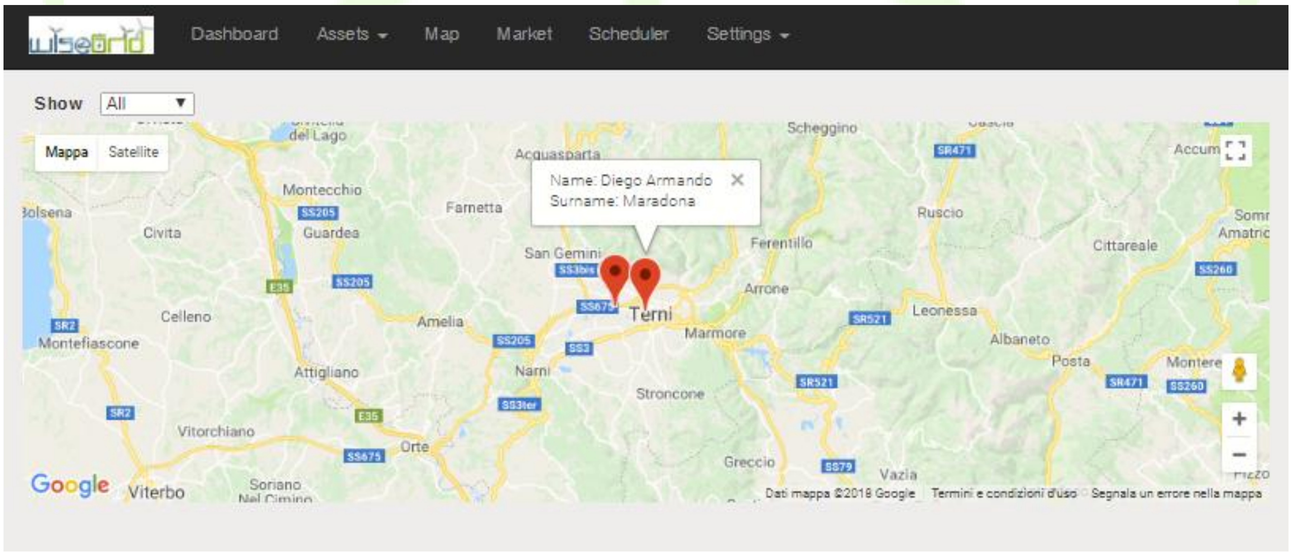


Figure 44 – Map Supply point Detail

5.5 MARKET

In this section the offer of capacities is managed. For each market and for testing purposes a subsection is intended. The most important subsection – the interface to the DSO – is illustrated in the Figure below. Via this interface the aggregator is informed about DSO requests, in particular the requested power, the time period and the location. Moreover, the aggregator is informed about the current available flexibilities with which a first rough estimation can take place if the DSO requests can be met. The aggregator then can choose in which period capacities should be offered and state a price. Based on this input data and via a forward simulation it is checked in more detail if enough capacities are available to fulfil the request. In case of a positive evaluation the capacities can be manually offered by clicking the button "Offer".

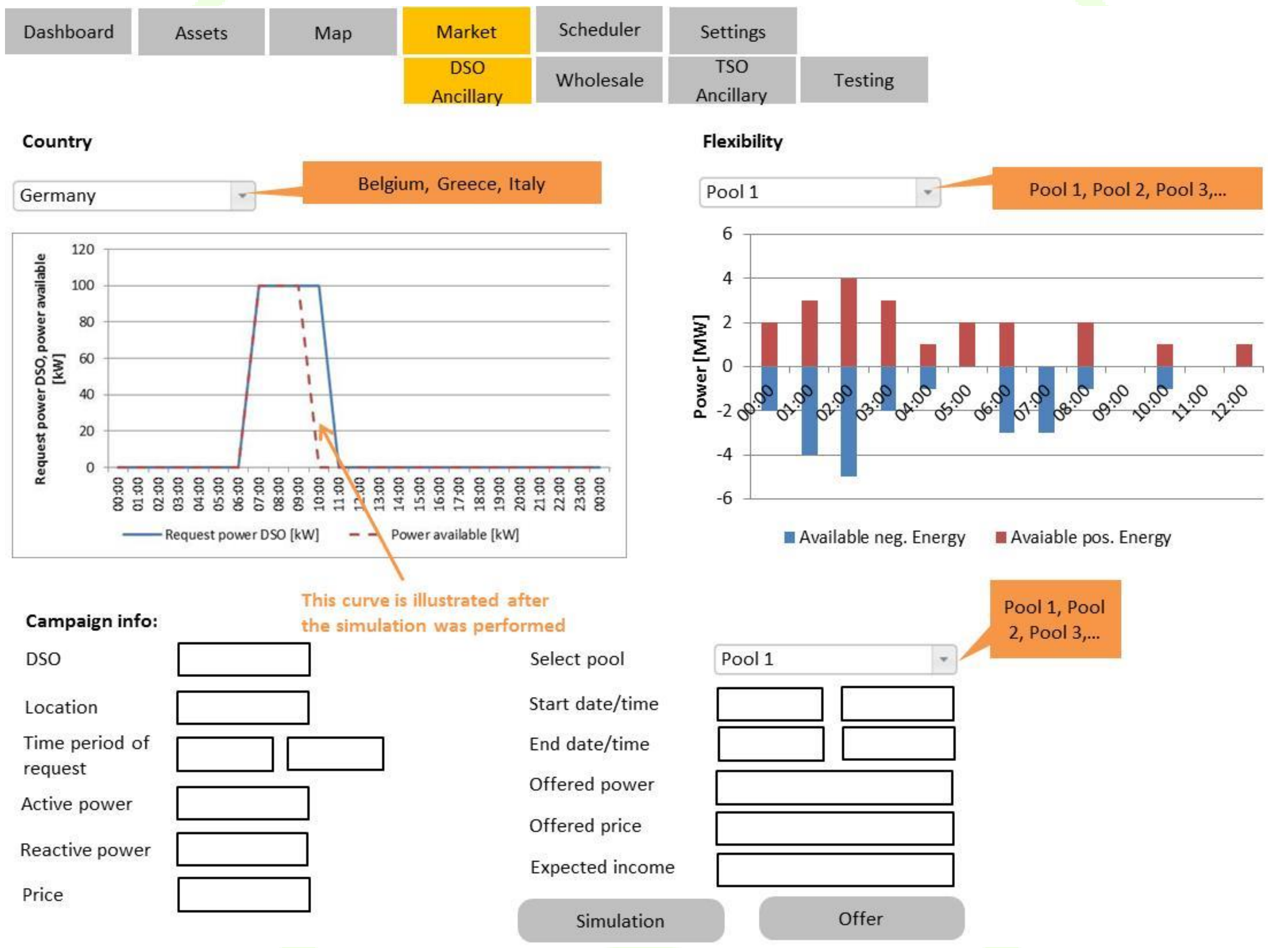


Figure 45 – Mockup for interface to DSO in Market section

For testing purposes, a specific subsection will be designed. By defining the power and price in each time, the necessary power profile input for the scheduler can be set manually. In the following Figure the setup is illustrated.

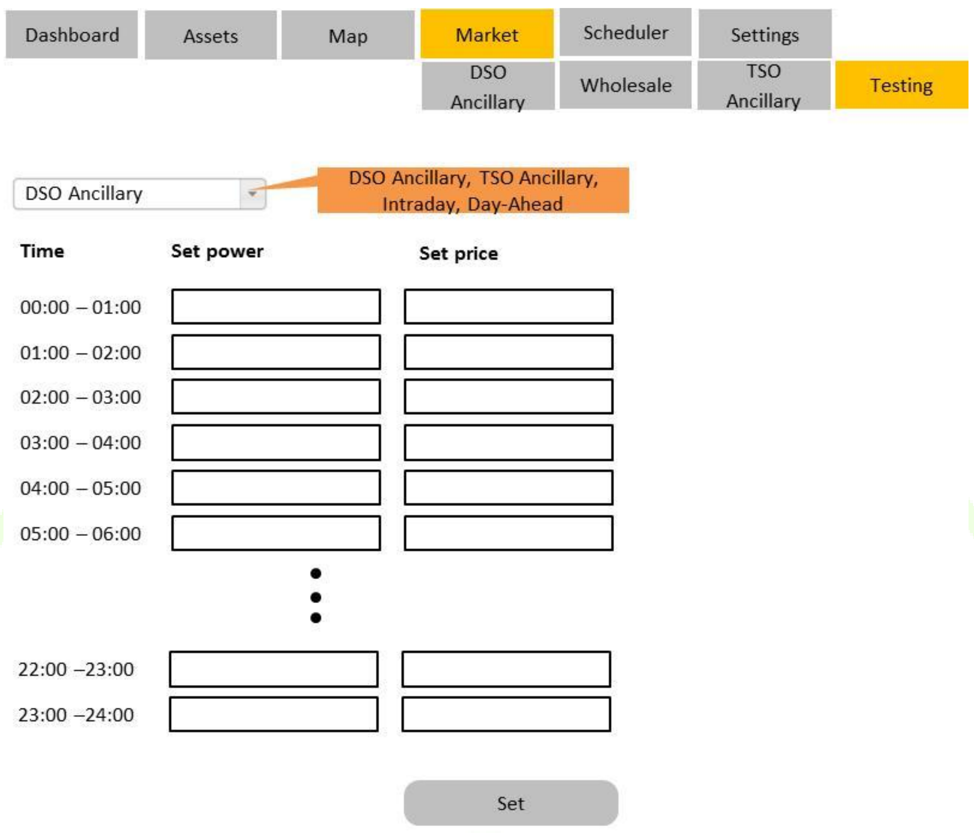


Figure 46 – Market testing subsection

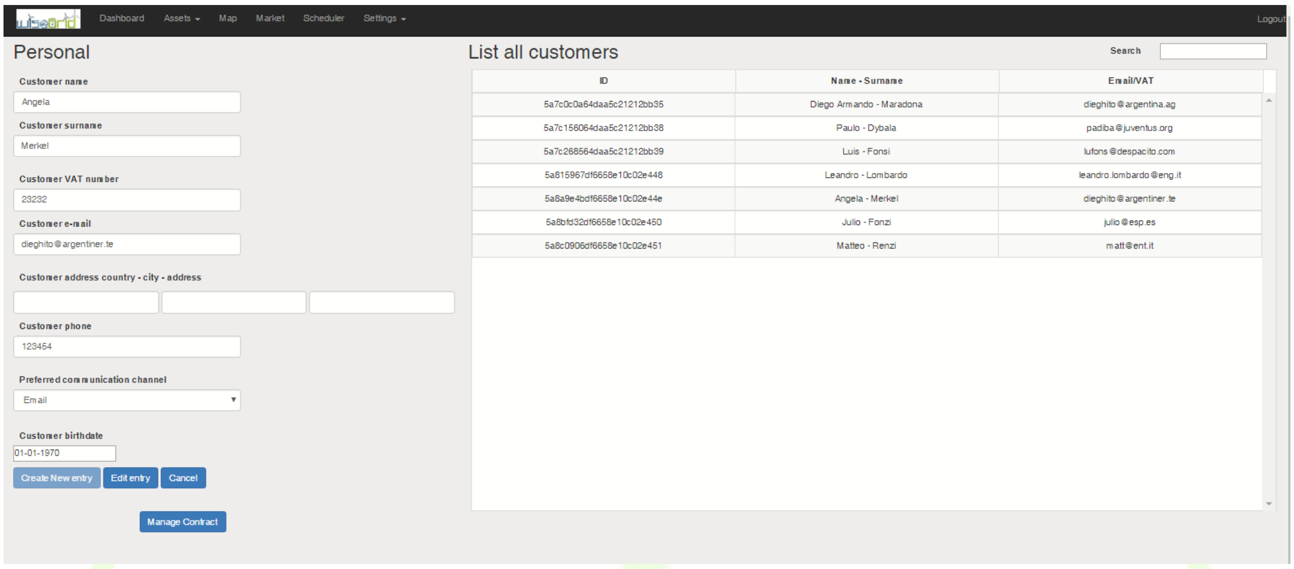
5.6 SETTINGS – CUSTOMER MANAGE/EQUIPMENT INVENTORY

5.6.1 Customer manage

By the "Customer manager" functionality available through the Settings menu the WG Staas/VPP logged in user can register a new customer to the system and update its attributes or delete it. The typical attribute that identify a Personal into the WG Staas/VPP tool are name, surname, address, birth date, preferred communication channel, vat number and email. Each of these fields have a specific role into the tool like to know the address to send a written communication, the email address to send communication and other information like contract variation etc.

The functionality, besides giving the possibility to create and modify a new customer within the application, provides a summary table of all customers present in the system with the salient information and allows a search using the search functionality at the top of the table.

It is possible to search for any of the values in the table, the system will automatically filter the result showing those that meet the filter directly in the table itself.



Personal

Customer name	Angela
Customer surname	Merkel
Customer VAT number	23232
Customer e-mail	dieghito@argentiner.te
Customer address country - city - address	
Customer phone	123454
Preferred communication channel	Email
Customer birthdate	01-01-1970

List all customers

Search

ID	Name - Surname	Email/VAT
5a7c0c0a64daa5c21212bb35	Diego Armando - Maradona	dieghito@argentina.ag
5a7c156064daa5c21212bb38	Paulo - Dybala	padiba@juventus.org
5a7c268564daa5c21212bb39	Luis - Fonsi	lufons@despacito.com
5a815967df6658e10c02e448	Leandro - Lombardo	leandro.lombardo@eng.it
5a8a9e4bdf6658e10c02e44e	Angela - Merkel	dieghito@argentiner.te
5a8bfd32df6658e10c02e450	Julio - Fonzi	julio@esp.es
5a8c0906df6658e10c02e451	Matteo - Renzi	matt@ent.it

Figure 47 – Customer manage page

This functionality is also a point of access for the Manage Contract functionality that is described in the next paragraph. The link is represented by the Manage Contract button that is enabled only when a customer has been selected from the List all customer table.

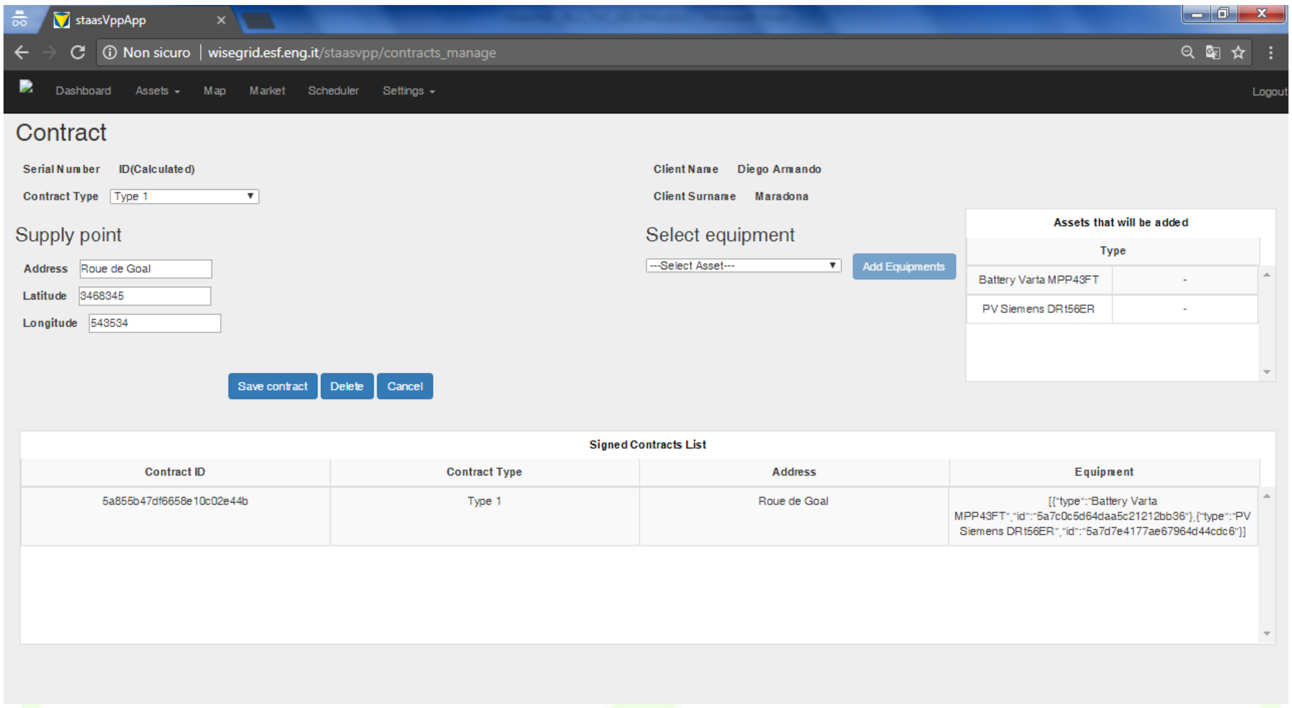
5.6.1.1 Contract manager

By the “Contract manager” functionalities available through the menu ”Manage contract” button of the Customer manage functionality the WG Staas/VPP logged in user can register , manage a contract with a specific customer registered at the system.

A customer may have several contracts (listed into the Signed contract list table), each of them linked to a particular dwelling (supply point). The main contracts information are the contract supply point location (address, latitude, longitude) to identify it in the map, references of installed assets like Batteries or RES (listed into the Asset list table) and the contract type. The exact design of the contracts is part of the current investigations. It is imaginable that different types of contracts exist. Two examples are stated in the following Figure:

- Contract Type 1: under this contract type, the WG StaasVPP trades the whole production of the assets installed in customer premises.

* Contract Type 2: under this contract type, the customer is allowed to self-consume energy and the WG Staas/VPP trades the production surplus and available capacities of the storage systems.



Contract

Serial Number	ID(Calculated)
Contract Type	Type 1
Client Name	Diego Armando
Client Surname	Maradona

Supply point

Address	Roue de Goal
Latitude	3468345
Longitude	543534

Select equipment

---Select Asset---

Assets that will be added

Type	
Battery Varta MPP43FT	-
PV Siemens DRt56ER	-

Signed Contracts List

Contract ID	Contract Type	Address	Equipment
5a855b47df6658e10c02e44b	Type 1	Roue de Goal	[{"type":"Battery Varta MPP43FT","id":"5a7c0c5d64daa5c21212bb36"},{"type":"PV Siemens DRt56ER","id":"5a7d7e4177ae67964d44cdc6"}]

Figure 48 – Contract manage page

5.6.2 Equipment inventory

By the Equipment inventory functionality showed in the next image the logged in user can add a new Equipment to the toll by filling several attribute fields like the System type (eg: Battery - PV – Wind – CHP), the System brand, the System model and so on with other relevant attribute. These equipments can be modified by the user, deleted (if not linked to a contract yet) and searched by specific functionalities able to filter the data present into the summary list without specify the column to filter (the system automatically search the word to search between the available one into the columns values).

The Equipment inventory function available from a sub-menu of the Settings menu allows the logged in user to manage all possible assets at his disposal and that will be installed at the various supply points after signing a contract. The main choice that the user must make is the choice of the “System type” of asset, on this depends the mandatory or not of all those fields in the functionality that can be related or not to the type of asset. The main attributes that can be filled are for example:

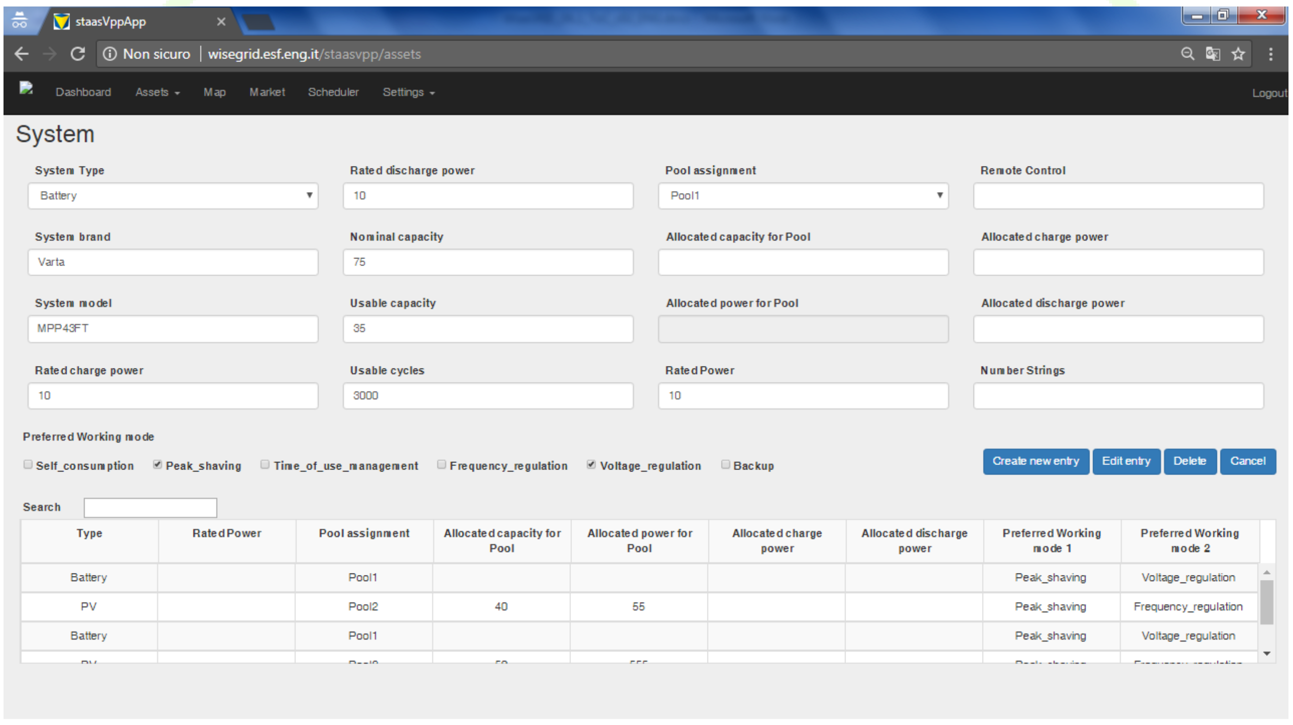
For a PV/Wind/CHP:

* System brand
* System model
* Allocated power for pool
* Rated power

- Remote control

For a Battery

- Rated charge power
- Rated discharge power
- Allocated charge power
- Allocated discharge power
- Usable cycles
- Nominal capacity
- …..



System

System Type: Battery	Rated discharge power: 10	Pool assignment: Pool1	Remote Control:
System brand: Varta	Nominal capacity: 75	Allocated capacity for Pool:	Allocated charge power:
System model: MPP43FT	Usable capacity: 35	Allocated power for Pool:	Allocated discharge power:
Rated charge power: 10	Usable cycles: 3000	Rated Power: 10	Number Strings:

Preferred Working mode

☐ Self_consumption ☑ Peak_shaving ☐ Time_of_use_management ☐ Frequency_regulation ☑ Voltage_regulation ☐ Backup

Search

Type	Rated Power	Pool assignment	Allocated capacity for Pool	Allocated power for Pool	Allocated charge power	Allocated discharge power	Preferred Working mode 1	Preferred Working mode 2
Battery		Pool1					Peak_shaving	Voltage_regulation
PV		Pool2	40	55			Peak_shaving	Frequency_regulation
Battery		Pool1					Peak_shaving	Voltage_regulation

Figure 49 – Equipment inventory page

The functionality, besides giving the possibility to create and modify or delete a new equipment within the application, provides a summary table of all equipment present in the system with the salient information and allows a search using the search functionality at the top of the table.

It is possible to search for any of the values in the table, the system will automatically filter the result showing those that meet the filter directly in the table itself.

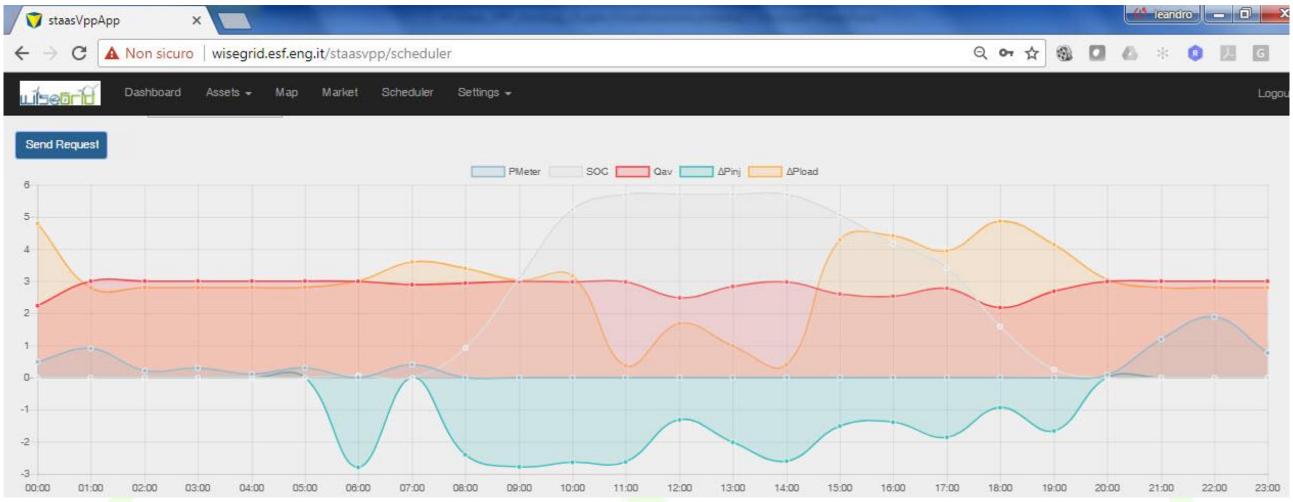


Figure 50 – Scheduler Flexibility algorithm page

6 CONCLUSIONS

The WG StaaS/VPP Component has been presented in this document. This component allows the integration of prosumers in the energy market offering demand response services by means of an optimal combination of production from distributed renewable energy sources, charge or discharge of energy storage systems and management of loads, such as peak shaving or load displacement.

To accomplish these objectives, the functionalities of the component have been divided in smaller parts or modules, each of them in charge of a specific commitment. A global architecture that ties all the different modules through communication interfaces has been designed, as a necessary step to allow the system to behave as a single entity; all these modules and the architecture that binds them have been implemented as a web application that is able to serve all the desired functionalities. The WG StaaS/VPP Component can be, with this approach, deployed and executed anywhere on the cloud or on dedicated servers, and is accessible from virtually anywhere where an internet connection is available. The use of up to three different energy storage systems from different vendors allow to validate the interoperability of the system; specific communication interfaces using the MQTT protocol have been defined and are publicly described for this purpose. Initial tests have been performed that validate this communication and its interoperability. Once specific MQTT adapters are developed by each system, the MQTT interface is common for all of them.

As stated, the WG StaaS/VPP Component represents a key development for the integration of Energy Storage resources in the electricity markets, providing an added value for these distributed systems, not only in economic terms, but also in efficiency and sustainability, since the available capacity from a (usually) renewable energy source is used to contribute to the balance of the electricity grid.

7 REFERENCES AND ACRONYMS

7.1 REFERENCES

[1] “USEF - Universal Smart Energy Framework,” [Online]. Available: https://www.usef.energy/.

[2] WiseGRID Consortium , “D2.1. WiseGRID requirements, Use Cases and pilot sites analysis,” 2017.

[3] WiseGRID Consortium, “D4.2. WiseGRID interoperable Integrated Process,” 2018.

[4] “H2020 ELSA Project (Energy Local Storage advanced system),” [Online]. Available: http://elsa-h2020.eu/.

[5] “paho.mqtt.cpp,” Eclipse, 2018. [Online]. Available: https://github.com/eclipse/paho.mqtt.cpp. [Accessed 5 April 2018].

[6] “http://nobelgrid.eu/,” [Online].

[7] “https://openweathermap.org/,” [Online].

[8] “https://solcast.com.au/,” [Online].

[9] X. Guo and J. Su, “Improved Support Vector Machine Short-term Power Load Forecast Model Based on Particle Swarm Optimization Parameters,” *Journal of Applied Sciences,* vol. 13, no. 9, pp. 1467-1472, 2013.

[10] [Online]. Available: http://meanjs.org/.

[11] “https://ui-router.github.io/,” [Online]. Available: https://ui-router.github.io/.

[12] [Online]. Available: http://expressjs.com.

[13] [Online]. Available: https://angularjs.org/.

[14] [Online]. Available: https://nodejs.org/en/.

[15] [Online]. Available: https://nginx.org/en/.

[16] [Online]. Available: https://getbootstrap.com/docs/3.3/css/.

[17] [Online]. Available: http://mongoosejs.com/.

7.2 ACRONYMS

Table 15 – List of Acronyms

Acronyms List	
BESS	Battery Energy Storage System
BRP	Balance Responsible Party
CNEA	Cascaded Neuro-Evolutionary Algorithm
DSO	Distribution System Operator
ESS	Energy Storage System
IoT	Internet of Things
PV	Photovoltaic
RES	Renewable Energy Source
StaaS	Storage as a Service
STLF	Short-Term Load Forecast
SVM	Support Vector Machine
USEF	Universal Smart Energy Framework
CUPS	Supply Point Universal Code
SOC	State of Charge
LCOS	Levelized Cost of Storage
FRR	Frequency Restoration Reserve
FCR	Frequency Containment Reserve
VPP	Virtual Power Plant

8 ANNEX A. MQTT INTERFACE MESSAGES

Base topic: **MQTT/battery/[UID of the battery]**

Base topic uniquely identifies the battery in the whole system

E.g. MQTT/battery/BAT0001

A “battery” in this context can be an individual, fully-integrated battery system (like VARTA pulse) or consist of several sub-units. In the latter case, SOC and other battery-specific parameters must represent an appropriate collective value for the complete system.

Whether the six BYES batteries at Terni are to be connected to WG Staas/VPP as one system with one UID or as six individual units with six individual UIDs is beyond the scope of this document and to be decided somewhere else.

The UID of each battery is configured during commissioning of the system or hard-coded into the system. There is not going to be a mechanism to dynamically assign or change UIDs via MQTT.

Topic for battery system data: **system**

Refer to Data Model page „Status & System“ for details.

The battery publishes upon connection to the MQTT broker a JSON message to this subtopic informing of the system information. This publication must be flagged as “retained” (i.e. broker will send it to any new subscriber automatically)

QoS-Level: 1

E.g. MQTT/battery/BAT0001/system

Exemplary payload (refer to Data Model for details):

```
{
        "SN": "0123456789A",
        "FWVersion": "1.0",
        "Power": 1.0,
        "Capacity": 1.0,
        "PVPower": 1.0
}
```

Topic for battery status: **status**

Refer to Data Model page „Status & System“ for details.

The battery publishes upon changes (or periodically) a JSON message to this subtopic informing of the status. This publication may be flagged as “retained” (i.e. broker will send the last status to any new subscriber automatically)

The battery must configure a *MQTT LWT message* updating the status to “disconnected” when the battery disconnects from the MQTT broker.

QoS-Level: 0

E.g. MQTT/battery/BAT0001/status

Exemplary payload (refer to Data Model for details):

```
{
        "MeterActivePower": 1.0,
        "MeterReactivePower": 1.0,
        "InverterActivePower": 1.0,
        "InverterReactivePower": 1.0,
        "InverterPVPower": 1.0,
        "ExternalPV": 1.0,
        "InverterBatteryPower": 1.0,
        "BatterySOC": 1.0,
        "BatterySOH": 1.0,
        "BatteryVoltage": 1.0,
        "MeterGridVoltage": 1.0,
        "MeterGridFrequency": 1.0,
        "Temperatures": 1.0,
        "CosPhi": 1.0,
        "ChargeDisp": 1.0,
        "DischargeDisp": 1.0,
        "Status": 1,
        "Alarms": 1,
        "InverterPVVoltage": 1.0,
        "WorkingMode": 1,
        "Load": 1.0
}
```

<u>Topic for battery commands: **command**</u>

Refer to Data Model page „Command“ for details.

The battery <u>subscribes</u> to this topic, where the control system will publish the commands to be executed by the battery. Publications on this topic shall not be marked as “retained” (i.e. the batteries will only receive commands in real-time)

QoS-Level: 1

E.g. MQTT/battery/BAT0001/command

Exemplary payload (refer to Data Model for details):

Params field must be adapted depending on the specific command parameters.

Id field uniquely identifies the command (to be used to acknowledge the execution)

```
{
        "Id": "b246798a-4574-4c02-9a3c-946407242f49",
        "Command": 0,
        "Params": {
                "WorkingMode": 1
        }
}
```

Topic for battery command acknowledgements: **command/ack**

Upon reception of a command, the battery publishes the result of the execution by publishing a message to this topic

QoS-Level: 1

E.g. MQTT/battery/BAT0001/command/ack

Exemplary payload (refer to Data Model for details):

Id field matches the *Id* field of the received command

Result field is a Boolean indicating result of the execution

Details field may accommodate further details of the result of the execution, depending on the command

```
{
        "Id": "b246798a-4574-4c02-9a3c-946407242f49",
        "Result": 1,
        "Details": {

        }
}
```