| Title: | Document Version: |
|---|---|
| D13.2 WiseGRID Cockpit implementation and lab-testing | 0.5 |

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

| Contractual Delivery Date: | Actual Delivery Date: | Deliverable Type*-Security*: |
|---|---|---|
| M21 (July 2018) | M21 (July 2018) | RD-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: |
|---|---|---|
| Alberto Zambrano | ETRA | WP13 |
| Álvaro Nofuentes | ETRA | WP13 |

| Authors (organization): |
|---|
| Álvaro Nofuentes (ETRA), Alberto Zambrano (ETRA), Foivos Palaiogiannis (ICCS), Ioannis Vlachos (ICCS),Georgina Asimakopoulou (ICCS), Panos Karafotis (ICCS), Stela Sarri (ICCS), Mihai Sanduleac (CRE), Catalin Chimirel (CRE), Julio César Díaz (ITE), Juan Pablo González (ITE), Amparo Mocholí (ITE), Javier Monreal (ITE), Pascual Mullor (ITE) |



| Abstract: |
|---|
| This document reports the work performed within Task 13.2 “WiseGRID Cockpit Implementation” and Task 13.3 “WiseGRID Cockpit lab-testing and refinement”, following the specifications and architecture designed in Task 13.1 and reported in D13.1. |



| Keywords: |
|---|
| Smart Grid, DSO, Cockpit, Distribution Grid, MV, LV, Lab-Testing, Integration, Implementation |

## Revision History

| Revision | Date | Description | Author (Organisation) |
| --- | --- | --- | --- |
| V0.1 | 22.06.2018 | ToC | Alberto Zambrano (ETRA) |
| V0.2 | 13.07.2018 | First Round of contributions | Alberto Zambrano (ETRA) |
| V0.3 | 30.07.2018 | Second round of contributions and ready for Peer Review | Alberto Zambrano (ETRA) |
| V0.4 | 02.08.2018 | Collection of the feedback received | Alberto Zambrano (ETRA) and Álvaro Nofuentes (ETRA) |
| V0.5 | 03.08.2018 | Ready for Submission | Alberto Zambrano (ETRA) and Álvaro Nofuentes (ETRA) |

INDEX

## List of Figures

## List of Tables

# EXECUTIVE SUMMARY


WG Cockpit is the WiseGRID technological product targeting small DSOs and micro grid operators, allowing them to control, manage and monitor their own network, using flexibility to improve stability and security of their network. Taking into account the goals of the project, the features to be implemented within WG Cockpit consider a scenario of increasing share of distributed energy resources (DER) and services provided by communities of prosumers (aggregated in the form of VPPs or cooperatives) in order to achieve higher participation and environmental, social and economic benefits.

Taking into consideration the aim of this tool, the respective architecture and modules were designed also having in mind the design of the other WiseGRID tools. Furthermore, having in mind the process of developing a software, this document explains the implementation and lab-testing activities performed for assuring the quality of the tool previously designed.

For that purpose, firstly was needed to put in common the terminology to be used during the lab-testing phase in order to assure that all the partners involved in this stage, work in the same direction and there are no misunderstandings. Then, it was established the test plan which basically consists in the following 5 steps:

1) Review the project requirements and use cases
2) Define the features to be tested from those and classify them into test groups
3) Detail test cases for validation of named features
4) Execute the test cases
5) Document the test protocols

Moreover, for assuring the coherence and the easy understanding of each test, the following template was created, which summarizes the most important information to be shown and the features to be tested.


**Table 1 – Test case specification template**

<table>
<tr><td>Name</td><td colspan="3">The test case code and name which is unique to the project.</td></tr>
<tr><td>Module under test</td><td>The devices or systems under test</td><td>Resp.</td><td>Main partner responsible for the test</td></tr>
<tr><td>Module requirement</td><td colspan="3">The requirement, use case, or certification rule which is validated by the test case</td></tr>
<tr><td>Test environment</td><td colspan="3">List of elements needed for the test execution</td></tr>
<tr><td>Features to be tested</td><td colspan="3">List of features to be tested</td></tr>
<tr><td>Features not to be tested</td><td colspan="3">Optional</td></tr>
<tr><td>Preparation</td><td colspan="3">Short list of steps needed for preparing the test environment for test execution</td></tr>
<tr><td>Dependencies</td><td colspan="3">(Optional) List of test case codes defining test cases which need to be passed before the test case at hand can be started</td></tr>
<tr><td>Steps</td><td colspan="3">Testing procedures</td></tr>
</table>

| | |
|---|---|
| **Pass criteria** | *Expected (measurable) results, allowing to unambiguously judge if the test is passed or not passed (i.e. the product requirement was validated or not validated)* |
| **Suspension criteria** | *(Optional) Conditions under which continuation of the test is considered pointless because testing results would be invalid* |
| **Results** | *(Optional) Short list of results* |

Anyway, before starting the lab-testing phase, the implementation activities were performed in order to correctly integrate all the modules of the tool.

The following table below lists all the test cases and test results covered in the document at hand:

Table 2 – WG Cockpit test cases

| Test case code | Test case | Test result[1] |
|---|---|---|
| **GPA001** | Import topology from CIM model | ✓ |
| **GPA002** | Simulation model is composed in the UI | ✓ |
| **GPA003** | Server receives simulation request from internal ESB. | ✓ |
| **GPA004** | Server computes and delivers simulation results to internal ESB | ✓ |
| **GPA005** | Simulation results are shown in the UI | ✓ |
| **TRP001** | Static grid data is retrieved from structural database. | * |
| **TRP002** | Data is retrieved from long-term database. | * |
| **TRP003** | Server receives requests from internal ESB. | * |
| **TRP004** | Server computes and delivers results to internal ESB. | * |
| **RTM001** | Read smart meter data from IOP | ✓ |
| **RTM002** | Store smart meter data to Long-term DB | ✓ |
| **KPI001** | Energy delta calculation | ✓ |
| **KPI002** | Aggregated production | ✓ |
| **KPI003** | Aggregated demand | ✓ |
| **KPI004** | Voltage deviation | ✓ |
| **KPI005** | Frequency deviation | ✓ |
| **PFM001** | Import topology from CIM model | ✓ |
| **PFM002** | Measurements from buses are read from long-term database | ✓ |
| **PFM003** | Calculation request is read from internal ESB. | ✓ |
| **PFM004** | Results are delivered to internal ESB. | ✓ |
| **PFM005** | Generation of results as virtual meters | ✓ |
| **SE001** | Import topology from CIM model | ✓ |
| **SE002** | Measurements from buses are read from long-term database | ✓ |
| **SE003** | Calculation request is read from internal ESB. | ✓ |
| **SE004** | Results are delivered to internal ESB. | ✓ |
| **SE005** | Generation of results as virtual meters | ✓ |

[1] ✓: Test passed as originally planned, *: Pending

| | | |
|---|---|---|
| **FOR001** | Demand/production forecasting training | ✓ |
| **FOR002** | Demand/Production forecasting | ✓ |
| **FOR003** | Request message parsing test of WG Cockpit forecast module | ✓ |
| **FOR004** | Forecast response message generation test of WG Cockpit forecast module | ✓ |
| **FOR005** | Forecast is periodically triggered | ✓ |
| **FOR006** | Forecast results are saved to operational DB | ✓ |
| **CF001** | Congestion forecast is periodically triggered | ✓ |
| **TM001** | Read smart meter data from internal ESB | ✓ |
| **TM002** | Detection of threshold surpass on voltage | ✓ |
| **TM003** | Detection of current surpass on line | ✓ |
| **OD001** | Read smart meter data from internal ESB | ✓ |
| **OD002** | Detection of outlier on voltage | ✓ |
| **OD003** | Detection of threshold surpass on frequency | ✓ |
| **PQM001** | Measurements are read from long-term database. | * |
| **PQM002** | Calculation request is read from internal ESB. | * |
| **PQM003** | Results are delivered to internal ESB. | * |
| **EMS001** | Read and publish data from structural database and forecast module. | * |
| **EMS002** | Calculation request is read from internal ESB. | * |
| **EMS003** | Results are delivered to internal ESB. | * |
| **GFM001** | Configuration of custom workflow | ✓ |
| **GFM002** | Configuration of custom workflow | ✓ |
| **GFM003** | Custom workflow triggers action on external module | ✓ |
| **FLI001** | Location of fault event on network | ✓ |
| **FLI002** | Isolation and Restoration of network after fault event | ✓ |
| **FLI003** | Automatic execution of FLISR module | ✓ |

Some tests cases are still in progress due to the prioritisation of more essential and/or necessary functionalities for the DSO and the complexity of the implementation of this tool. In any case, these test cases will be performed and their outcome documented in a future deliverable.

All these activities have been supported by the setup of a virtual environment which replicates up to the possible extent the conditions that will be found in the deployment of the applications in the different pilot sites. The lab-testing platform consists of a couple of virtual machines running in the VMWare vSphere infrastructure of ETRA I+D.

# 1 INTRODUCTION

## 1.1 PURPOSE OF THE DOCUMENT

The purpose of this document is to summarise the results from Tasks 13.2 “WiseGRID Cockpit implementation” and 13.3 “WiseGRID Cockpit lab-testing and refinement”. In these tasks, the WG Cockpit design and development within Task 13.1 “WiseGRID Cockpit design”, is verified in a controlled environment before deploying it at the pilot sites.

## 1.2 SCOPE OF THE DOCUMENT

This deliverables covers the development of the WG Cockpit during its implementation and lab-testing phase, including an overview of the designed architecture in order to make aware the reader about the previous work performed. In this way, this document describes the test cases that were performed to validate the WG Cockpit framework before deploying it at the pilot sites.

## 1.3 STRUCTURE OF THE DOCUMENT

The document starts with the settlement of the lab-testing basis that will be use for the evaluation of the test cases. Then, the document continues with the explanation of the implementation of the different WG Cockpit modules. After this implementation, it starts the lab-testing phase, in which are described the different tests done to evaluate the performance of the tool and their results. Finally, it is included a section for extracting the conclusions of these tasks and settle the next steps to be follow.

## 2 LAB-TESTING APPROACH

The lab-testing approach that has been followed for this tool is the same one that was followed in the NOBEL GRID project. This methodology has demonstrated that is successful for this kind of projects so it has been properly studied and used but taking into account the particularities of the WiseGRID project.

### 2.1 DEFINITION OF TERMS

In order to provide a common methodology for testing WiseGRID tools, a common definition of terms was used. The following definitions were developed considering the state of the art in software, smart grid and system integration testing, especially with respect to the IEEE 829 Standard for software test documentation [1] [2] [3].

**Table 3 – Definition of terms**

| Term | Definition |
|---|---|
| **Device under test** | a product or software which is verified by a certain test case. It is part of the test environment |
| **Expected results** | a description of the status of the test environment after a test case was carried out and pass criteria have been met |
| **Features (not) to be tested** | a list of product requirements or specifications which are (not) covered by a certain test case |
| **Pass/fail criteria** | a definition of how to judge or measure if a product under test conforms to specifications and requirements that shall be validated by a certain test case |
| **Retesting** | re-execution of a test case that previously returned a “fail” result, to evaluate the effectiveness of intervening correction actions |
| **Subsystem acceptance criteria** | conditions to be fulfilled by a subsystem for including it into the system integration test. Conditions should include the availability of testing protocols for standalone subsystem tests. Also, subsystems should have similar level of maturity |
| **Suspension criteria** | a description of conditions which indicate that the test was carried out incorrectly or that any situation was produced which renders the testing results unusable, making test continuation pointless and requiring the test to be halted and restarted |
| **System integration test** | a test designed to verify that a system made up of two or more interacting products (subsystems) conforms to system-wide specifications and requirements. The device under test is the system itself. It is specially designed for finding inconsistencies which emerge only through the subsystem interaction. The system integration test plan may define partial system integration tests which allow for adding subsystems subsequently |
| **(System integration test) Level** | The number of system layers which are included in a system integration test case minus one |
| **System layer** | a group of one or more subsystems which is defined prior to the system integration test. According to group definition for a given system should be used for all system integration test cases |
| **Testing** | set of activities conducted to facilitate discovery, validation and/or evaluation of properties of one or more test WiseGRID components |
| **Test analysis** | elaboration about why a test result emerged. It may also include a conclusion about what the test result implies for the future work |
| **Test case** | a collection of features (not) to be tested, testing procedures, and pass/fail criteria used for testing a system or device under test. Test cases may refer to specific types of product requirements, e.g. the function, reliability, stability, safety, or vulnerability. Test cases may be applied to different test environments, e.g. the same test case may be applied to different pilot sites |
| **Test case code** | an identifier for a test case which is unique throughout the project, e.g. WP13:03 |
| **Test case** | documentation of one or more test cases |

| | |
|---|---|
| **specification** | |
| **Test coverage** | a list of product requirements or specifications which are verified by a test plan |
| **Test data** | data created or selected which is needed for executing one or more test cases. It may be defined in the test case specification |
| **Test environment** | a list of all elements (software, hardware, information, external conditions) needed to carry out a test case, including the device or system under test and all elements needed to judge the test outcome |
| **Test environment set-up process** | a list of actions needed for establishing and maintaining a required test environment |
| **Test execution** | the actions needed to carry out the testing procedures for a given test case |
| **Test group** | a collection of test cases which share at least one defined criterium. E.g. all test cases which relate to cybersecurity testing might be defined to make up a test group |
| **Test method** | a general definition of testing procedures and test environment for a test plan |
| **Test plan** | a strategy or list of tasks used to verify that a product conforms to design specifications and product requirements |
| **Test preparation** | a definition of steps which are needed to prepare a test environment for test execution |
| **Testing procedures** | a specific list of steps which are needed to carry out a test case |
| **Test protocol** | a summary of the test results of all test cases defined in a test plan. It may also contain the test analysis for said test cases |
| **Test requirements** | a definition stating the status of the test environment which is needed for carrying out a specific test case or a test group. Ideally, it is also stated how it can be checked if the test environment is ready for test execution |
| **Test responsibilities** | a definition stating which persons or organizations are needed for the test. It may also include an assignment of tasks to those persons or organizations |
| **Test result** | an indication of whether a specific test case has passed or failed. May also include any data that has been obtained through execution of the test case |

## 2.2 TEST PLAN

The test plan used for testing this application consists of the following steps:

1) Review the project requirements and use cases
2) Define the features to be tested from those and classify them into test groups
3) Detail test cases for validation of named features
4) Execute the test cases
5) Document the test protocols

## 2.3 FEATURES TO BE TESTED

The partners have defined a number of features to be tested. Those features are based on the project requirements defined by the consortium and use cases for the WiseGRID project.

The features to be tested were classified into different test groups which are defined in Table 4. The table also defines which criteria are shared by the test cases within the test groups.

**Table 4 – Test groups**

| Test group | Common Criteria |
|---|---|
| **Visualisation and analysis** | The feature under test provides visualization and/or analysis of the data collected |
| **Control** | The feature under test provides control of assets. |

| | |
|---|---|
| **Compliance** | The feature under test relates to compliance of the tool with the USEF standard or other standards. |
| **Functionality** | The feature under test is a complex function provided by a combination of software and communication between multiple WiseGRID subsystems. |
| **Communication** | The feature under test is basic data transmission between two communication endpoints, one being the tool. |
| **Robustness and stability** | The feature under test is related to fault tolerant and stability. |
| **Cyber Security** | The feature under test mitigates vulnerabilities of the software or malicious attacks aimed at it. |

Table 5 shows the features tested as defined for testing the WiseCOOP and WiseCORP. Each feature shall define one test case.

**Table 5 – WG Cockpit's features tested**

| Test case code | Feature to be tested | Test group |
|---|---|---|
| **GPA001** | Static data needed for the simulation (buses, lines, adjacency matrix) can be extracted from topology model | Communication |
| **GPA002** | Excel file required to simulate the scenario can be successfully created from the UI | Functionality |
| **GPA003** | GAMS solver successfully loads the input file (.xlsx format). | Functionality |
| **GPA004** | GAMS solver successfully runs, produces results and delivers them to internal ESB. | Functionality |
| **GPA005** | Response of the GAMS simulation is properly shown in the UI | Visualisation and analysis |
| **TRP001** | Required data from structural database is collected and request by ESB is published. | Communication |
| **TRP002** | Required data from long-term database are collected and request by ESB is published. | Communication |
| **TRP003** | Module and submodule successfully receive requests and the corresponding input files. | Communication |
| **TRP004** | Module successfully runs, produces results and delivers them to internal ESB. | Functionality |
| **RTM001** | Data from SMX is properly collected in the operational database of WiseGRID Cockpit | Communication |
| **RTM002** | Data from SMX is properly collected in the long-term database of WiseGRID Cockpit (big data) | Communication |
| **KPI001** | Smart meters provide information of the total accumulated energy demand/production. The system therefore needs to calculate the energy deltas across consecutive readings in order to properly monitor the energy demand/production profiles.<br>Three different aggregation of the deltas are considered: quarterly, hourly and daily. | Communication, Visualisation and analysis |
| **KPI002** | Smart meters linked to buses of the grid with big RES installations provide information of the total production of the RES installation. There is a need to aggregate all of those in order to get an overview of the total energy production in the | Communication |

| | | |
|---|---|---|
| | grid | |
| **KPI003** | Smart meters linked to buses of the grid provide information of the total demand of the loads connected to them. There is a need to aggregate all of those in order to get an overview of the total energy demand in the grid | Communication |
| **KPI004** | The module enables the early detection of significant deviations of voltage on a given bus. This is performed by automatically evaluating a control chart on the voltage magnitude of each phase per bus. Alerts are triggered whenever percentiles 1 or 99 are surpassed | Functionality |
| **KPI005** | The module enables the early detection of significant deviations of frequency on a given bus. This is performed by automatically evaluating a control chart on the frequency magnitude per bus. Alerts are triggered whenever percentiles 1 or 99 are surpassed | Functionality |
| **PFM001** | Static data needed for the power flow calculation (buses, lines, adjacency matrix) can be extracted from topology model | Communication |
| **PFM002** | Real-time measurements required as an input for power flow are available in the long-term database | Visualisation and analysis |
| **PFM003** | Power flow calculator receives input data file (.xlsx format) and algorithm is executed. | Functionality |
| **PFM004** | Server delivers to internal ESB the output file (.xlsx format) with the power flow results. | Functionality |
| **PFM005** | Results of the 3PPF module are published to the IOP in the form of virtual smart meters, becoming therefore available to the any other module of the WiseGRID Cockpit requiring them. | Visualisation and analysis |
| **SE001** | Static data needed for the state estimation calculation (buses, lines, adjacency matrix) can be extracted from topology model | Communication |
| **SE002** | Historic measurements required as an input for state estimation calculation are available in the long-term database | Visualisation and analysis |
| **SE003** | State estimator receives input data file and algorithm is executed. | Functionality |
| **SE004** | Server delivers to internal ESB the output file (.xlsx format) with the power flow results. | Functionality |
| **SE005** | Results of the state estimation module are published to the IOP in the form of virtual smart meters, becoming therefore available to the any other module of the WiseGRID Cockpit requiring them. | Communication |
| **FOR001** | WG Cockpit forecast module is trained | Functionality |
| **FOR002** | WG Cockpit forecast module performs demand/production forecasting training | Functionality |
| **FOR003** | Performance of WG Cockpit forecast module, at parsing forecast queries. | Functionality |
| **FOR004** | Performance of WG Cockpit forecast module, at generating and submitting the forecast response. | Functionality |
| **FOR005** | WiseGRID Cockpit periodically posts a demand and a production forecast request per bus to the corresponding queue of the internal ESB | Communication |

| | | |
|---|---|---|
| **FOR006** | WiseGRID Cockpit receives the results of the forecast module, formats them following the same format used to store real-time data, and stores the in the operational database | Communication, Visualisation and analysis |
| **CF001** | WiseGRID Cockpit periodically posts the necessary inputs to the corresponding topics of the internal ESB (MQTT, Congestion_Forecast/+ topics) | Communication |
| **TM001** | Data published to the IOP (MQTT protocol) can be collected by the module in real-time | Communication |
| **TM002** | Data published to the IOP (MQTT protocol) can be collected by the module in real-time | Communication |
| **TM003** | Data published to the IOP (MQTT protocol) can be collected by the module in real-time | Communication |
| **OD001** | Data published to the IOP (MQTT protocol) can be collected by the module in real-time | Communication |
| **OD002** | Module detects outliers on measured voltage | Functionality |
| **OD003** | Module detects outliers on measured frequency | Functionality |
| **PQM001** | Measurement data required for the calculation of the power quality of a node are extracted from the long-term database by the internal ESB. | Functionality |
| **PQM002** | Power quality module receives input data file and algorithm is executed. | Communication, Functionality |
| **PQM003** | Server delivers to internal ESB the output file (.xlsx format) with the power quality results. | Communication, Visualisation and analysis |
| **EMS001** | Required data from structural database and forecast module are collected and request by ESB is published. | Communication |
| **EMS002** | EMS for islands module receives input data file and algorithm is executed. | Communication, Functionality |
| **EMS003** | Server delivers to internal ESB the output file with the results. | Communication |
| **GFM001** | New workflows can be defined from the UI | Functionality, Visualisation and analysis |
| **GFM002** | Defined incident workflows are executed when the configured incident is detected | Functionality |
| **GFM003** | Custom defined workflows have the ability to command the Market Hub module in order to initiate new demand-response campaigns | Functionality |
| **FLI001** | Ability to identify the location of a fault event in the network lines | Functionality |
| **FLI002** | Ability to reconfigure network after a fault event in the network to restore maximum power load | Functionality |
| **FLI003** | Ability to trigger the FLISR module upon detection of a change on a switch status and retrieve the results of the FLISR module | Control |

## 2.4 TEST CASES SPECIFICATION

All test cases are specified in the following subsection using a template sheet as shown in Table 6.

Table 6 – Test case specification template

| | | | |
|---|---|---|---|
| **Name** | *The test case code and name which is unique to the project.* | | |
| **Module under test** | *The devices or systems under test* | **Resp.** | *Main partner responsible for the test* |
| **Module requirement** | *The requirement, use case, or certification rule which is validated by the test case* | | |
| **Test environment** | *List of elements needed for the test execution* | | |
| **Features to be tested** | *List of features to be tested* | | |
| **Features not to be tested** | *Optional* | | |
| **Preparation** | *Short list of steps needed for preparing the test environment for test execution* | | |
| **Dependencies** | *(Optional) List of test case codes defining test cases which need to be passed before the test case at hand can be started* | | |
| **Steps** | *Testing procedures* | | |
| **Pass criteria** | *Expected (measurable) results, allowing to unambiguously judge if the test is passed or not passed (i.e. the product requirement was validated or not validated)* | | |
| **Suspension criteria** | *(Optional) Conditions under which continuation of the test is considered pointless because testing results would be invalid* | | |
| **Results** | *(Optional) Short list of results* | | |

## 2.5 LAB-TESTING PLATFORM DETAILS

The implementation and lab-testing phases of the development of WiseGRID Cockpit has been supported by the setup of a virtual environment which replicates up to the possible extent the conditions that will be found in the deployment of the applications in the different pilot sites.

The lab-testing platform consists of a couple of virtual machines running in the VMWare vSphere infrastructure of ETRA I+D.

Table 7 – Characteristics of the lab-testing platform servers

| Characteristics | wisegridpre.lab.id | wintest.lab.id |
|---|---|---|
| **OS** | Ubuntu Server 16.04 | Microsoft Windows Server 2012 |
| **CPU** | 2 CPU | 1 CPU |
| **Memory** | 8GB | 5GB |
| **Hard disk** | 50GB | 35GB |

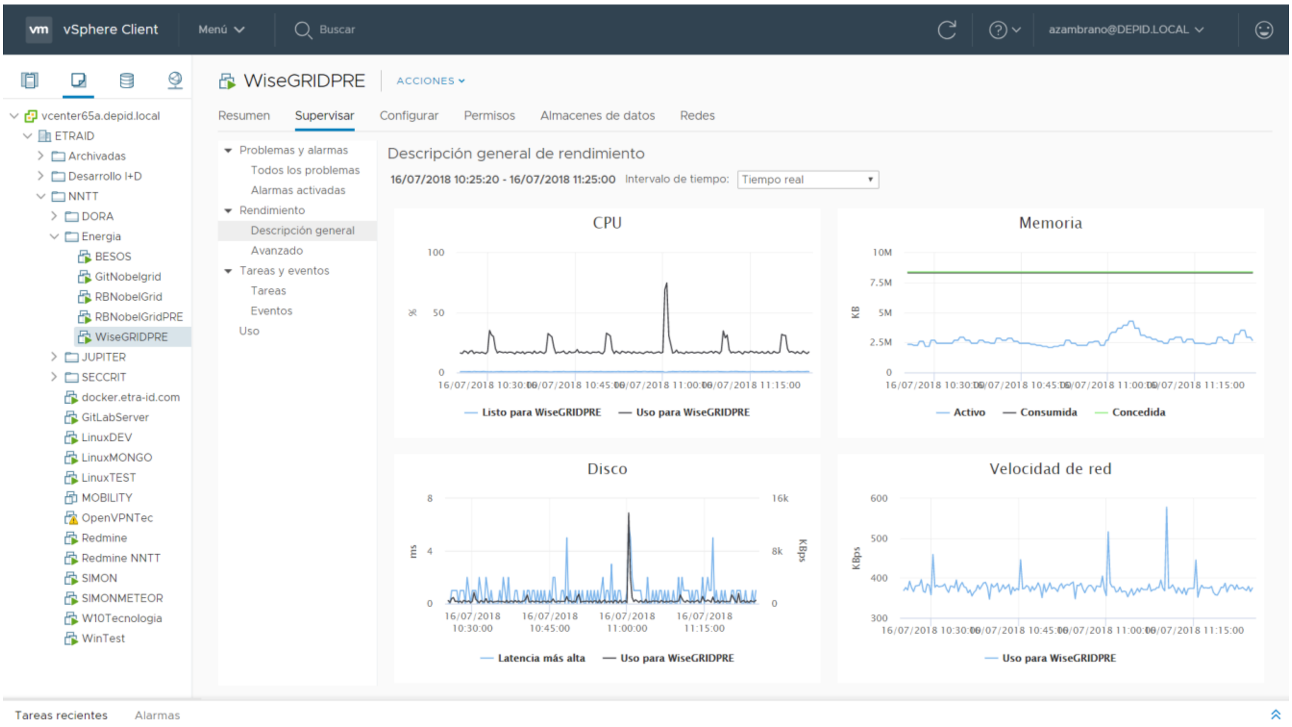



**Figure 1 – Screenshot of wisegridpre.lab.id server on vSphere platform**

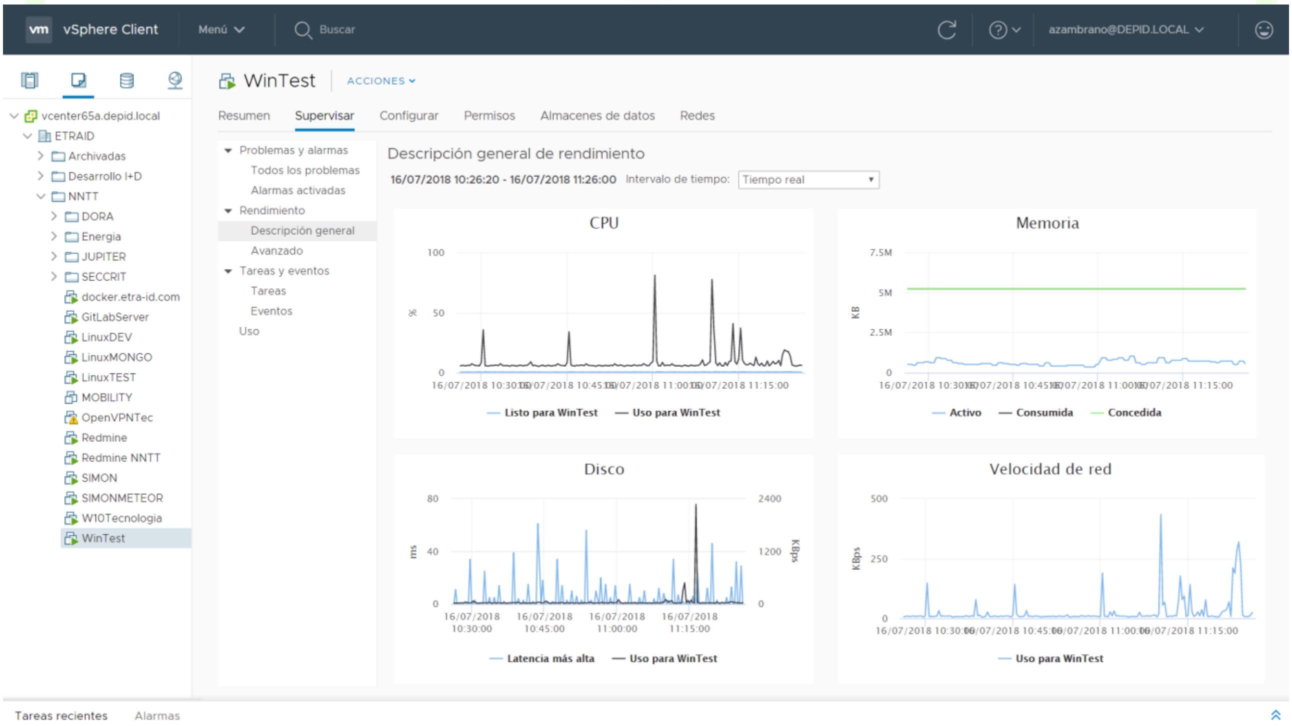



**Figure 2 – Screenshot of wisegridpre.lab.id server on vSphere platform**

Following the architecture of the tools, a number of common services have been installed in those servers and made accessible to the rest of the partners taking part in the project via public URL (protected with the corresponding access credentials). Data protection principles have been also considered in line with deliverable D3.2.

Table 8 – Server directions

| Server | Module | URL |
|---|---|---|
| **wisegridpre.lab.id** | Internal ESB (RabbitMQ).<br>One virtual hosts (wgcockpit) | AMQP: amqp://etra-id.com<br>MQTT: tcp://etra-id.com:1883 |
| | Database server (MongoDB) | mongodb://etra-id.com |
| | Big data processing (Spark Server – 1 master + 1 server) | [internal access only] |
| | WiseGRID Cockpit User Interface | https://wgcockpit.etra-id.com |
| **wintest.lab.id** | Maintenance management module | http://windeptec.etra-id.com/GiManWiseGRIDWSRest |

The described configuration, together with the lab-testing instance of the WiseGRID IOP, allowed the different partners, in charge of the development and testing of specific modules within each one of the applications, to connect each one to the other parts and perform the necessary integration tests to make sure that all modules work together as expected. Particularly, the following points have been covered:

- Connection of real and simulated assets to the WiseGRID IOP, in order to test the data ingestion of the applications
- Intercommunication among the different modules, by connecting them all via Internet to the internal ESB of each application
- Hosting most of the modules composing the applications
- Testing of the KPI engine implementation on a local instance of Spark Server (in parallel to the development of the Big Data Platform)
- Testing technologies that will facilitate deployment of the modules in the pilot sites (Docker and Docker-compose)
- Access to preliminary versions of the User Interfaces of the applications

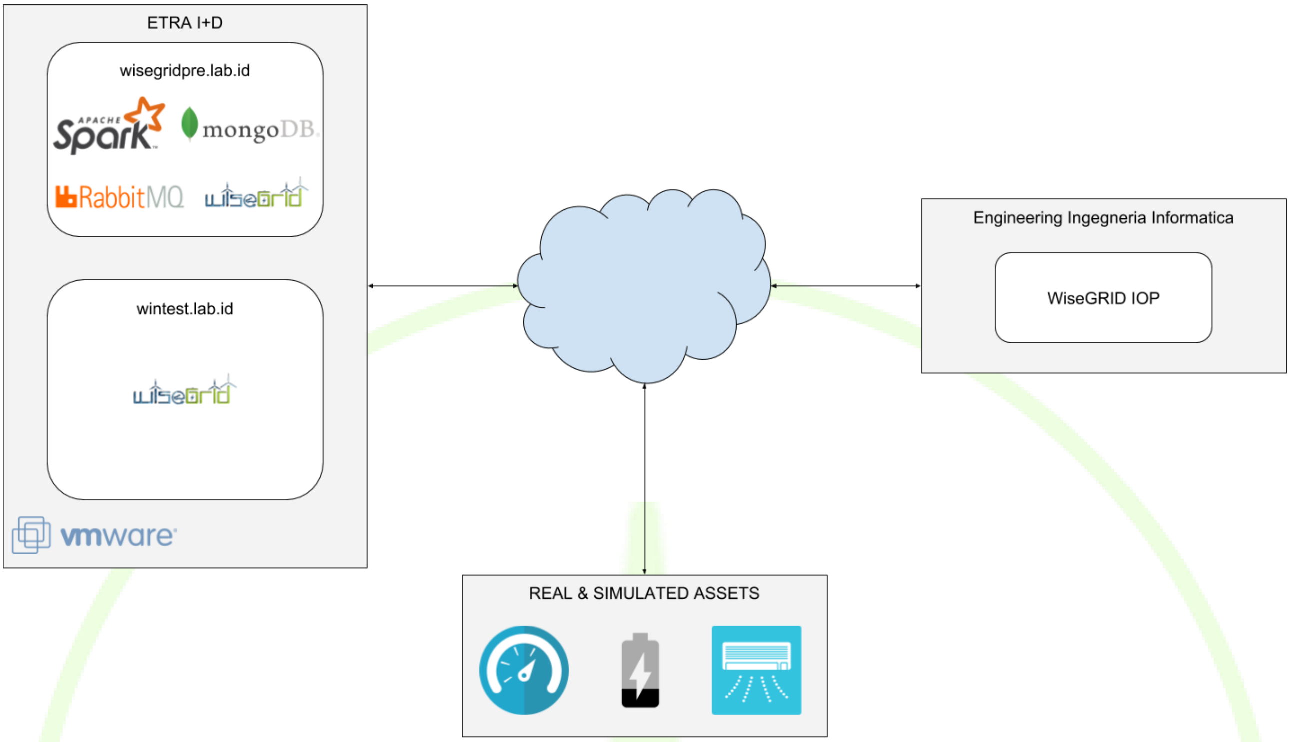



**Figure 3 – Lab-testing platform core modules overview**

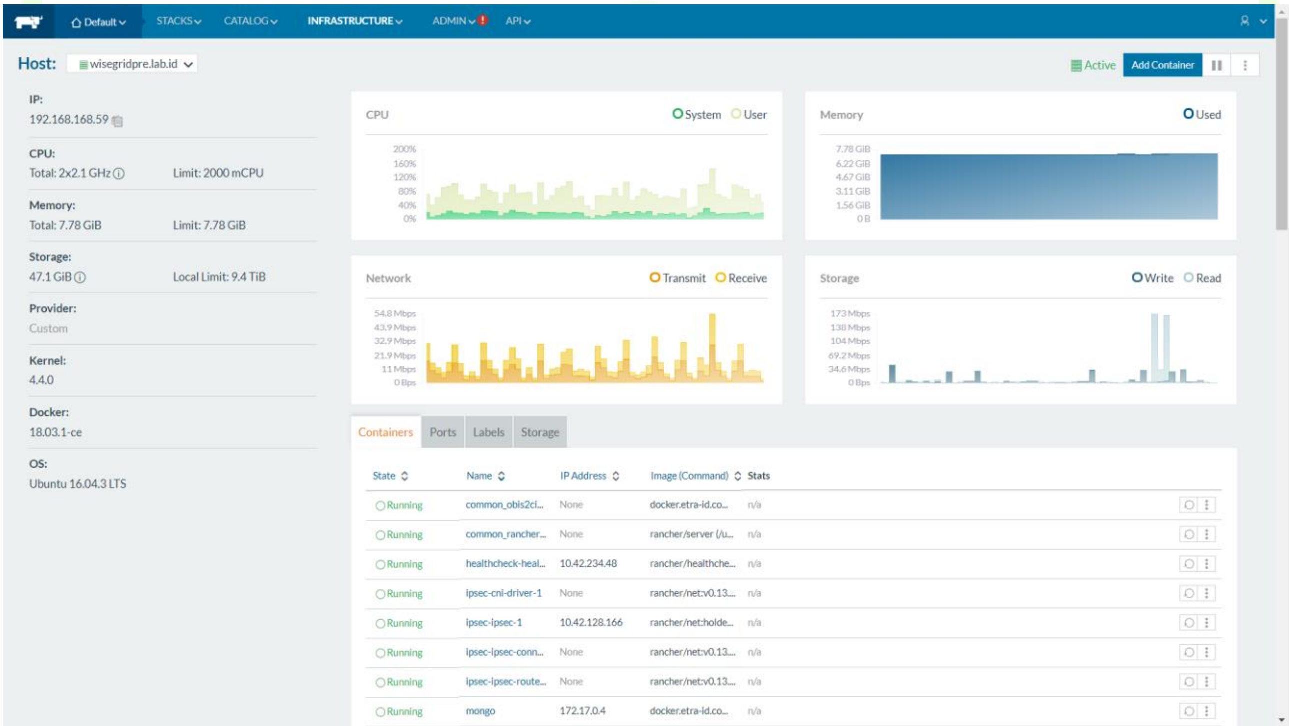



**Figure 4 – Dashboard of wisegridpre.lab.id, all modules installed as Docker containers**

## 3 IMPLEMENTATION

### 3.1 ARCHITECTURE OVERVIEW

The architecture of the application finally implemented does not differ significantly from the architecture presented in the previous deliverable D13.1 WiseGRID Cockpit Design [4]. The architecture is summarized in this section for completeness of this document.

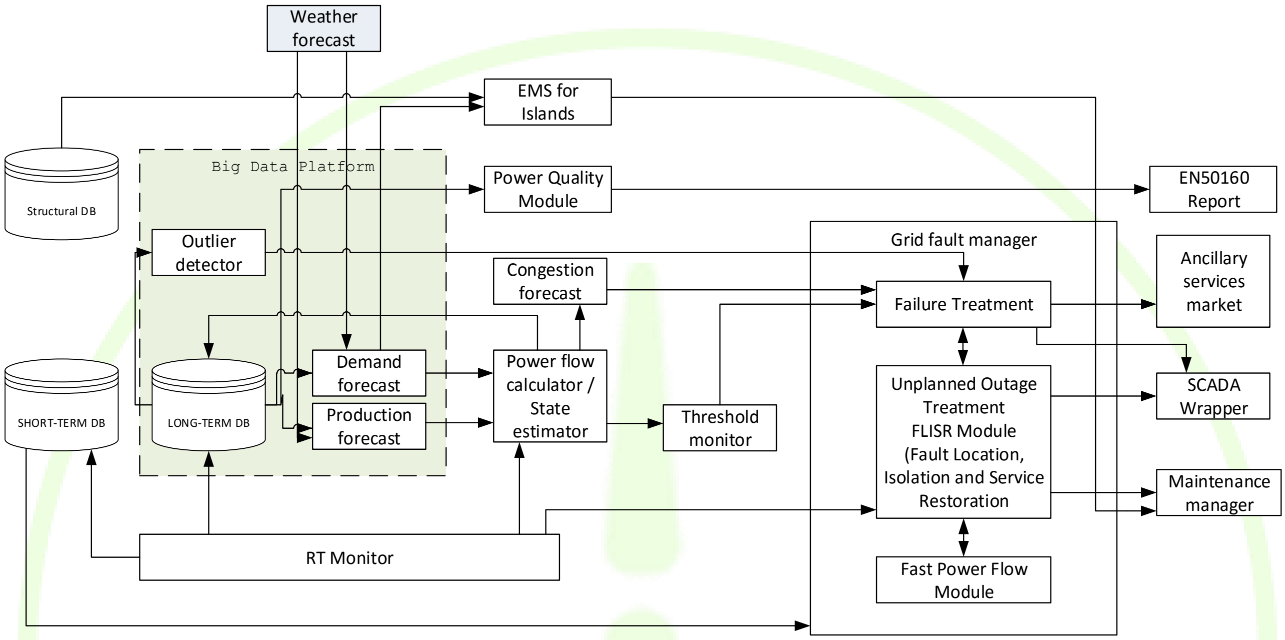



Figure 5 – Overview of the interaction among the main modules of the WiseGRID Cockpit application

**Offline processes**

A number of modules have been identified whose functionalities are needed either during the commissioning of the system, or which provide functionalities that do not rely in the treatment of data in real time. Those modules include:

- *Topology processor*, the software that translates the topological representation of the distribution grid and the technical characteristics of its elements into the Common Data Model that can be managed by WG Cockpit
- *GIS provider*, the software that translates GIS data from the original format managed by the DSO into the Common Data Model accepted by WG Cockpit
- *Grid planning assistant*, a module allowing the DSO to perform simulation of different scenarios and assess the possible issues met when considering different patterns for demand, production and flexibility in the distribution grid
- *Thermal and RES Planning*, a tool used for network planning, which analyses the capacity of an energy system in RES, and how this is influenced if thermal (classical) units are also considered. This tool provides investment decision support manager in the long-term period to the DSO

**Data ingestion**

The first step considered in the design of the real-time features of the application is the data ingestion. The procedure followed is common to other applications in the project, and implies the following steps:

1. Publication of data from *Wrappers* to the *WG IOP Message Broker*. Following the principle taken in the overall project, data sources publish data to the Interoperable Platform, allowing different application with the corresponding permissions to access to those data flows
2. Subscription to data flows of interest. In the case of the WiseGRID Cockpit, the main data flows considered are those available by the DSO to monitor the distribution grid. Within the project, 3 different elements are considered: Advanced Metering Infrastructure systems, SCADA systems and Unbundled Smart Meters (SMX) devices. All three data sources will provide energy readings and electrical measurements on different points of the grid. Necessary software adaptations will be developed to allow those elements to publish data that can be used by the WiseGRID Cockpit to monitor the grid. This subscription to these data flows is performed by the *RT monitor* module
3. Store data for further analysis. The *RT monitor* module is in charge of populating both the *Operation* and the *Long-term DB* for further analysis
4. Analysis to increase observability of the grid. In the particular case of the WiseGRID Cockpit, Power Flow and State Estimation algorithms will be executed periodically to get an overview of the whole distribution network based on the available monitoring points.

**Data analysis, operation and control**

Under this group, different modules have been defined in order to process the raw data coming from the different data sources in order to get the relevant information out of those. The objective of those can be either informative (obtain information of interest to the DSO operator), or action triggering (i.e. monitoring certain aspects of the grid and triggering specific workflows when certain situations are met). These modules include:

- *KPI engine* module, in charge of extracting different indicators and patterns from the raw data, focusing in energy demand of the prosumers connected to the distribution grid and the frequency and reaction time of the DSO operator upon the detection of certain issues in the grid.
- *Demand and production forecast* module, providing forecasts for the energy demand and production of the customers of the DSO (grid users).
- *Threshold monitor* module, analysing the real-time flow of data to trigger alerts whenever operational conditions are exceeded.
- *Outlier detector* module, analysing the real-time flow of data to trigger alerts when certain parameters measured in the grid deviate from their statistical trend.
- *Congestion forecast* module, periodically executing an analysis algorithm that estimates the probability of congestion issues in the scheduled operating period to come.
- *Failure treatment* module, allowing the DSO operator to define custom workflows that will be automatically executed whenever a certain preconfigured condition is met and identified by the other modules of the WG Cockpit.
- *Unplanned Outage Treatment/FLISR* module, which will automatically calculate the optimum operations to recover from an outage in the distribution grid to get the minimum possible impact, also considering the implementation of automatic restoration of the supply.
- *Maintenance Manager* module, assisting the DSO operator to properly perform the preventive and corrective maintenance of the elements of the grid.

**Interaction with other applications**

Within the WiseGRID project, a special focus is set in demonstrate how different kind of technologies can be beneficial to support the operation of the Smart Grid. These technologies will be addressed by different applications within the project, and will include electric vehicles, storage systems and other controllable demand assets. In this context, WiseGRID Cockpit, will interact with the different applications of the project addressing aggregators (WiseEVP, WG StaaS/VPP and WiseCOOP) through the *Ancillary Services Market*, in order to request support from those for assisting the correct operation of the distribution grid when required. Each application will evaluate up to which extent the corresponding technologies can assist the DSO by considering both the needs of the DSO, the business requirements of the aggregator and the particularities of the technologies and customers being aggregated.

**Horizontal and support functionalities**

Different modules will be used indirectly by the WiseGRID Cockpit application. Summarizing, these modules are data providers that offer information needed for other modules of the application to fulfil their duties, which are reused among different applications developed within the project. The list includes the *Weather Forecast* – whose information will assist the forecast modules - , *Energy mix provider* – whose data is used to assess the environmental impact of the used energy – and the *Big Data platform* that will support the long-term storage and analysis. Finally, the *WG Cockpit User Interface* is included in this category, providing web-based access to the information and functionalities provided by the other modules.

## 3.2 BACKOFFICE MODULES

### 3.2.1 Internal Enterprise Service Bus

As depicted in the architectural overview, the application is actually composed of several modules with well-defined functionalities, which collaborate with each other in order to enable the high-level functionalities of the application. In order to facilitate the communication among the modules, it was decided during the design phase to incorporate an internal Enterprise Service Bus to the application. The selected technology for deploying this communication bus has been RabbitMQ, since it covers most of the requirements settled during the design phase and exposed in D13.1.

RabbitMQ has been configured with the following main characteristics:

- Credential-based access control: one credential has been given to each partner requiring access.
- Protocols enabled: AMQP, MQTT and HTTP.
- Virtual hosts: a specific virtual host (/wgcockpit) has been configured to partition the communication flows of the modules of this application

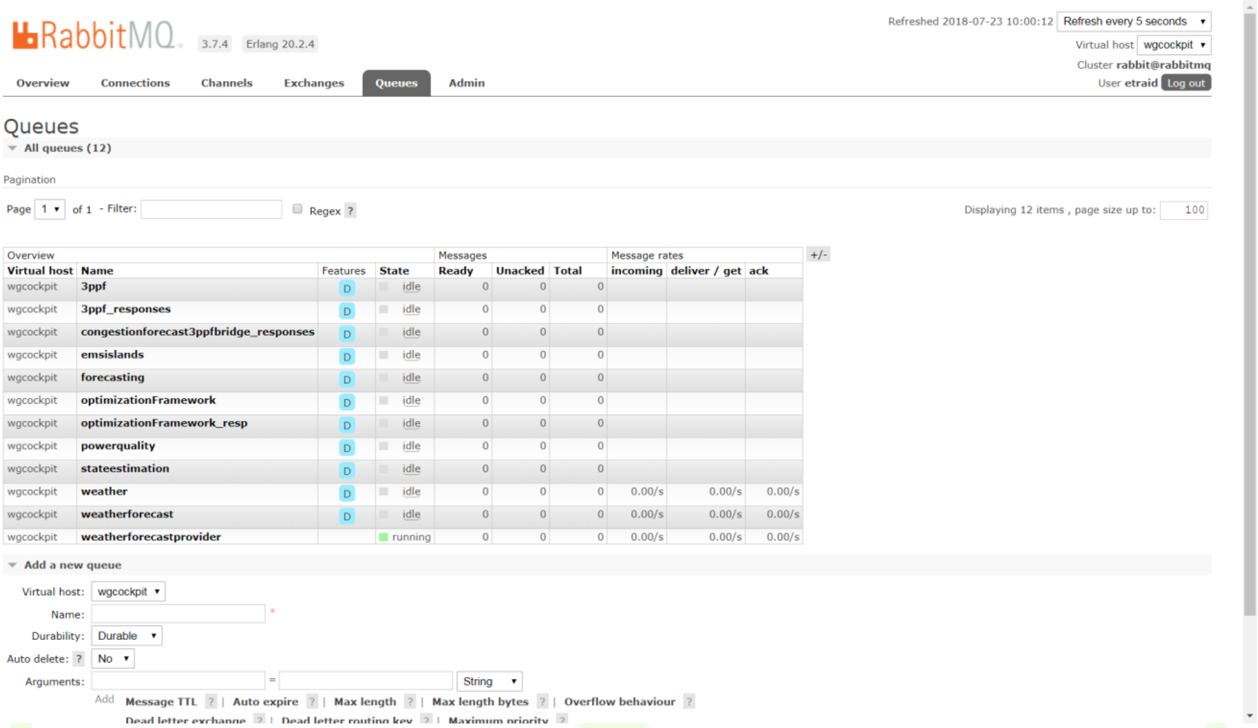



Figure 6 – Queues created in the internal ESB for exchange of information among modules of WiseGRID Cockpit

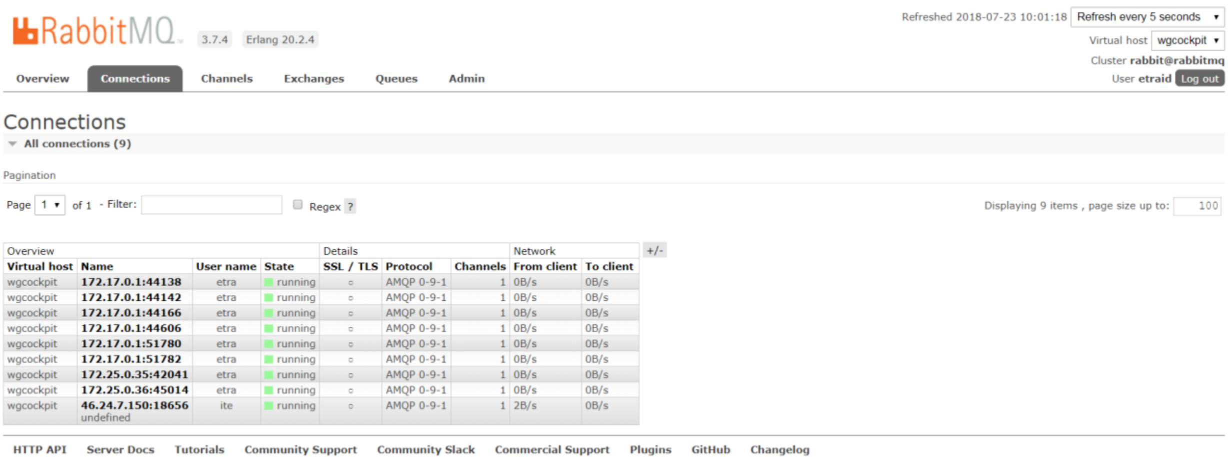



Figure 7 – List of actives connections (9) to the internal ESB of WiseGRID Cockpit

### 3.2.2 RT –monitor

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

Particularly for WG Cockpit, this module is in charge of tracking and storing in the databases of WG Cockpit the data items shown in the following table.

Table 9 – Data items tracked by Real-time monitor in WiseGRID Cockpit application

| Data item | Source | Operational DB | Long-term DB |
|---|---|---|---|
| **Energy readings** | Field assets (SMX, AMI systems, SCADA) | X | X |

| | | | |
|---|---|---|---|
| **Weather** | Weather forecast provider | X | X |
| **Weather forecast** | Weather forecast provider | X | X |
| **Energy mix** | ENTSOE energy mix provider | X | X |
| **Energy mix forecast** | ENTSOE energy mix provider | X | X |
| **Incidents** | Congestion forecast module<br>Threshold monitor<br>Outlier detector | | |

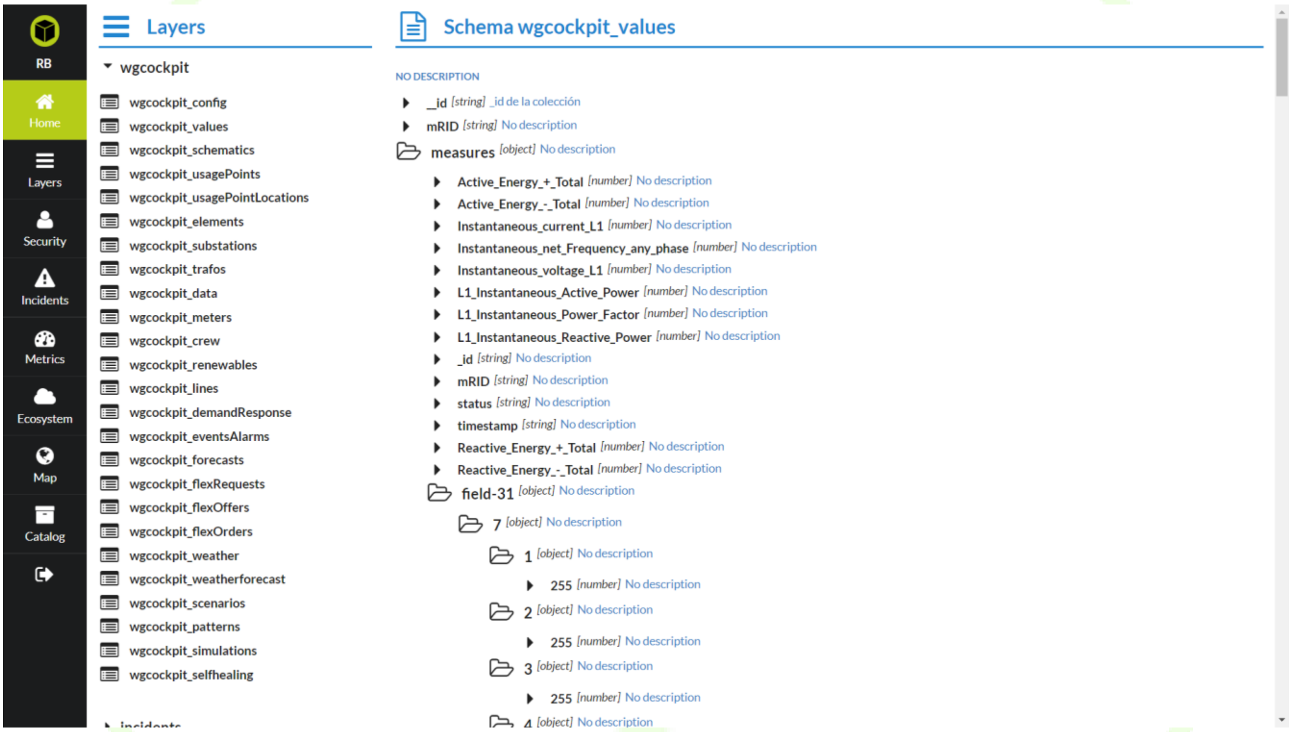



**Figure 8 – Screenshot of the Real-time monitor UI, showing energy readings schema**

### 3.2.3 KPI engine

The KPI engine of WiseGRID Cockpit application has been implemented as a set of Spark jobs that are periodically triggered on the long-term database to perform the necessary calculations and push the results back to different collections of the database.

**Table 10 – WiseGRID Cockpit – Spark jobs of the KPI engine**

| Spark job | Module | Description | Result KPIs |
|---|---|---|---|
| **WGCockpit summaryCalculation 15m** | summarycalculator-assembly-0.2.jar | Calculates aggregated registers for every meter and every 15 minutes | Energy demand<br>Energy production<br>Equivalent $CO_2$ |

| | | | |
|---|---|---|---|
| | | | emissions<br>Average current<br>Average voltage<br>Average Active Power<br>Average Reactive Power |
| **WGCockpit summaryCalculation 1h** | summarycalculator-assembly-0.2.jar | Calculates aggregated registers for every meter and every hour | Energy demand<br>Energy production<br>Equivalent $CO_2$ emissions<br>Average current<br>Average voltage<br>Average Active Power<br>Average Reactive Power |
| **WGCockpit summaryCalculation 1d** | summarycalculator-assembly-0.2.jar | Calculates aggregated registers for every meter and every day | Energy demand<br>Energy production<br>Equivalent $CO_2$ emissions<br>Average current<br>Average voltage<br>Average Active Power<br>Average Reactive Power |

Spark 2.3.0 Spark Master at spark://wisegridpre.lab.id:7077

**URL:** spark://wisegridpre.lab.id:7077
**REST URL:** spark://wisegridpre.lab.id:6066 *(cluster mode)*
**Alive Workers:** 1
**Cores in use:** 1 Total, 1 Used
**Memory in use:** 1024.0 MB Total, 1024.0 MB Used
**Applications:** 1 Running, 1 Completed
**Drivers:** 0 Running, 0 Completed
**Status:** ALIVE

**Workers (1)**

| Worker Id | Address | State | Cores | Memory |
|---|---|---|---|---|
| worker-20180723082940-172.17.0.4-8881 | 172.17.0.4:8881 | ALIVE | 1 (1 Used) | 1024.0 MB (1024.0 MB Used) |

**Running Applications (1)**

| Application ID | | Name | Cores | Memory per Executor | Submitted Time | User | State | Duration |
|---|---|---|---|---|---|---|---|---|
| app-20180723092319-0001 | (kill) | WGCockpit_summaryCalculation_1h | 1 | 1024.0 MB | 2018/07/23 09:23:19 | etraid | RUNNING | 2.1 min |

**Completed Applications (1)**

| Application ID | Name | Cores | Memory per Executor | Submitted Time | User | State | Duration |
|---|---|---|---|---|---|---|---|
| app-20180723083005-0000 | WGCockpit_summaryCalculation_15m | 1 | 1024.0 MB | 2018/07/23 08:30:05 | etraid | FINISHED | 14 min |

**Figure 9 – Screenshot of Spark server with executed WiseGRID Cockpit jobs**

### 3.2.4 Power Quality

The Power Quality module gives the WG Cockpit the capability to monitor and evaluate the power quality at important nodes of the system. With this functionality the operator can identify whether or not the provided power is at the quality standard levels specified by the standard EN 50160.

Using the WG Cockpit interface the user can specify the node to monitor the power quality, provided that the node is measured, and configure appropriately the input parameters of the algorithm. The WG Cockpit, through the Internal ESB, fills the input file of the module (which is in .xlsx. format) with the data from the User Input and the long-term database (where the requested measurements are stored). At the following figures the input for the evaluation of power quality up to 5$^{th}$ harmonic with samples of 10min is presented. This input is used as an example and is not based on real measurements. Note that for the case of WG Cockpit the First Option (RMS values given) of the algorithm is used.

| f (Hz) | Number of harmonics | Option | First Option: RMS values given | | Second Option: Waveform construction | | Voltage dips classification |
|---|---|---|---|---|---|---|---|
| | | | Total examination time (days) | Sample rate (min) | Total examination time (min) | Sample rate (ms) | Number of voltage dips |
| 50 | 5 | 1 | | | | | |
| | | | 7 | 10 | 10 | 0,1 | 75 |

**Figure 10 – User data from WG Cockpit interface for the Power Quality module.**

| Samples | Harmonic order | Voltage | | | | | | Current | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Phase a | | Phase b | | Phase c | | Phase a | | Phase b | | Phase c | |
| | | Vrms (V) | Angle (°) | Vrms (V) | Angle (°) | Vrms (V) | Angle (°) | Irms (A) | Angle (°) | Irms (A) | Angle (°) | Irms (A) | Angle (°) |
| 1 | 1 | 230 | 0 | 221 | -121 | 233 | 121 | 20 | 0 | 21 | -121 | 21 | 122 |
| | 2 | 34 | 1 | 33 | -118 | 32 | 119 | 9 | 1 | 10 | -116 | 9 | 119 |
| | 3 | 25 | 2 | 21 | -105 | 20 | 120 | 5 | 2 | 7 | -105 | 7 | 120 |
| | 4 | 13 | 5 | 11 | -120 | 11 | 110 | 5 | 2 | 6 | -120 | 7 | 111 |
| | 5 | 5 | 1 | 5 | -118 | 6 | 115 | 3 | 1 | 6 | -117 | 6 | 115 |
| | | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| 2 | 1 | 230 | 0 | 219 | -122 | 233 | 122 | 21 | 0 | 22 | -121 | 20 | 121 |
| | 2 | 34 | 1 | 35 | -118 | 35 | 119 | 9 | 1 | 10 | -116 | 9 | 119 |
| | 3 | 25 | 3 | 21 | -105 | 20 | 120 | 5 | 3 | 7 | -105 | 7 | 120 |
| | 4 | 13 | 5 | 11 | -120 | 11 | 110 | 5 | 2 | 6 | -120 | 6 | 111 |
| | 5 | 5 | 1 | 5 | -118 | 6 | 115 | 3 | 1 | 6 | -117 | 6 | 115 |

**Figure 11 – Measurement data coming from the long-term DB for the Power Quality module.**

Based on this input the module assesses the power quality of the module based on the criteria specified by EN50160. The results are stored at an output file (.xlsx format) and through the internal ESB of the WG Cockpit will be then presented to the user at the WG Cockpit User Interface. Currently, the interface for the

Power Quality is not implemented. At the following figures, the results as stored at the output file are presented.

| Harmonics | | | | |
|---|---|---|---|---|
| **Power Quality Indices** | **Value (95% of 10 min rms) (%)** | **Limits according to EN 50160 (%)** | **Difference %** | **Error** |
| **Relative Harmonic Voltage** | | | | |
| U2 | 1,7 | 2 | -15,00 | NO |
| U3 | 5,4 | 5 | 8,00 | YES |
| U4 | 0,8 | 1 | -20,00 | NO |
| U5 | 7,1 | 6 | 18,33 | YES |
| **THD** | 6,4 | 8 | -20,00 | NO |

| Unbalance | | | | |
|---|---|---|---|---|
| **Power Quality Index** | **Value (95% of 10 min rms) (%)** | **Limits according to EN 50160 (%)** | **Difference %** | **Error** |
| **VUF** | 1,2 | 2 | -40,00 | NO |

| Voltage Dips | | | | | |
|---|---|---|---|---|---|
| u(%) / t(ms) | 10≤t≤200 | 200≤t≤500 | 500≤t≤1000 | 1000≤t≤5000 | 5000≤t≤60000 |
| 90>u≥80 | 8 | 7 | 7 | 5 | 6 |
| 80>u≥70 | 9 | 9 | 2 | 2 | 3 |
| 70>u≥40 | 6 | 5 | 5 | 3 | 1 |
| 40>u≥5 | 3 | 3 | 3 | 1 | 1 |
| 5>u | 5 | 1 | 1 | 0 | 0 |

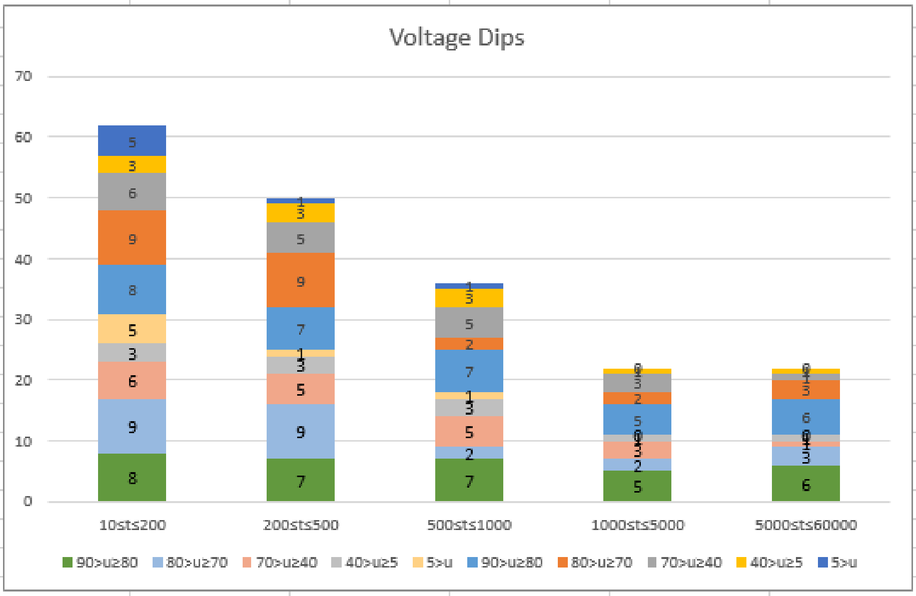



**Figure 12 – Output results of the Power Quality module.**

### 3.2.5 Grid planning assistant

The main purpose of the Grid Planning Assistant is allowing the distribution operator to simulate *what if* scenarios, in order to assess the effect of different load, generation and flexibility patterns to the grid. This module integrates the results of the tasks T4.1 and T4.2 within the WiseGRID Cockpit application.

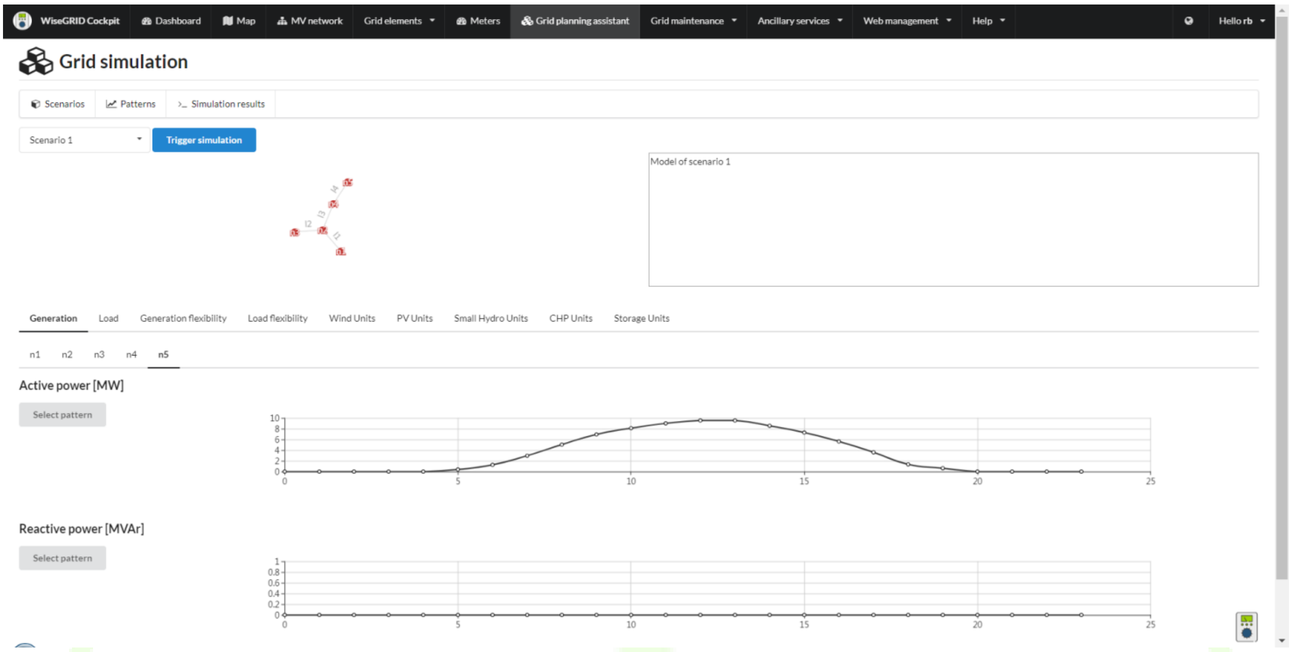



Figure 13 – Grid Planning Assistant

A complete section in the user interface of WiseGRID Cockpit has been developed in order to allow the DSO operator to include all necessary information to define the scenario, trigger the simulation and display the results.

### 3.2.6 Energy Management System for Non-Interconnected Systems

This module's main purpose is the Unit Commitment and Economic Dispatch at non-interconnected systems (islands) with increased RES penetration. It is a very useful tool for small DSO's that operate at electrical islands in liberalized market environments which facilitates the increase of RES penetration in the local system.

This module is completed and currently runs as a web-based tool. For the WG Cockpit implementation the Internal ESB will prepare the input file at .json format. The module will execute and return the results at the same format (.json).

A sample of the input file format is shown at the following tables. There, the wind and solar weather predictions and the corresponding PV and Wind power production are given at an hour time step. These values are necessary, in combination with the load forecast, so that the algorithm can estimate the remaining necessary production to cover the demand and ensure system stability and economic viability.

Table 11 – Solar prediction as an input to the module.


```
{
        "results": [
                {
                        "date": "2018-07-30 14:00",
                        "GHI": 961.7453948,
                        "DNI": 858.6731745,
                        "DHI": 160.5656053,
```

```
            "WindSpeed": 1.127685084,
            "WindDirection": 358.4624066,
            "Temperature": 32.21984253,
            "TotalCloud": 0,
            "Pressure": 1007.198438
        },
        {
            "date": "2018-07-30 15:00",
            "GHI": 900.5633237,
            "DNI": 854.1472134,
            "DHI": 151.3252534,
            "WindSpeed": 0.8129750646,
            "WindDirection": 29.53854616,
            "Temperature": 32.51503906,
            "TotalCloud": 0,
            "Pressure": 1006.980625
        },
```


**Table 12 – Prediction for PV production as an input to the module.**


```
{
    "message": "Latitude 37.9838 Longitude 23.7275 TimeZone Europe\/Athens",
    "results": [
        {
            "date": "2018-07-30 14:00",
            "DCPower": 3.171442685,
            "ACPower": 2.5
        },
        {
            "date": "2018-07-30 15:00",
            "DCPower": 2.997909468,
            "ACPower": 2.5
        },
```

Table 13 -– Wind prediction as an input to the module.


```
{
	"results": [
		{
			"date": "2018-07-30 19:00",
			"WindSpeed": 5.221492539,
			"WindDirection": 284.5533975,
			"Temperature": 27.12392578
		},
		{
			"date": "2018-07-30 20:00",
			"WindSpeed": 5.11254753,
			"WindDirection": 284.6719448,
			"Temperature": 27.05761108
		},
```


Table 14 – Prediction of wind production as an input to the module.


```
{
	"message": "Latitude 36.5666 Longitude 26.2993 TimeZone Europe\/Athens",
	"results": [
		{
			"date": "2018-07-30 19:00",
			"WindPower": 0.1856249036
		},
		{
			"date": "2018-07-30 20:00",
			"WindPower": 0.1735516427
		},
```


### 3.2.7 Load Demand and Peak Prediction

This module of the WG Cockpit consists in a RPC server which makes use of the ESG. In addition, this module makes use of the long-term database of the WG Cockpit, which is implemented over a MongoDB database. The RPC server of this module is permanently running to manage the received queries through the RabbitMQ queue enabled to make use of this forecasting service.

Within the message queries are specified the id of the supply point, and the period and horizon of the desired forecast.

Once the query is deserialized and parsed, the forecast module retrieves from the long-term database the necessary information to perform the forecast. To perform the forecast it is retrieved information related to the consumed energy, working calendar and weather information related to the queried installation, being this information available in the long-term database, as it is possible to appreciate in the next picture.

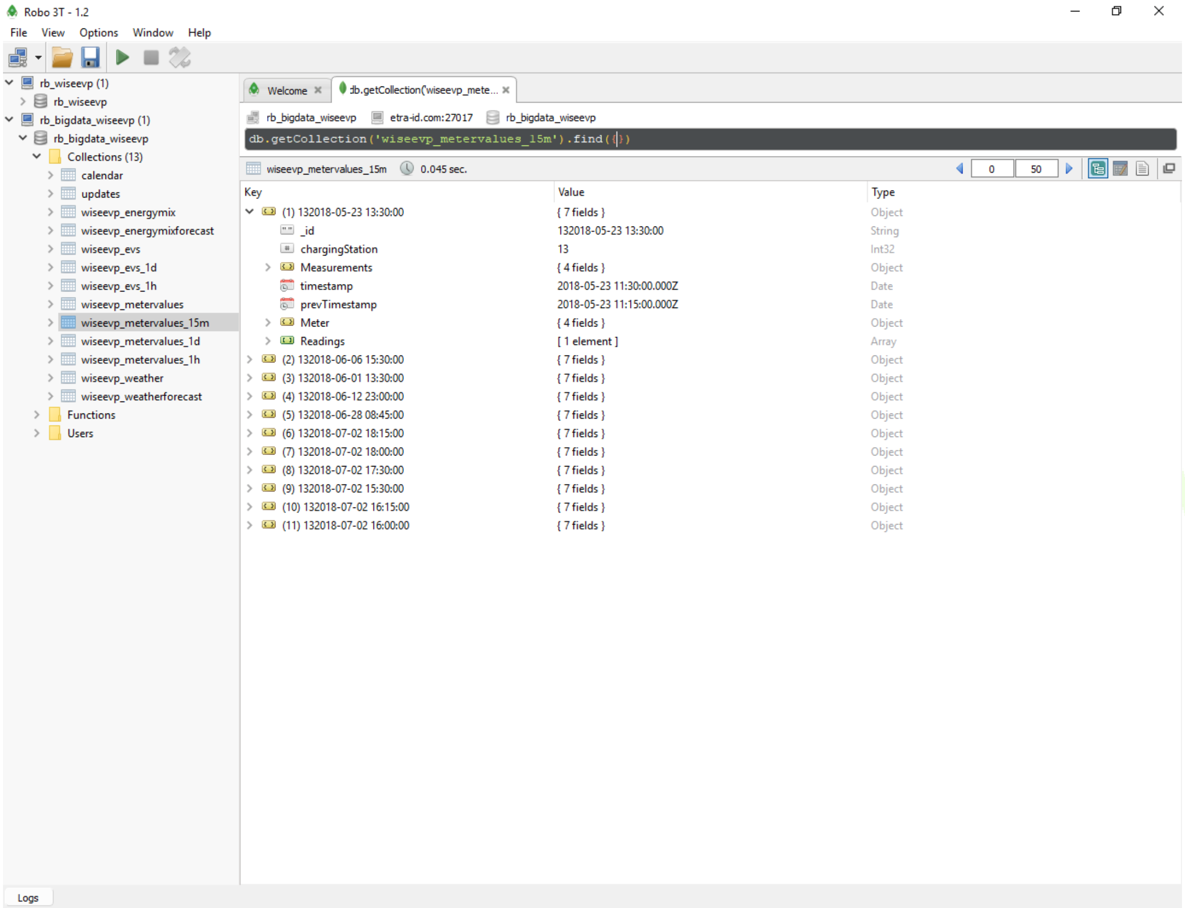



**Figure 14 – WG Cockpit long-term database screenshot**

Once the algorithm is run, the response provided by this, in which it is specified the demand forecast and the peaks this will have, is serialized and sent back through the corresponding RabbitMQ queue, providing the queried information.

### 3.2.8 Threshold monitor

The purpose of the threshold monitor is to provide alerts upon the detection of an excess of power line or deviation of voltage on the monitored buses of the distribution grid, for the DSO operator to take the corresponding actions.

The module has been developed as an akka Stream application [5], thus supporting by design horizontal scalability to cope with scenarios with a high number of sensors producing very big flows of data to be processed.

The module is subscribed to information published by sensors that are bound to buses, and virtual sensors providing information of the power flows in each one of the lines (output of the state estimation module described in section 3.2.10). Voltage values are compared with acceptable operational limits (defined as a percentage deviation over nominal voltage) and apparent power flows are compared against operational limits of the lines.

### 3.2.9 Outlier detector

The purpose of the outlier detector is to implement a *control chart* over the voltage and frequency magnitudes that are measured at different sections (nodes) of the distribution grid.

The module has been developed as an AKKA Stream application [5], thus supporting by design horizontal scalability to cope with scenarios with a high number of sensors producing very big flows of data to be processed.

The module is subscribed to information published by sensors that monitor the grid (SMXs and SCADA), and, upon reception of new data, keeps track of statistical metrics (percentiles 1, 50 and 99) over measured voltage and frequency, by using the t-digest algorithm [6]. With that information, a dynamic control chart is implemented, which allows to identify the following information:

- Normal value for voltage and frequency in each one of the buses
- Normal deviation from nominal frequency and voltage in each one of the buses, allowing the identification of buses where these deviations has become usual
- Percentile 1, thus triggering an alert whenever values below this percentile are detected
- Percentile 99, thus triggering an alert whenever values above this percentile are detected

**Table 15 – Extract of the outlier detector database, keeping track of percentiles for voltage and frequency**


```
{
    "_id" : "BBB5976|0.0.0.12.0.1.15.0.0.0.0.0.0.0.224.0.33.0",
    "timestamp" : ISODate("2018-02-23T07:02:21.461Z"),
    "q01" : 51.32,
    "q50" : 51.52,
    "q99" : 51.69,
    "digest" : …
}
{
    "_id" : "BBB5976|0.0.0.0.0.1.54.0.0.0.0.0.0.0.128.0.29.0",
    "timestamp" : ISODate("2018-02-23T07:02:21.464Z"),
    "q01" : 241.57,
    "q50" : 242.53,
    "q99" : 243.34,
    "digest" : …
}
```


### 3.2.10 Power flow calculator / State estimator

Power flow calculator and state estimator are both used for the calculation of the power flows, currents and voltages of the grid in order to assist the decision-making of the operator looking towards stable operating mode. Their main difference, from the operator point of view, lies on the fact that power flow calculator, which implements a three-phase power flow algorithm, is scenario based, meaning that the input data are not necessarily real measurements from the nodes of the grid but data defined by a specific scenario, whereas the state estimator is used for the evaluation of the actual state of the grid based on available real measurements, which are usually less than the size of the grid. For more information considering the algorithms see D12.2.

Both modules receive the data at an input file (.xlsx format) which are filled by the Internal ESB and the results are stored at an output file (.xlsx format) which are delivered to the Internal ESB and then presented to the WG Cockpit UI. The input files of the two modules have similar structure with the main difference being the fact that state estimation requires also real measurements from the measured nodes. Both algorithms have been evaluated on various benchmark and real networks. IEEE-13 bus and IEEE-123 bus are common benchmark networks on which both algorithms were evaluated. At the following figures the network under study is IEEE-13. Note that although the benchmark is the same the input data are different because the algorithms were evaluated at different scenarios that were selected to suit the requirements of the study.

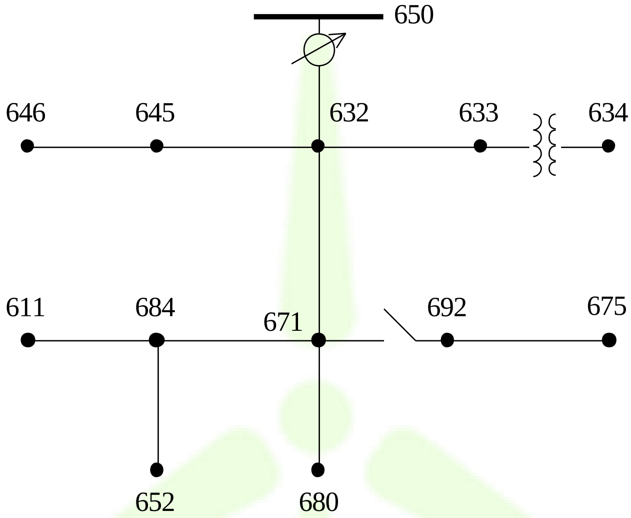



Figure 15 – IEEE - 13 bus benchmark network.

The following figure shows the bus data for the Three Phase power flow. Note that for this algorithm, specific data of all the nodes are required in order to run properly.

| bus i | ground_id | connection_id | Type | P_ph.A (kW) | P_ph.B (kW) | P_ph.C (kW) | Q_ph.A (kvar) | Q_ph.B (kvar) | Q_ph.C (kvar) | Vnom(kV) |
|---|---|---|---|---|---|---|---|---|---|---|
| 650 | 1 | 0 | 0 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 4,16 |
| 632 | 1 | 0 | 0 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 4,16 |
| 633 | 1 | 0 | 0 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 4,16 |
| 634 | 1 | 1 | 1 | 160,00 | 120,00 | 120,00 | 110,00 | 90,00 | 90,00 | 0,48 |
| 645 | 1 | 1 | 1 | 0,00 | 170,00 | 0,00 | 0,00 | 125,00 | 0,00 | 4,16 |
| 646 | 1 | 2 | 3 | 0,00 | 230,00 | 0,00 | 0,00 | 132,00 | 0,00 | 4,16 |
| 671 | 1 | 2 | 1 | 385,00 | 385,00 | 385,00 | 220,00 | 220,00 | 220,00 | 4,16 |
| 692 | 1 | 2 | 2 | 0,00 | 0,00 | 170,00 | 0,00 | 0,00 | 151,00 | 4,16 |
| 675 | 1 | 1 | 1 | 485,00 | 68,00 | 290,00 | 190,00 | 60,00 | 212,00 | 4,16 |
| 684 | 1 | 0 | 0 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 4,16 |
| 611 | 1 | 1 | 2 | 0,00 | 0,00 | 170,00 | 0,00 | 0,00 | 80,00 | 4,16 |
| 652 | 1 | 1 | 3 | 128,00 | 0,00 | 0,00 | 86,00 | 0,00 | 0,00 | 4,16 |
| 680 | 1 | 0 | 0 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 4,16 |

Figure 16 – Part of the input data (bus data) for the Power Flow Calculator.

| f_bus | t_bus | Phase A Active Power(kW) | Phase A Reactive Power (kvar) | Phase B Active Power(kW) | Phase B Reactive Power (kvar) | Phase C Active Power(kW) | Phase C Reactive Power (kvar) |
|---|---|---|---|---|---|---|---|
| 650 | 632 | 1217,030136 | 638,6956134 | 897,1524678 | 345,9496179 | 1185,40934 | 557,5407769 |
| 632 | 633 | 163,3012904 | 115,8666469 | 121,7278218 | 93,13659497 | 122,0680486 | 93,39906789 |
| 633 | 634 | 162,8955542 | 115,264644 | 121,5675111 | 92,85002027 | 121,7158361 | 93,11970198 |
| 632 | 645 | 0 | 0 | 318,3105939 | 128,3664635 | 69,61063256 | 122,6254224 |
| 645 | 646 | 0 | 0 | 145,7333036 | 1,157001763 | 69,43591429 | 122,4360466 |
| 632 | 671 | 1028,442663 | 437,4374092 | 462,7095853 | 94,07749355 | 956,5400252 | 274,925919 |
| 671 | 692 | 528,9775206 | 132,8581355 | 68,28087254 | -139,4864234 | 411,7921174 | 60,54261646 |
| 692 | 675 | 488,4777045 | 15,20815982 | 68,28087565 | -139,4864224 | 290,6552713 | 34,62117657 |
| 671 | 684 | 114,1881339 | 76,50166418 | 0 | 0 | 160,476221 | -12,4637076 |
| 684 | 611 | 0 | 0 | 0 | 0 | 160,1030854 | -12,72581041 |
| 684 | 652 | 114,0002753 | 76,30901667 | 0 | 0 | 0 | 0 |
| 671 | 680 | -0,000759547 | -0,007801497 | 0,001154643 | -0,007846963 | -0,000394759 | -0,006507668 |

**Figure 17 – Part of the output data of the Power Flow Calculator showing the power flows at the network branches.**

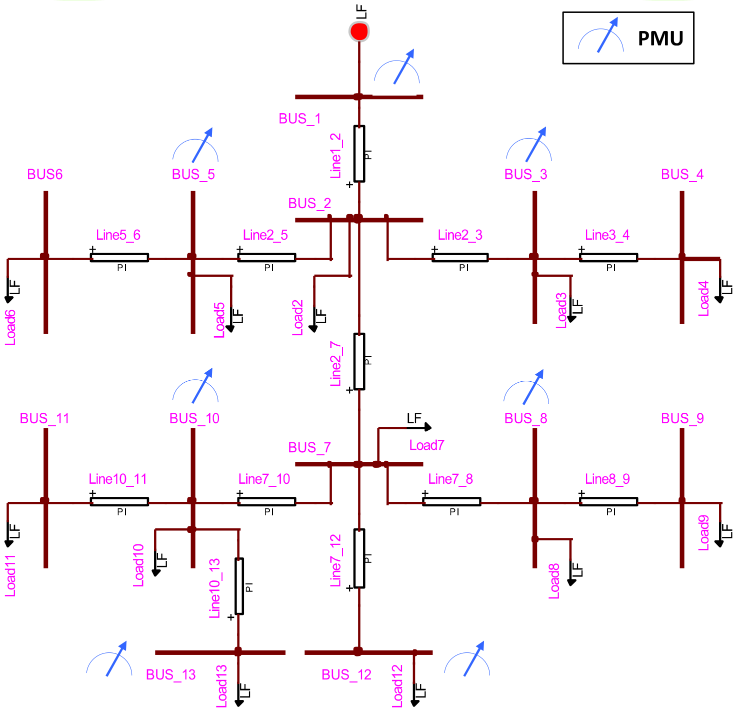



**Figure 18 – The IEEE-13 bus network used for the State Estimator with the measuring equipment (PMUs) installed at certain buses.**

For the evaluation of the state estimator algorithm, initially it was assumed that all the loads were known. With this as an input the real state of the grid was calculated using a three-phase power flow algorithm. Then using the calculated values for the monitored nodes (bus 5, bus 3 etc..) intentionally corrupted with some disturbance (in order to simulate a real measurement) the state of the network was estimated. The input for this benchmark is real measurements from the MV network of Switzerland. Also, in this case energy is injected from a PV and a small hydro.

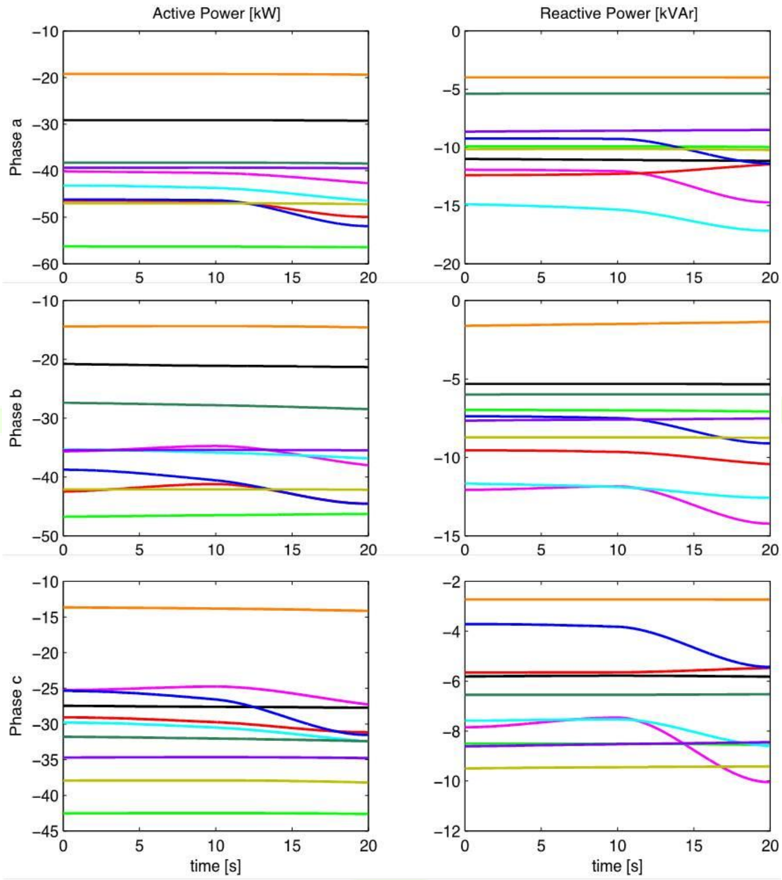



**Figure 19 – Load with respect to time for the evaluation of the state estimation algorithm.**

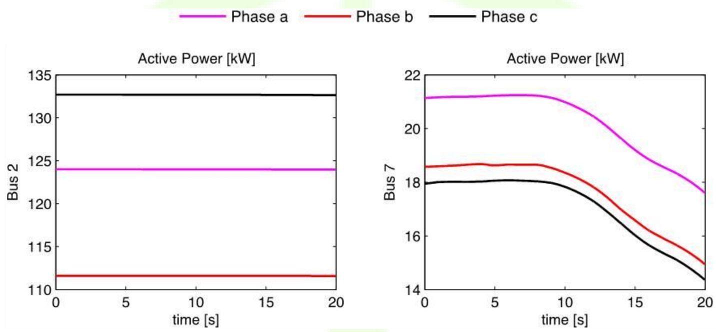
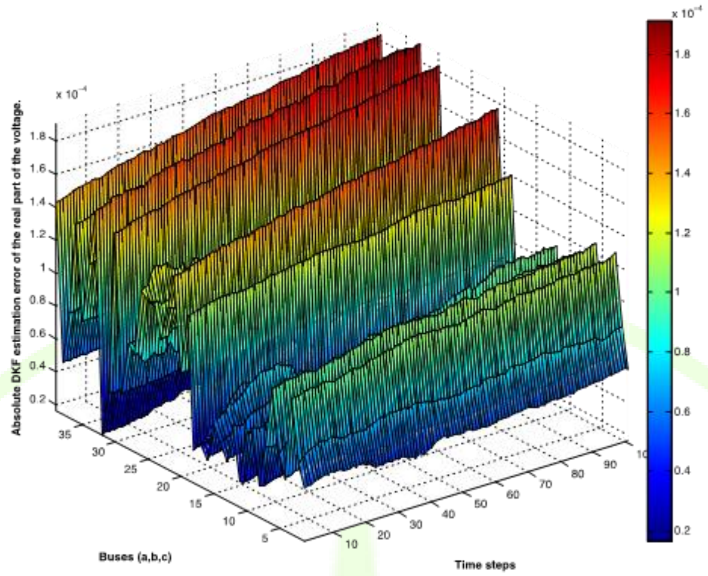



**Figure 20 – Injected power from PV (left) and small hydro (right).**



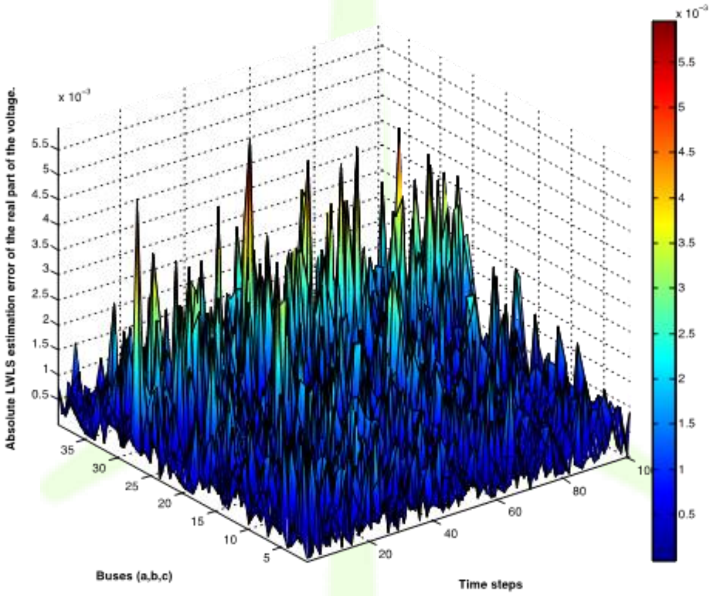




**Figure 21 – Absolute estimation errors of the real part of the voltage for two different methods (Discrete Kalman Filter – up – and Linear Weighted Least Squares – down).**

Using the errors, the real state of the network is estimated (power flows, currents and voltages at nodes). Since the output values are similar to the three-phase power flow a similar structure output file (.xlsx format) will be used.

### 3.2.11 Congestion forecast

For such an application the main principle is concerning the steady state calculations (power flow) for various input data.

The logic diagram would be as:

**A) Prepare grid data needed for the power flow**

- Retrieve from SCADA topology-related real-time ***grid topology data*** (topology connections, tap changers possibilities, etc) and ***grid parameters data (***reactance, admittances for lines and transformers, maximum currents on feeders)
- Retrieve ***nodal data***:
    - $P_g$ and U for all generators, including renewables, based on forecast of renewables; forecast will be based on next K hours, for instance next 24 hours
    - $P_c$ and $Q_c$ for all loads, based on forecast of consumers; forecast will be based on next K hours, for instance next 24 hours
    - U and θ for the main supply point (main grid connection for balancing with the main grid).
- Identify main feeders (subjects of congestion) and compute main feeders' available capacity; These would be the possibly congested feeders and would also include the transformers as case.

**B) Compute steady state regime through power-flow algorithm (r**un steady state regime for each forecasted scenario at N without N-1**.**

**C) Assess congestions**

Compare results of power flow calculations with determined available capacity for each main feeder within each forecasted scenario. Alarm capacity overload (congestions) based on agreed margins (settable)

Assess voltage levels. Alarm for over and under-voltage.

**D) Evaluate possible grid topology switching** (tap changers, capacitors, reactors, etc) to fix alarmed congestions.

Simulate the operation such as switching devices (mitigation) for congestion fixing.

Run steady state regime for each of the previous forecasted scenario with mitigation switching.

List capacity overload cases (for each main feeder within each evaluated scenario) grouped in N regime (normal grid status).

##### 3.2.11.1.1 Congestion forecast specification

The ***Constrains MAnagement Application*** (CMAP) module will need input data files and will provide output data files with a list of constraints.

The two lists are for:

- The case with basic topology (the topology on normal grid status projected on forecasted production and consumption)
- The case with improved topology, where some predefined changes in topology are considered for the power flow as forecasted

  Input data files for the load flow calculation (Data collection in the Figure below):

- Topology and parameters file from WG Cockpit (considered in figure below also as "normal grid status")

- Forecast of consumption in all nodes ($P_c$ (t), $Q_c$ (t) series).
- Forecast of all production, including on renewables ($P_g$ (t), U(t) series);

  Output data files from the Load flow calculation:
- P, Q, U, θ in all nodes
- P, Q, I on each line

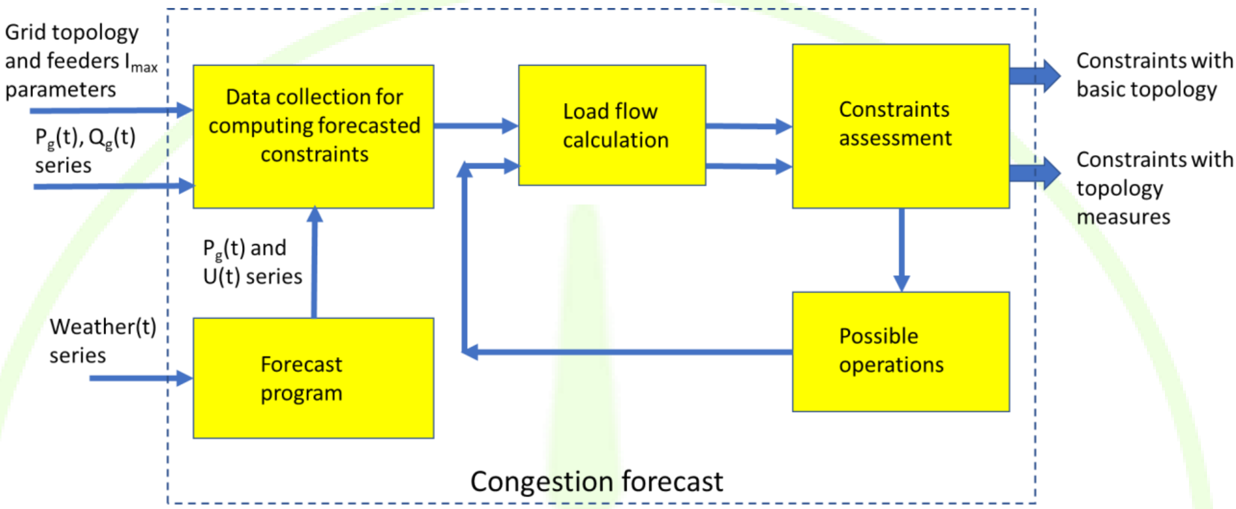



**Figure 22 – The workflow of the CMAP module responsible for constraints management**

The horizon of the future depends on the horizon for the forecast. It is considered as standard horizon a 24 hours timeframe, in steps of one hour. Shorter periods can be also assessed.

For a specific timeframe of K hours (e.g. K=24), as the project receives forecasts based on one hour steps, it means that the set of calculations considers K different steady state power flows regimes. For instance, if the system receives at hour 22:35 a forecast for 24 hours, meaning at hour 01:00, 02:00 .. 24:00 of next day, a corresponding number of 24 steady state regimes (load-flows) will be generated for each one hour. The forecast should be received with more than one hour before first assessed congestion period.

This procedure can be described formally as follows:

- At t= $T_0$, the constrains management application (CMAP) asks to WG COCKPIT for the grid status by writing in a specific subdirectory a CMAPtoWG COCKPIT_req.txt file; after a certain period WG COCKPIT writes a WG COCKPITtoCMAP_ans.txt file with all grid status data written in the text (format to be refined); the request is honoured in Δt1 < Δt1WG Cockpit_timeout, which means no later than Δt1WG Cockpit_timeout = 1 to 2 minutes after the request (timeout).
- CMAP selects consumption and production points and sends to the forecast program (FOP) a request for a forecast, by writing in a specific subdirectory a CMPAtoFOL_req.txt file; after a certain period FOP writes a FOPtoCMAP_ans.txt file with all forecast data (generation and consumption) written in the text (format to be refined); the request is honoured in Δt2 < Δt2fop_timeout, which means no later than Δt1fop_timeout = 1 to 2 minutes after the request (timeout is 1 to 2 minutes)
- CMAP prepares an input file for the load flow calculation application (LFC) and sends to LFC a request for a load-flow calculation, by writing in a specific subdirectory a CMPAtoFOL_req.txt file; after a certain period LFC writes a LFCtoCMAP_ans.txt file (or specific results of LFC, which will be

converted in a LFCtoCMAP_ans.txt file (formats to be refined); the request is honoured in Δt3 < Δt3lfc_timeout, which means no later than Δt1lfc_timeout = 30 to 45 seconds after the request (timeout is 30 to 45 seconds); the load-flow is made for each time interval of the forecast (max 24 runs)

- CMAP sends the results to the Constraints manager (CnsMng) the output file
- CnsMng selects the nodes and lines where constraints are violated (U outside boundaries and/or current I greater than a limit) and give a report as a file with the name CnsMngtoCMPA_ans.txt;
- CMAP analyses if there are constraints, and if so, a set of usual operations are simulated by CMAP, by producing a set of additional scenarios CMPAtoFOL_req.txt, which are sent to LF and then to CnsMng. For each scenario CMPA receives a report in a CnsMngtoCMPA_ans.txt file.
- CMAP makes a final report showing the nodes and lines where there are constraints, within a file CMAP_ans.txt, which is written in a specific subdirectory, to be read by WG Cockpit.

The following time-flow applies to the whole process:

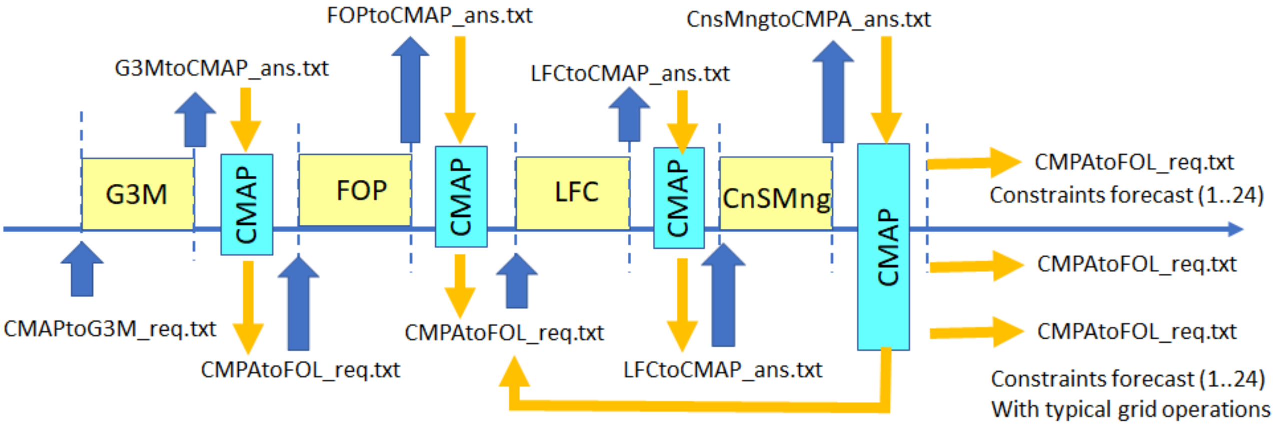



**Figure 23 – time sequence of messages**

For a forecast of 24 hours, a complete process, without grid operations will take maximum:

T_max = 2 + 2 + 0.75 x 24 = 22 minutes

Where 0.75 means 45 seconds, or 3/4 minutes.

For each additional grid operation, it is used additional time

T_max_go = 18 minutes

It is advisable to have at least two different grid operations (usual operations of the grid operator) parametrized in the CMAP, such that in a full hour the process can have:

T_tot_max = T_max + 2 x T_max_go = 22 + 2 x 18 = 58 minutes

For more grid operations to be tested it is needed a quicker LFC run. For reasons of stable interconnection, for a medium size MV or LV grid it is possible to go down to 30 seconds per LFC run + CMAP assessment and files generation for process control. Moreover, the maximum time of interaction with WG Cockpit and FOP may also be reduced to one minute.

A minimum time for constraints plus four different grid simulated operations may give:

T_max = 1 + 1 + 0.5 x 24 = 14 minutes

T_max_go = 0.5 x 24 = 12 minutes

T_tot_max = T_max + 4 x T_max_go = 14 + 4 x 12 = 62 minutes

Which is more than one hour.

This finding shows that a robust operation requests that the whole process should be started in best case every two hours, to allow up to eight different grid simulated operations, which is enough complex to help the grid operator with valuable support.

Moreover, the entire set of calculations may first start with the (emerging) next hour and all its related possible different grid operations. This may give answers for the next hour in:

T_tot_max_1h_low = 1 + 1 + 0.5 x 8 = 6 minutes

In fast mode (all the timings at the lowest values) or in case of higher values in:

T_tot_max_1h_high = 2 + 2 + 0.75 x 8 = 10 minutes

This time of reaction is fast enough to give in-time data to the grid operator in WG Cockpit platform.

The grid simulated operations will be in of the following types:

- Change the place for sectioning an MV loop;
- Modify tap changer to a transformer
- Change the voltage set-point to a PV production plant which is controlling the voltage through reactive power.

The chosen operations are based on grid operator experience.

###### 3.2.11.1.2 Configuration required by the moduleS (parameters, config. files)

As per Figure 1, the congestion forecast module need to interact with the Cockpit modules as well as with forecast prediction of both production and consumption.

The Congestion Forecast Application (CFA) data flow and communication details are given in Figure 3.

A Congestion Forecast Orchestrator (CFO) is organising the procedure for obtaining congestion forecast information, to be used by WiseGRID Cockpit.

In the figure are presented:

- Triggers for external independent application
- Data files which pass the information from a module/application to another one.
- Independent modules which interact, in order to implement the whole FCA functionality.

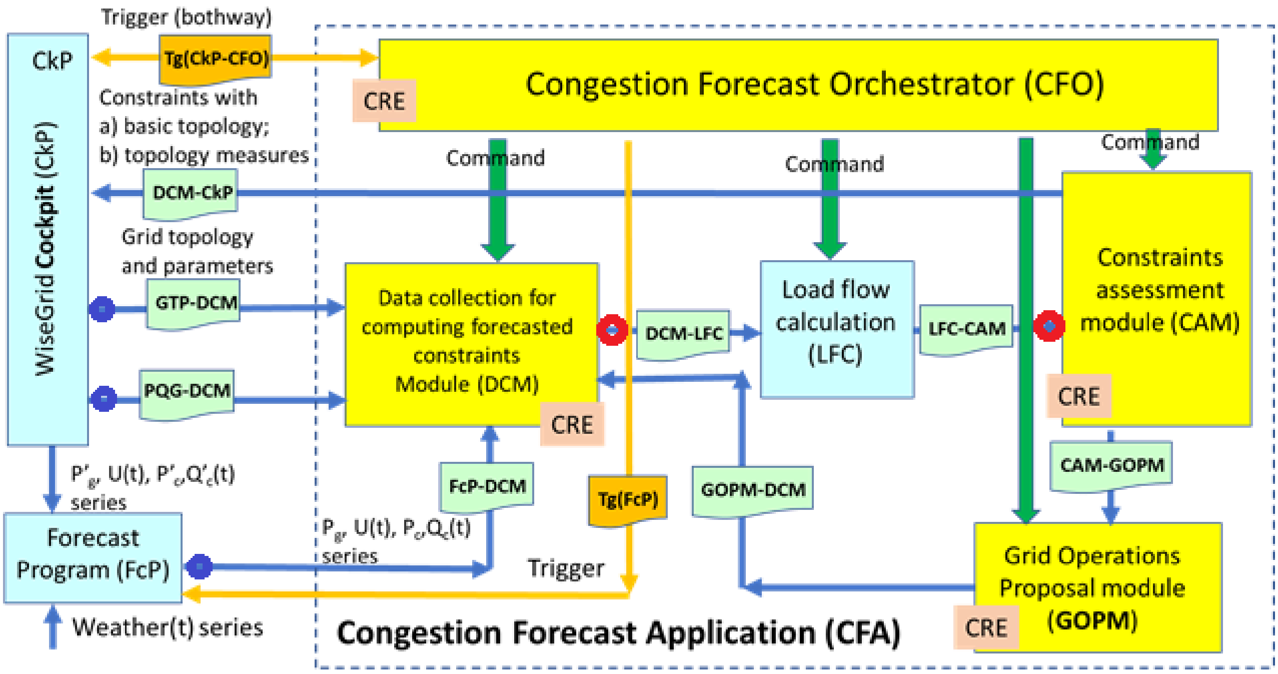



**Figure 24 – Congestion Forecast application (CFA) data flow and communication details**

The BLUE points indicate triggers produced by the module "congestionForecastScheduler". This module will periodically retrieve the data from the different sources (CIM topology model, forecast module, long-term database…), compose and publish the required updated data under the specified MQTT topics (GTP-DCM, PQG-DCM, FcP-DCM)

The RED points indicate interfaces covered by the module "congestionForecast3PPFBridge" This module will:

1. Subscribe on MQTT to topic "Congestion_Forecast/DCM-LFC" to listen messages published by DCM
2. Transform JSON to Excel
3. Publish Excel to the LFC (3 phase power flow module) using AMQP
4. Retrieve the response from LFC in Excel format
5. Transform Excel to JSON
6. Publish JSON to topic "Congestion_Forecast/LFC-CAM"

#### 3.2.11.1.3 Triggers.

There are two triggers for interaction between FCA and independent external applications:

- A trigger between CFO and the WiseGRID Cockpit; this trigger can be unidirectional, or bi-directional, to be decided. In case that the trigger is initiated by CFO, then CkP will produce the files GTP-DCM and PQG-DCM. If the trigger is from CkP, then CkP need to produce in advance to the trigger the same files GTP-DCM and PQG-DCM.
- A trigger between CFO and FcP; This trigger will make FcP to produce the file FcP-DCM

#### 3.2.11.1.4 Modules and I/O files description

Each internal module of FCA will run through express request of the orchestrator (FCO). When staring the execution of the module, each module will read already made input files and will produce output files. The files are always of JSON type, except eventually for the LFC which is an already made application

##### 3.2.11.1.4.1 Module DCM

DCM (Data collection for computing forecasted constraints Module) need as inputs files:

- GTP-DCM (Annex 7.1)
- PQG-DCM (Annex 7.6)
- FcP-DCM (Annex 7.2)
- And as output files:
- DCM-CkP (Annex 7.7)

The exact format of the JSON files are presented in Annex 1

##### 3.2.11.1.4.2 Module CAM

CAM (Constraints assessment module) need as input file:

- LFC-CAM (Annex 7.5)
- And as output files:
- CAM-GOPM (Annex 7.8)

The exact format of the JSON files are presented in Annex 1

##### 3.2.11.1.4.3 Module GOPM

GOPM (Grid Operations Proposal module) need as inputs files:

- CAM-GOPM (Annex 7.8)
- And as output files:
- GOPM-DCM (Annex 7.3)

The exact format of the JSON files are presented in Annex 1

##### 3.2.11.1.4.4 Module LFC

LFC (Load Flow Calculation) need as inputs files:

- DCM-LFC (Data collection for computing forecasted constraints Module → Load Flow Calculation), described in Annex 7.4
- And as output files:
- LFC-CAM (Load Flow Calculation → Constraints assessment module), described in ANNEX 1: CONGESTION FORECAST EXCHANGE FORMAT

#### 3.2.11.2 Coordination with WiseGRID Cockpit

In order to orchestrate the operation of the Congestion Forecast, a number of additional modules has been developed. The demand and production forecast modules developed with the WiseGRID project, as used in the context of the WiseGRID Cockpit application, analyse the demand and production profiles per bus and provide forecasts on active and reactive power terms. Since the Congestion Forecast also requires forecast on voltage magnitude per bus, a power flow analysis on the forecasted scenarios becomes necessary.

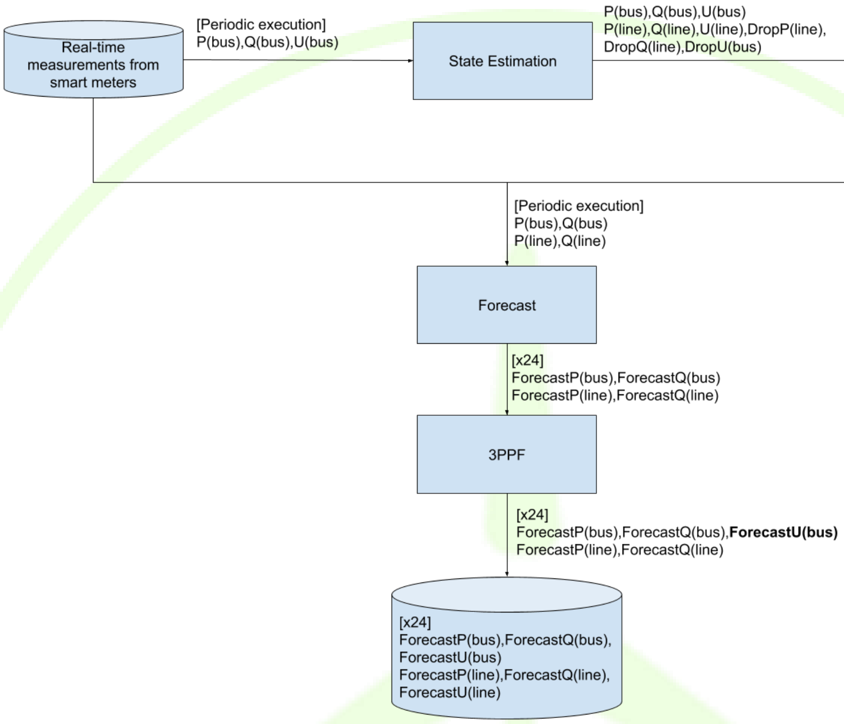



**Figure 25 – Congestion forecast orchestration, power flow analysis on forecasted scenarios**

Once the forecast of the voltage per bus and phase has been obtained, an initiator module is in position of periodically composing and publishing in the internal ESB the three different files required to start with the computation of the congestion forecast:

- GTP-DCM information can be extracted mainly from the CIM description of the pilot site topology, together with some static configuration needed during the commissioning of the WiseGRID Cockpit application in each pilot site (e.g. operational limit of current in each of the lines)
- FcP-DCM information can be directly extracted from the forecast results explained above
- PQG-DCM can be extracted directly from the operational database, which registers the latest measurements received by each one of the monitored sensors

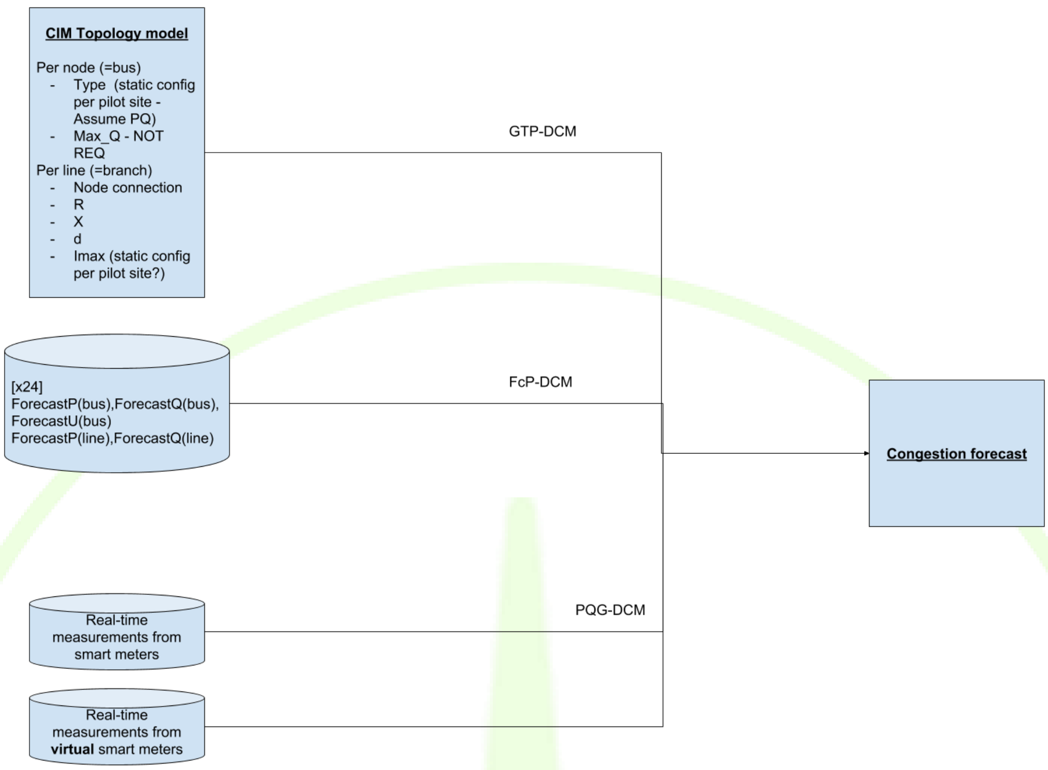



**Figure 26 – Congestion forecast orchestration, triggering generation of all necessary inputs**

Additionally, a translation module has been developed to help bridging the modules that expect input/output in different formats. Congestion Forecast module internally uses the Power Flow module as part of the assessment of the different possible congestion scenarios. This module is therefore responsible for translating from JSON to Excel format, making it possible to them to interact and exchange information seamlessly.

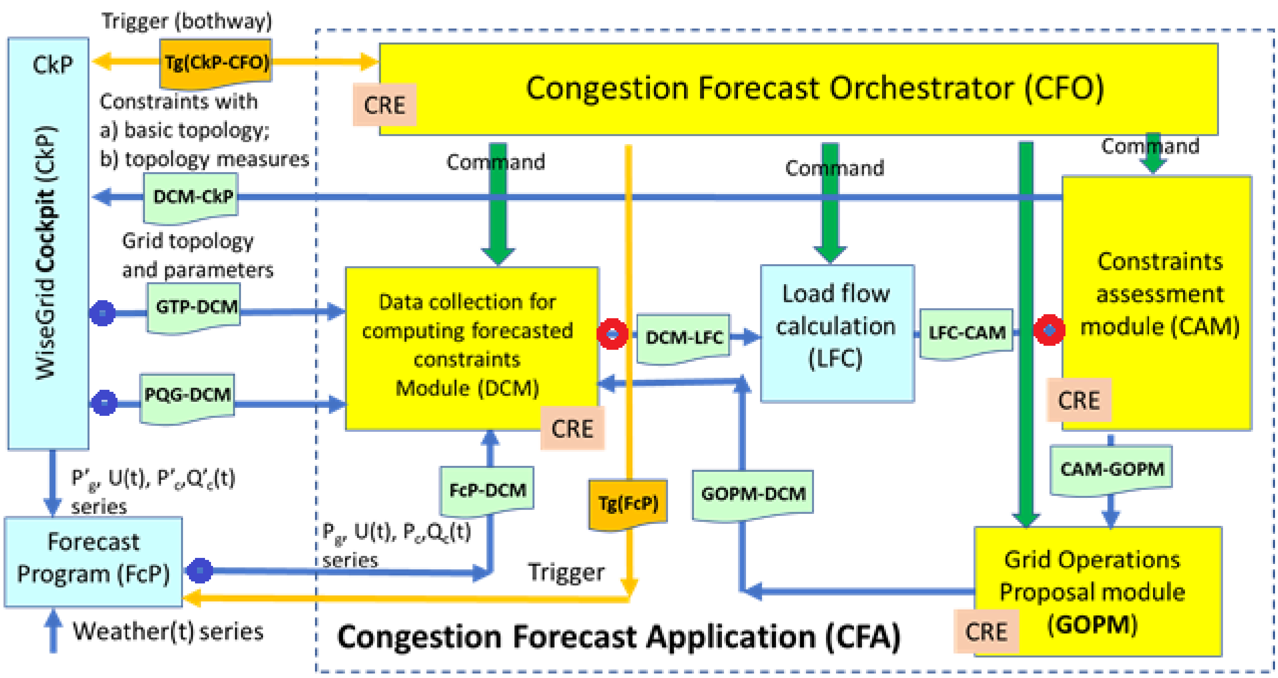



**Figure 27 – Congestion forecast orchestration, red points are supported by the translation module**

### 3.2.12 Grid fault manager

WG Cockpit will integrate a component for incident management allowing creating and monitoring incidents notified by other modules of the platform. This module will allow generalization and customization of the process to be performed upon an incident by using a BPMN engine as its core.

The implementation has been based on the open-source BPMN engine Flowable [7].

**Figure 28 – Flowable BPMN Engine is used at the core of the Grid Fault Manager**

The integration of this module within the WiseGRID Cockpit has followed the design presented in D13.1. Each one of the modules making different analysis on metrics of the grid are free to post an *incident* object to a specific collection of the database whenever an anomalous situation is detected (anomalous metric, congestion forecast…). The Real-time monitor module is configured to monitor this incident collection, and trigger the corresponding workflows defined in the Flowable BPMN engine. The DSO operator can use Flowable UI to define any action needed to be taken automatically by the system as a reaction to those events.

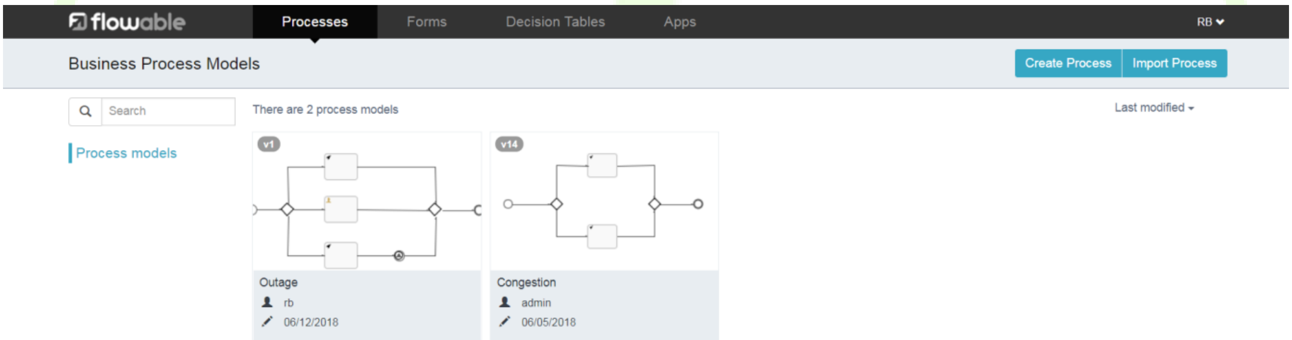



**Figure 29 – Integration of Flowable with WiseGRID Cockpit, workflows defined for Outage and Congestion events**

As already detailed in the design deliverable, BPMN allows the definition of several different types of actions, not only to be executed by other modules of the WiseGRID Cockpit, but also being able to trigger and integrate external actions, thus introducing a great versatility and freedom in the actions to be executed when an event is detected. Examples have been developed showing how to trigger internal modules of the WiseGRID Cockpit (e.g. automatically triggering demand-response campaigns or registering maintenance actions) and how to integrate social media to keep customers informed of any actions being taken by the DSO (e.g. by posting tweets on the DSO account).

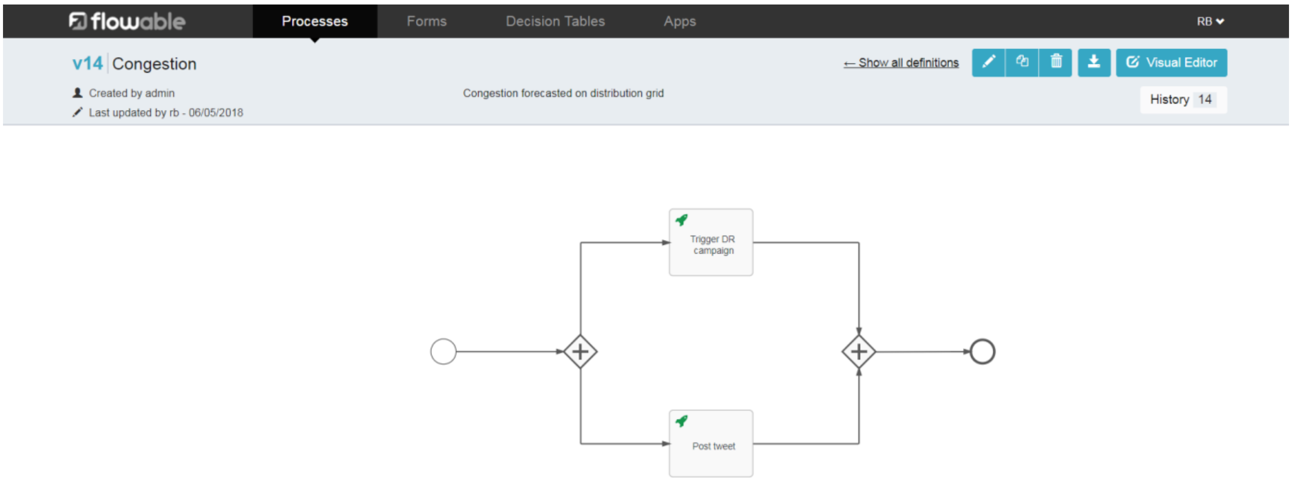



**Figure 30 – Integration of Flowable with WiseGRID Cockpit, detail on workflow to be executed upon forecast of congestion**


**Figure 31 – Example of workflow automatically updating social media of the DSO**

### 3.2.13 FLISR

This module can be executed in three self-healing modes. In the first mode all switches (both remote and manually controlled ones) are considered for the reconfiguration. In the second mode only remote ones are considered for the reconfiguration. In the third mode, topology optimization is carried out in two phases. In the first one, only remotely controlled switches are considered in order to speed up service recovery without human intervention. In the second phase, only manually controlled switches are considered to further reduce the extension of the area without service.

Some data tables have been slightly modified regarding to deliverable D13.1 but all of them are included again for the sake of information consistence.

#### 3.2.13.1 Fault Location submodule

***Inputs***

The required data inputs are gathered on the following table:

**Table 16 – FLs Data Inputs**

| Data | Description | Type | Use | Origin |
|---|---|---|---|---|
| **Network** | Connectivity of network nodes, branches and switches | *to be defined* | Configuration | Structural DB |

| | | | | |
|---|---|---|---|---|
| **topology and parameters** | Location of fault pass indicators | | | |
| | Line segments impedance | | | |
| **Switching status** | Current status of switches | Vector of boolean values | Running | RT monitor |
| **Fault information** | Status and directionality of FPIs | vector of integer values | | RT monitor |
| | Distance (ohms)from fault distance relays[2] | Vector of float values | | RT monitor |

***Outputs***

The outputs of localization algorithm are summarized on the following table:

**Table 17 – FLs Data Outputs**

| Data | Description | Type | Destination |
|---|---|---|---|
| **Faulted line segments** | Faulted network line segment(s) [3] | Integer value(s) | Service Restoration module |

#### 3.2.13.2 Service Restoration Submodule

***Inputs***

The required data inputs are gathered on the following table:

**Table 18 – SRs Data Inputs**

| Data | Description | Type | Use | Origin |
|---|---|---|---|---|
| **Network topology and parameters** | Connectivity of network nodes, branches and switches. | *to be defined* | Configuration | Structural DB |
| | Location of loads | | | |
| | Location of power sources (head of feeders, distributed generators,…) | | | |
| | Nominal power of power sources | | | |

[2] Fault distance relay is assumed to be located at the head of the substation output feeders.

[3] Depending on the actual DFPI and distance relays deployment, there is the possibility that the algorithm proposes several candidates for faulted line segment.

| | | | | |
|---|---|---|---|---|
| | Type[4] of switching points | | | |
| | Line segments impedance | | | |
| | Line segments current capacity | | | |
| **Network operational state** | Current status of switches | Vector of boolean values | Running | RT monitor |
| | Current load and distributed generators power | Vector of float values | | RT monitor |
| | Faulted line segments | Vector of integer values | | Fault location submodule |

***Outputs***

The outputs of the Service Restoration Submodule are summarized on the following table:

**Table 19 – SRs Data Outputs**

| Data | Description | Type | Destination |
|---|---|---|---|
| **Connection scheme** | Optimal connection scheme suggesting switching points status to be changed after network restoration | Integer vector/bool vector | SCADA Wrapper / Maintenance manager |
| | Switching sequence from current (faulty) topology to of the optimal connection scheme | Integer vector | SCADA Wrapper / Maintenance manager |
| **FITNESS** | Final value (optimal) of the fitness function | Float value | SCADA Wrapper / Maintenance manager |
| **Criteria values** | Final values with respect to every optimization criteria:<br>• Non restored power (MW)<br>• Power losses (MW)<br>• Voltage deviation (%)<br>• Number of switches to be operated<br>• Current capacity of branches (%)<br>• Load imbalance (%). | Float vector | SCADA Wrapper / Maintenance manager |

#### 3.2.13.2.1 Configuration

[4] Operated either manually or remotely.

**Service Restoration Submodule**

This sub-module requires some configuration parameters. Operational ones must be specified each time. User associated parameters are specific of each user:

Table 20 – SRs Operational parameters

| Operational parameters | Description | Type |
|---|---|---|
| **Criteria weights** | Weights assigned to each criterion of the fitness function[5]:<br>• Power losses (MW)<br>• Voltage deviation (%)<br>• Number of switches to be operated<br>• Current capacity of branches (%)<br>• Load imbalance (%).<br>Sum of all weights must be 1. | Float vector |
| **Voltage limit** | Maximum deviation from nominal voltage admitted. Default: 7%. | Real value |
| **Voltage violations** | Quantity of node voltage violations allowed for a solution to be considered a feasible solution | Integer value |
| **Self-Healing** | Indicates if the reconfiguration is done with all the switches (0), only with remotely controlled ones (1) or it is carried out in two phases (2).<br>In the first phase of the last mode, only remotely controlled switches are considered. In the second phase, only manually controlled switches are considered and remote switches operation is disabled. | Integer value |

Table 21 – SRs user associated parameters

| User associated parameter | Description | Type |
|---|---|---|
| **GA parameters** | Configuration of algorithm parameters (operators such as selection, mutation, crossover, …) related to the GA genetic algorithm[6] | Enumerate |

[5] The weight for the criterion of Power Not Supplied (PNS) is fixed to 1, and only the remaining criteria are allowed to be weighted.

[6] Advanced configuration, recommended only for expert users.

#### 3.2.13.2.2 Formal description of API

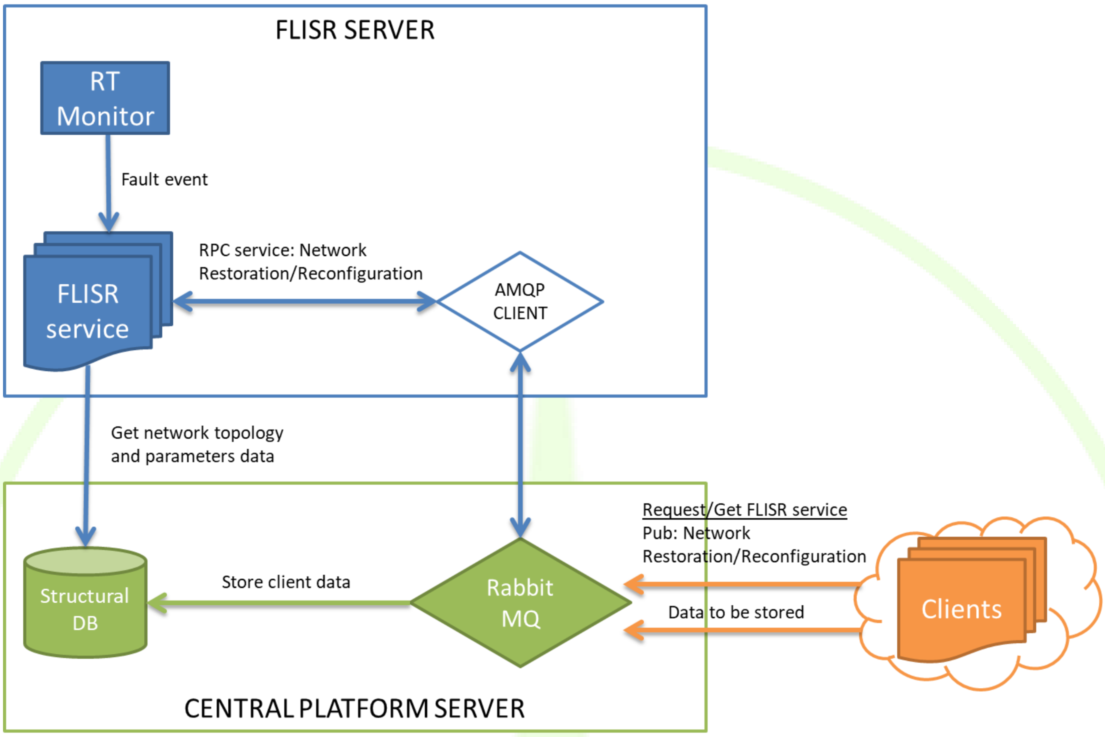



**Figure 32 – API communication**

**Subscribed to**: FLISR (Fault Location Isolation Service Restoration)

**Payload**:

**client**: identifier of the client that ask for FLISR

**Configuration:**

> **criteria_weights:** array of weights assigned to each criteria of the fitness function (excepting Power Not Supplied).
>
> **voltage_limit:** Maximum deviation from nominal voltage allowed.
>
> **voltage_violations**: total quantity of node voltage violations allowed.
>
> **self-healing**: integer indicating reconfiguration with all switches (0), only with remotely controlled ones (1) or carried out in two phases (2) .

Example:

```
[{
        "client_id" :  2,
        "configuration" : [{
                    “criteria_weights”: [ 0.2, 0.2, 0.2, 0.2, 0.2],
                    “voltage_limit”: 0.07,
```

"voltage_violations" : 5,
"self-healing" : 1,
}]
}]

**Response payload**:

[{
"client_id" : 2,
"connection_scheme" : [{10, OFF}, {5, ON},{7, ON}, {1, OFF}]
"FITness" : [-4.564]
"criteria_values" : [3.2, 0.08, 0.12, 4, 0, 0.56 ]
}]

**Results**

An example of the FLISR performance has been carried out on a test network where a set of fault pass detectors (FPIs) are already located in some positions.

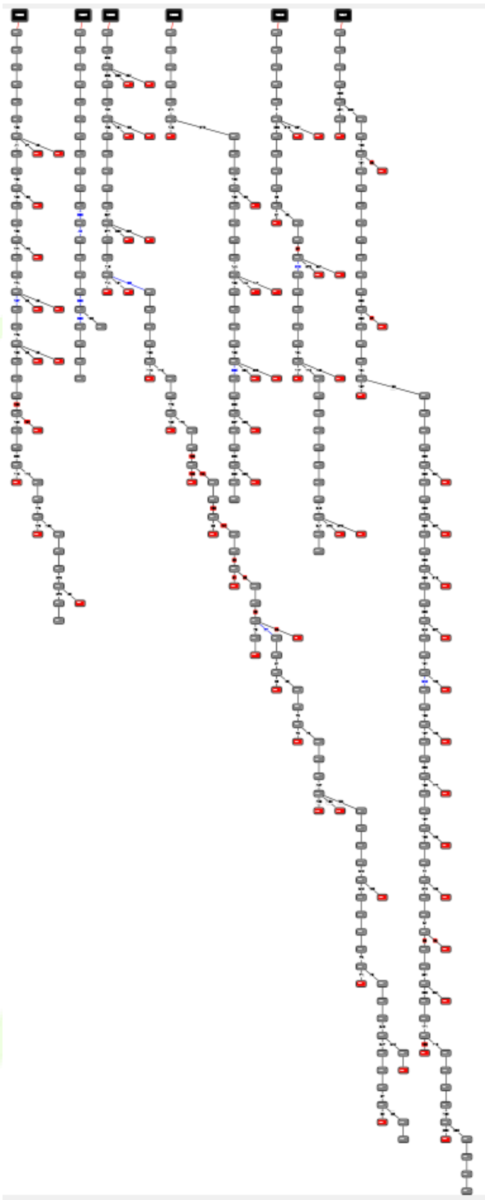


Figure 33 – Original network topology

DPF detectors corresponding to lines 103 and 108 are assumed to be activated. It is further assumed that there is a fault distance detector in the header of the first feeder with (header node = N336)

**Fault Detection**

The Fault Detection submodule provides the following fault information:

| DPFs activated | Fault sectors | Fault lines |
|---|---|---|
| 103 - 108 | 1-7-9 | 13 |

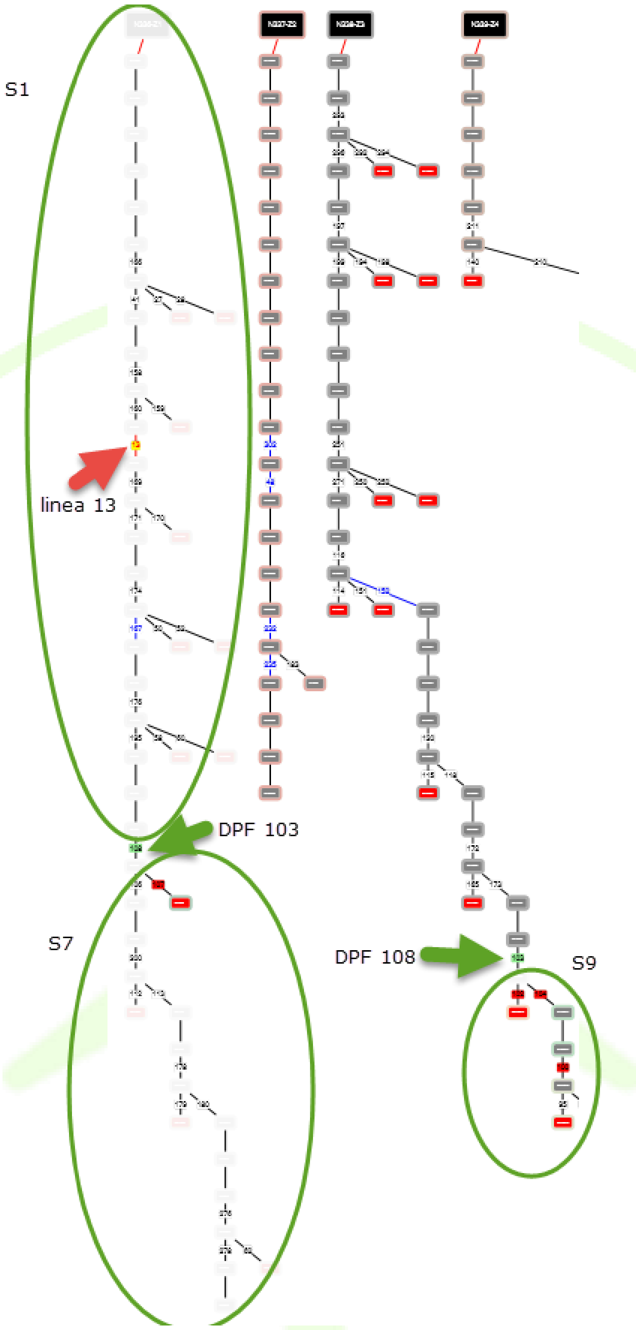



Figure 34 –Line simulation

In the first feeder there are 2 faults, the first is detected by the fault distance detector, indicating that sector 1 and more specifically line 13 is faulted. The activated DPF on line 108 indicates that another fault exists in sector 7 as well due to a fault on load node 297. Finally, DPF on line 103 indicates a fault in sector 9 due to a fault on node N292.

The result is in accordance to what was expected according to the indications provided by DPF detectors and distance relays.

## Service Restoration

The optimal restoration solution is achieved in less than 1 minute for 500 generations, both operating all the switches (0) as only operating remotely controlled ones (1).

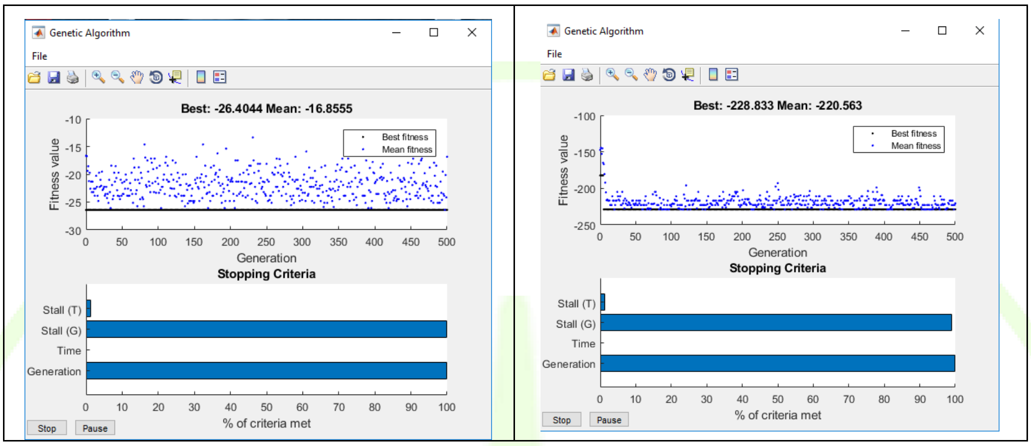



Figure 35 – Service Restoration

In both cases the power not dispatched after the service replacement is less than the initial power not dispatched after the fault is isolated, fulfilling the main objective of the reconfiguration of the network to maximize the replacement of service after failure. We notice that the Non restored power value (criteria (1)) operating both the remote and manual switches is obviously smaller than operating only remote switches since there exist more switching combinations and therefore more feasible radial network reconfigurations

| | MV Network | Fault Node | Fault Branch | OPERATED SWs | SCORE | (1) | (2) | (3) | (4) | (5) | (6) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| #1 OPT | Test Network | 292 , 297 | 67 - 68 | 65 - 77 - 152 - 167 - 193 - 220 - 231 - 234 - 236 | -5,8313 | 0,8574 | 0,0029 | 0,2911 | 9 | 0 | 0,3446 | REMOTE |
| #1 INI | Test Network | 292 , 297 | 67 - 68 | 77 - 152 - 167 | -5,8875 | 1,0506 | 0,0027 | 0,2559 | 3 | 0 | 0,3335 | |
| #2 OPT | Test Network | 292 , 297 | 67 - 68 | 65 - 102 - 103 - 104 - 112 - 160 - 169 - 193 - 220 - 231 - 234 - 236 - 303 | -38,8386 | 0,1250 | 0,0041 | 0,4603 | 13 | 0 | 0,4693 | REMOTE + MANUAL |
| #2 INI | Test Network | 292 , 297 | 67 - 68 | 102 - 103 - 104 - 112 - 160 - 169 | -8,7638 | 0,7524 | 0,0029 | 0,2644 | 6 | 0 | 0,3513 | |

Figure 36 – Restoration results

In Figure 37 – Network topology after restoration the resulting network topology after remote reconfiguration is depicted. We notice that in order to isolate the faults some nodes get disconnected from the feeders

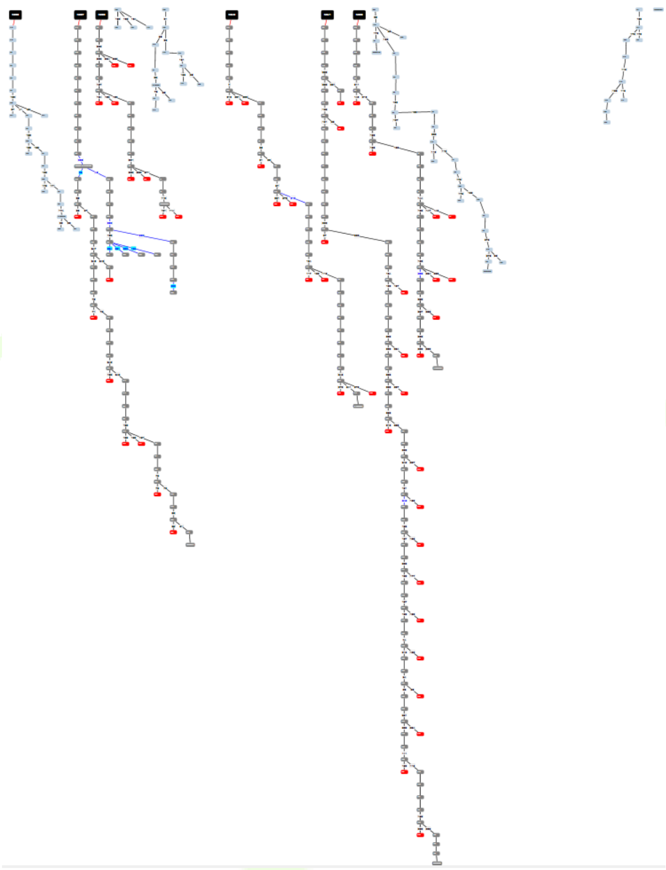


Figure 37 – Network topology after restoration

### 3.2.14 Ancillary services market hub

This module implements the interaction between the DSO and the other tools of the project for the provision of ancillary services (congestion and voltage support, related to explicit demand response campaigns as discussed in D10.2 [8]).

This module connects to the WiseGRID IOP platform, which is used as a common point for intercommunication among the different applications of the WiseGRID ecosystem. The module implements the DSO-side business logic of the Ancillary Services Market as defined by the USEF framework. This includes:

- Triggering demand-response campaigns (flexibility requests): upon detection of a congestion forecast, the necessary information is calculated and the corresponding flexibility request is posted to the Ancillary Services Market.
- Retrieving flexibility offers: the module is responsible for listening and storing all received offers (pairs quantity-price for each settlement period as define here for one hour) within the validity timeframe.
- Processing offers, issuing merit order list, selecting the most appropriate ones and posting the corresponding flexibility orders: once the validity timeframe for reception of offers is over, the module computes which is the combination of offers that fulfils the requested flexibility at the lowest cost, automatically posting orders for those offers.

Table 22 – Extract of market hub logs tracking communications between DSO and three different aggregators

```
etraid@wisegridpre:~/demo$ docker logs --tail 100 -t wgcockpit_mar
```

```
2018-07-18T13:35:01.205417552Z Connecting to RabbitMQ: wise-grid@109.232.32.221/wisegrid

2018-07-18T13:35:09.468060875Z REST server started at http://0.0.0.0:80

2018-07-20T10:58:35.398421492Z >> Request _id=wg4NHFzA4iYhAtkwz Conver-sationID=05407c4b-ae45-4a4a-b7ff-7a3009e2a7a1 MessageID=9ae092e9-cf62-414c-a166-ed231cfec4aa

2018-07-20T10:58:35.435104670Z << Offer _id=7RpPg4aayY9vr63e9 Conversa-tionID=05407c4b-ae45-4a4a-b7ff-7a3009e2a7a1 MessageID=b55d58c4-db3e-4739-a7d1-40b0cff1239c

2018-07-20T10:58:42.820705975Z << Offer _id=xAycZfe97aeermXuK Conversa-tionID=05407c4b-ae45-4a4a-b7ff-7a3009e2a7a1 MessageID=80f3b346-938d-406b-bcac-af4e4f6ee429

2018-07-20T10:58:42.954783986Z << Offer _id=AdFMS8pmFtx7wWofJ Conversa-tionID=05407c4b-ae45-4a4a-b7ff-7a3009e2a7a1 MessageID=ef7c90a1-b87e-4824-a124-616ab3794c18

2018-07-20T10:59:35.605665543Z -- Evaluating offers for _id=wg4NHFzA4iYhAtkwz ConversationID=05407c4b-ae45-4a4a-b7ff-7a3009e2a7a1

2018-07-20T10:59:35.855395177Z >> Order _id=gFAAdvyWuth9EATgo Conversa-tionID=05407c4b-ae45-4a4a-b7ff-7a3009e2a7a1 MessageID=3d3adbbd-51ba-404b-8d51-46a3e25468e7
```

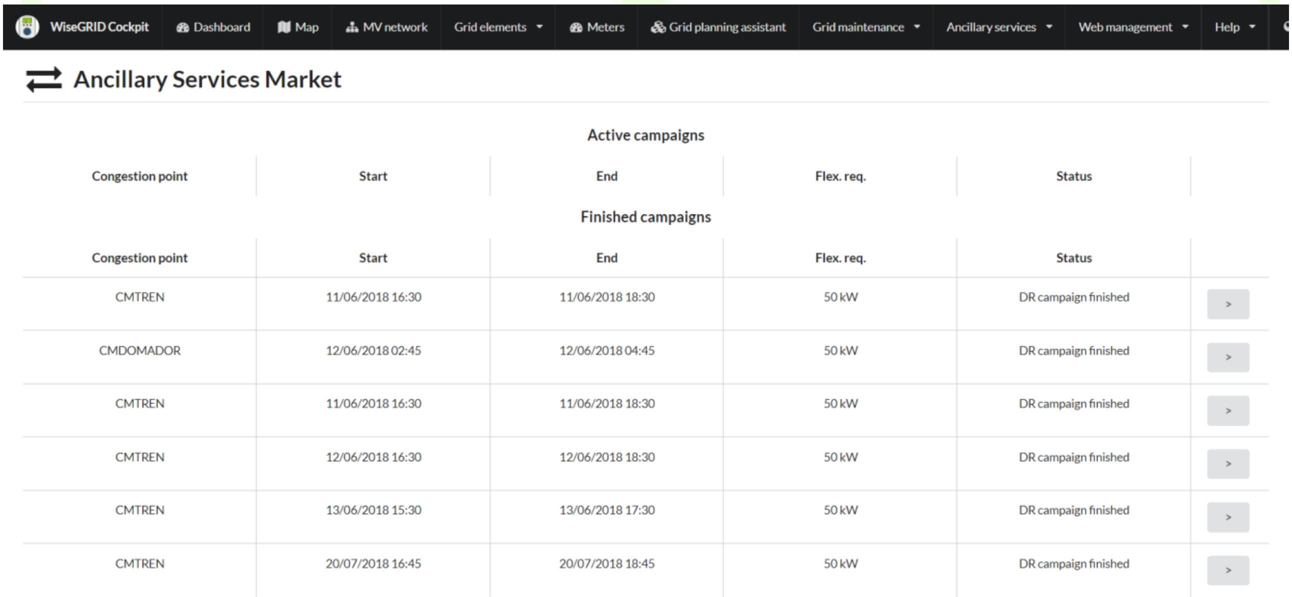



**Figure 38 – Details of the events taking place in the Ancillary Services Market, displayed on the UI**

### 3.2.15 Demand and production forecast services

This module of the WG Cockpit has been implemented with the RabbitMQ libraries to manage the queries to this module. In addition, this module makes use of the long-term database of the WG Cockpit, which is implemented over a MongoDB database. The RPC servers of this application are permanently running to manage the received queries through the RabbitMQ queues enabled to make use of the demand and production forecast.

Within the message queries are specified the id of the supply point, and the period and the horizon of the desired forecast. In the case of production forecast, in addition of the defined fields, it is specified the type of generation technology.

Once the query is deserialized and parsed, the forecast module retrieves from the long-term database the necessary information to perform the forecast. To perform the forecast it is retrieved information related to the consumed/produced energy, working calendar, and weather information related to the queried installation, like in the case of the load demand and peak forecast module.

Once the algorithm is run, the response provided by it is serialized and sent back through the corresponding RabbitMQ queue, providing the queried information. The next message is an example of the response received by the WG Cockpit application, which is printed in the graphic of the forecast view.

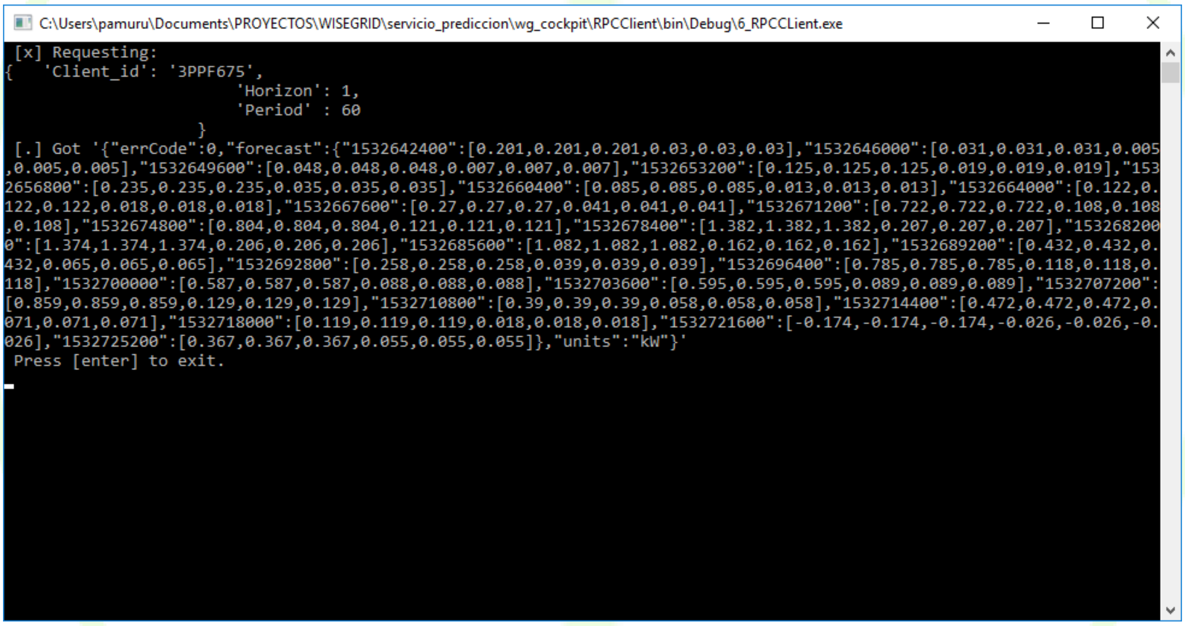



Figure 39 – Screenshot of forecast response message.

### 3.2.1 Maintenance manager

The maintenance manager is an independent module focused on integral maintenance. The module can be hosted in the cloud (SaaS), and can be interfaced using a REST web service. It also offers web-based access to show the managed information and perform actions, and an Android App for the crew.

The module has been integrated with the WiseGRID Cockpit as one of the actions that can be configured by the DSO operator to trigger whenever a problem is detected by other modules of the WiseGRID Cockpit. The Grid Fault Manager (based on the BPMN engine *Flowable*) can be configured to trigger creation of new maintenance actions by using the REST API of the Maintenance Manager

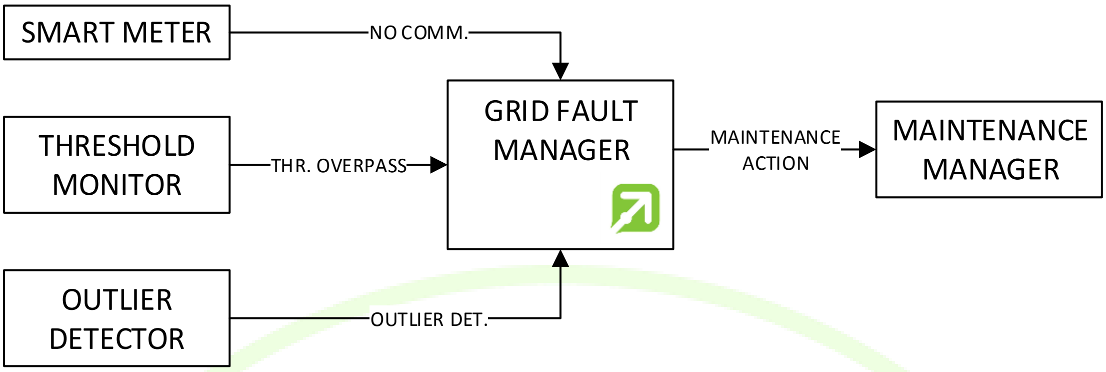



Figure 40 – Schematic overview of the integration of Maintenance Manager

The following table shows the kind of events configured so far to be treated by the Maintenance Manager. Nevertheless, due to the versatility inherent to the Grid Fault Manager, this list is open to modifications and additions if required by the DSO in the different pilot sites.

Table 23 – Actions handled by Maintenance Manager

| Source | Event | Action |
|---|---|---|
| **Smart meter (SMX)** | No communication | Maintenance action open on corresponding smart meter |
| **Threshold monitor** | Voltage overpass detected on bus | Maintenance action open on corresponding substation |
| **Outlier detector** | Voltage outlier detected on bus | Maintenance action open on corresponding substation |

## 3.3 USER INTERFACE

In this part of the document, the main sections and functionalities of the WiseGRID Cockpit GUI are described, including some screenshots of the actual interfaces.

For the implementation of the WG Cockpit unified GUI, the **MeteorJS** web framework has been used. MeteorJS (or simply 'Meteor'), is *"A free and open-source JavaScript web framework written using Node.js. Meteor allows for rapid prototyping and produces cross-platform (Android, iOS, Web) code. It integrates with MongoDB and uses the Distributed Data Protocol and a publish–subscribe pattern to automatically propagate data changes to clients without requiring the developer to write any synchronization code. On the client, Meteor depends on jQuery and can be used with any JavaScript UI widget library".*

On the client part of a Meteor application, a number of plugins and technologies can be used to provide user a better user experience. The main plugins we have used for the client side are:

- **SemanticUI** as CSS framework. CSS frameworks are pre-prepared software frameworks that are meant to allow for easier, more standards-compliant web design using the Cascading Style Sheets language. They are mostly design oriented and unobtrusive. This differentiates these from functional and full JavaScript frameworks. By using this CSS framework we achieve easily a modern and coherent style across the whole user interface. The selection of SemanticUI over other CSS framework is mainly based on our expertise and the fact that this one has been designed to be easily understandable and usable. Other framework tends to become quite hard to use as interfaces becomes bigger

- **LeafletJS** as the mapping solution for the web client. This JavaScript-based framework provides a wide range of mapping providers to use and offers a big set of plugins to personalize the user interaction with the map and the display of the information. And everything is open source and free
- **HighchartsJS** is a charting JavaScript framework that helps displaying data in the form of charts for web environments
- **BlazeJS** for the user interface lay out**.** It is a powerful library for creating user interfaces by writing reactive HTML templates.

For the server part, **MongoDB** has been used as the database for keeping real-time data. This is a no-SQL database that helps storing unstructured information. It is tightly coupled with Meteor. The reactive nature of the data changes in MongoDB database is at the core of the web application.

The web application is protected with a user/password credential system to avoid non-authorized personnel to access sensible information. These credentials are requested before accessing the rest of the application.

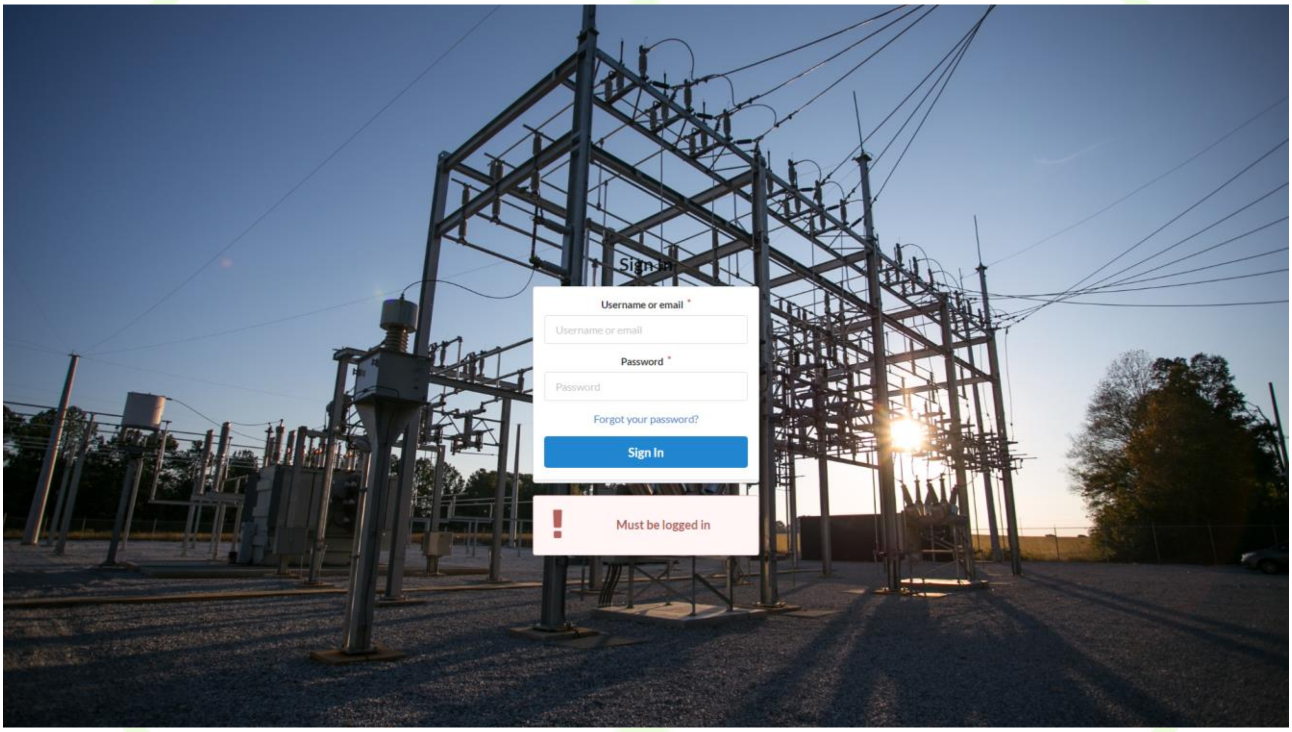



**Figure 41 – WG Cockpit UI – Login**

This credential system also permits the definition of different user profiles to grant or deny access to each section of the application independently. This functionality provides an additional level of privacy, as well as flexibility for the system administrator and the operators that make use of the application.
Once the user has been granted access to the application, diverse functionalities will be available as described in the sections below.

### 3.3.1 Dashboard

The dashboard presents an overview of the current situation of the grid, by displaying 4 different charts:

- Total demand measured in the grid over last hour
- Total production in the grid over last hour
- Next 24 hours demand forecast
- Next 24 hours production forecast

In addition, the dashboard contains additional relevant information for the DSO, such as a summary of forecasted congestions for the next 24 hours, and the current weather.


Figure 42 – Dashboard

### 3.3.2 Map

The map section displays all monitored georeferenced assets on a map. Those assets include the lines, substations, renewable energy sources and usage point locations. By clicking on each one of those elements, the corresponding section displaying further details is presented.

The user can select among two different views of the assets:

- Status view: in this view the colour code represents the current status of each one of the assets, making it very easy to find the assets on which an incident is open (e.g. operational limits have been reached, anomalous measurements have been detected, congestion is forecasted…)
- Topology view: in this view, the colour code represents relationships among the elements represented. For instance, all lines fed by the same substations follow the same colour, making it very easy to visually inspect those dependencies.

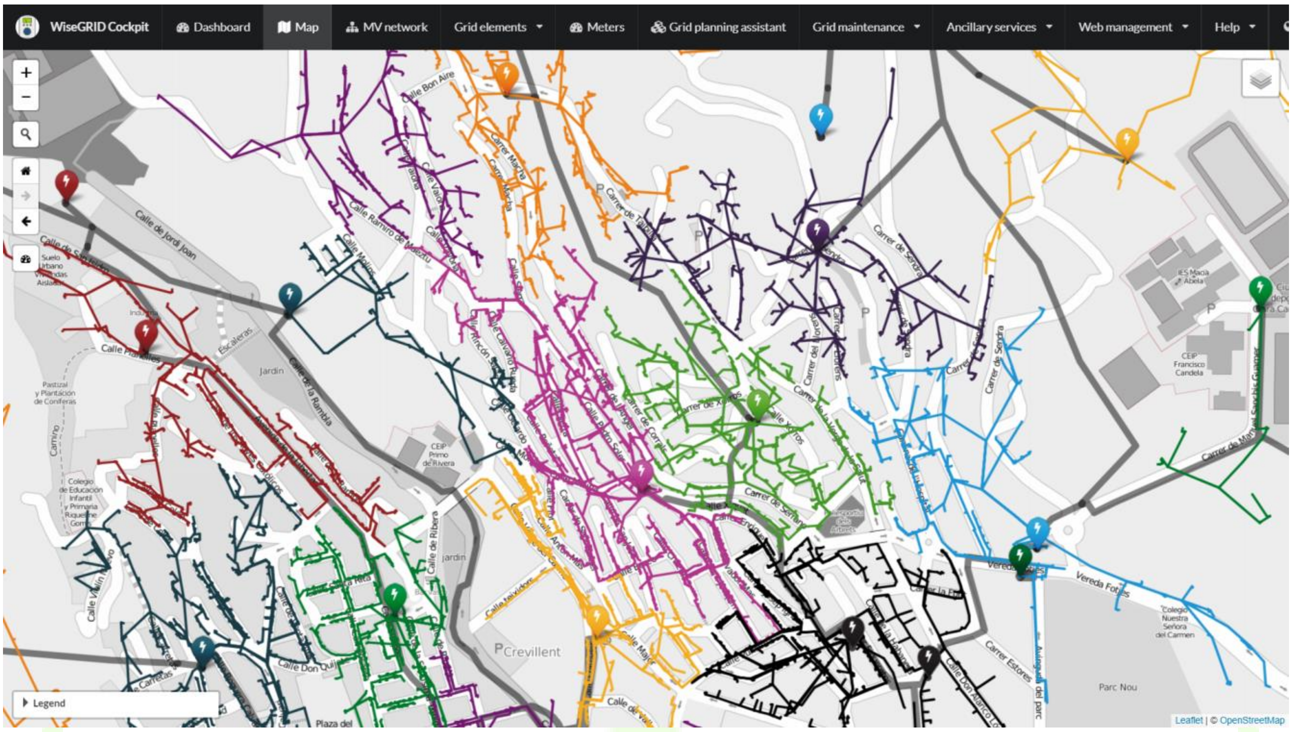



Figure 43 – Map section, topology view

### 3.3.3 MV network

Under this section, the schematic view of the medium-voltage network is represented. There are two different views of the grid considered at this point:

- Single-line diagram view: permits the DSO operator to easily inspect the single-line diagram of the grid within the WiseGRID Cockpit, getting rid of the need of other applications (e.g. CAD viewer application)

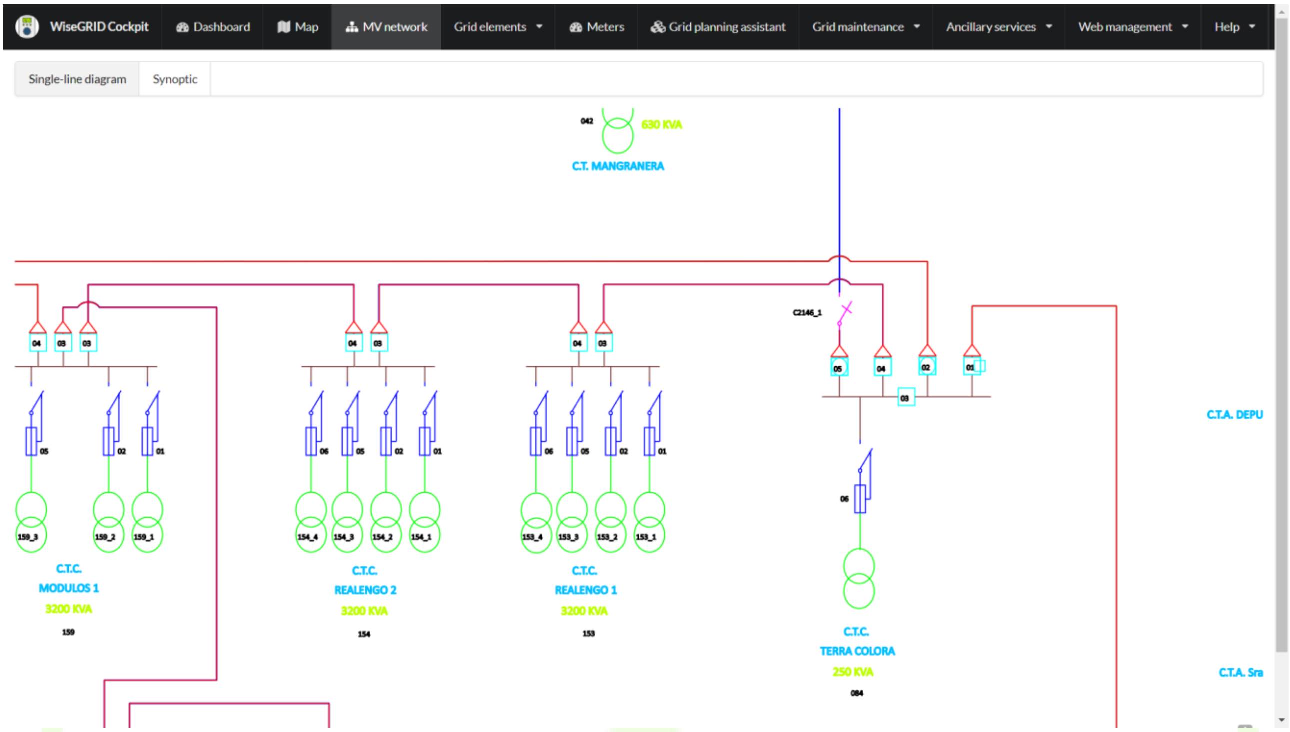



Figure 44 – MV network, single-line diagram

- Synoptic view: the synoptic view is a simplified representation of the medium voltage network monitored by the WiseGRID Cockpit. Under this view, substations are represented as boxes and the connections among those are displayed. A colour code is followed representing the status of the substations (e.g. if an alert has been detected, or no data is available). This representation is also interactive, in the sense that by clicking in each one of the elements the corresponding section displaying further information about the element is loaded

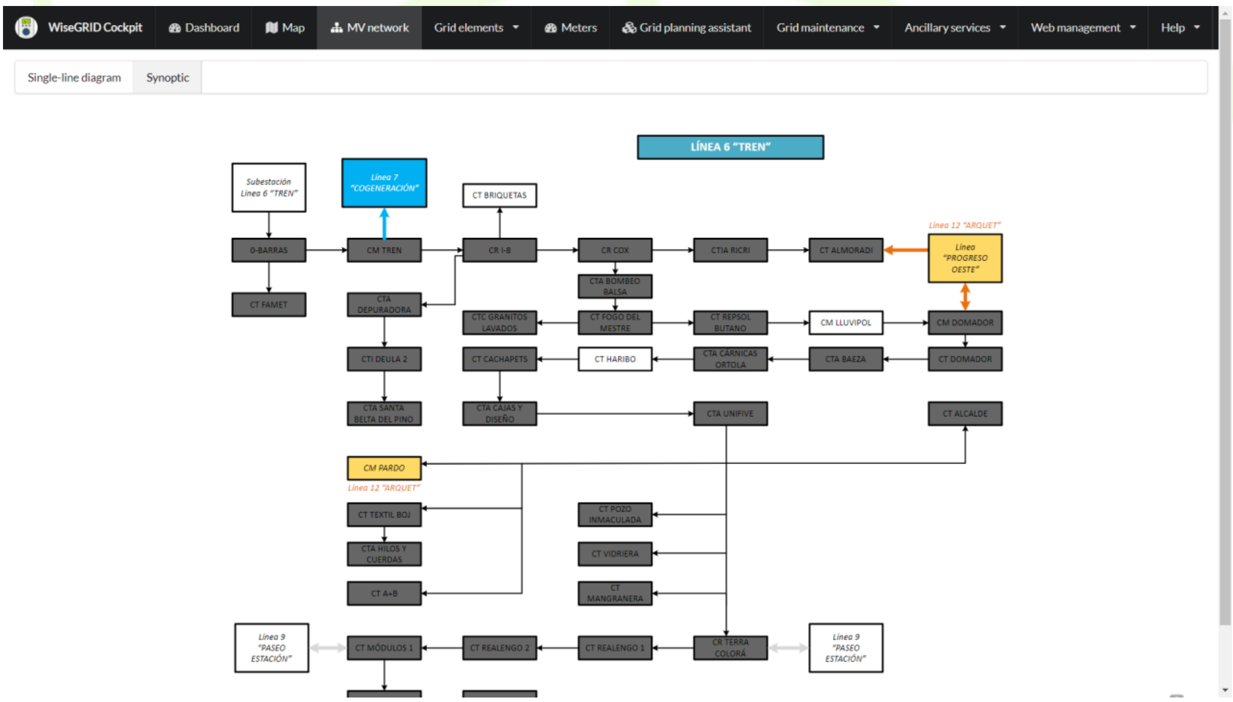



Figure 45 – Synoptic view

### 3.3.4 Grid elements – Tree map

This section allows the DSO operator to inspect all elements of the distribution grid that have been modelled in the CIM Topology model, particularly focusing on the dependencies among those. For this reason, the CIM model is represented in the form of a tree, starting with the geographical regions that contain all the other elements. The DSO operator is enabled to navigate in a friendly way through the topological definition, whose information is complemented with the georeferenced visualization of each one of the elements, if it is available.

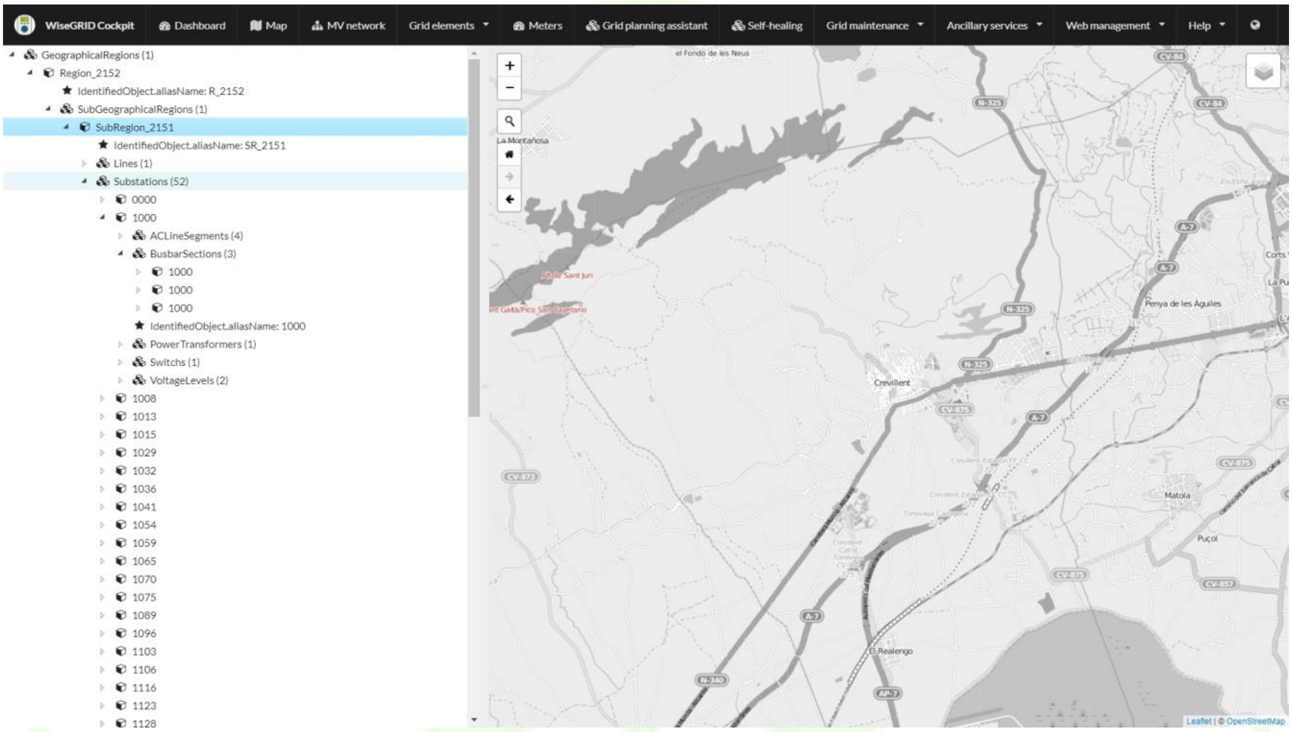



Figure 46 – Grid elements, tree map

### 3.3.5 Grid elements – Assets

Similarly to the previous section, this section allows the DSO operator to inspect all elements of the distribution grid that have been modelled in the CIM Topology model, along with its details and relationships. All defined elements are shown including the following ones:

- AC line segments
- Base voltages
- Connectivity nodes
- Voltage levels
- Substations
- Switches
- Power transformers
- Lines
- Geographical regions
- Sub-geographical regions

The UI automatically gets adapted to display the characteristics of each one of the assets, as defined in the CIM model.

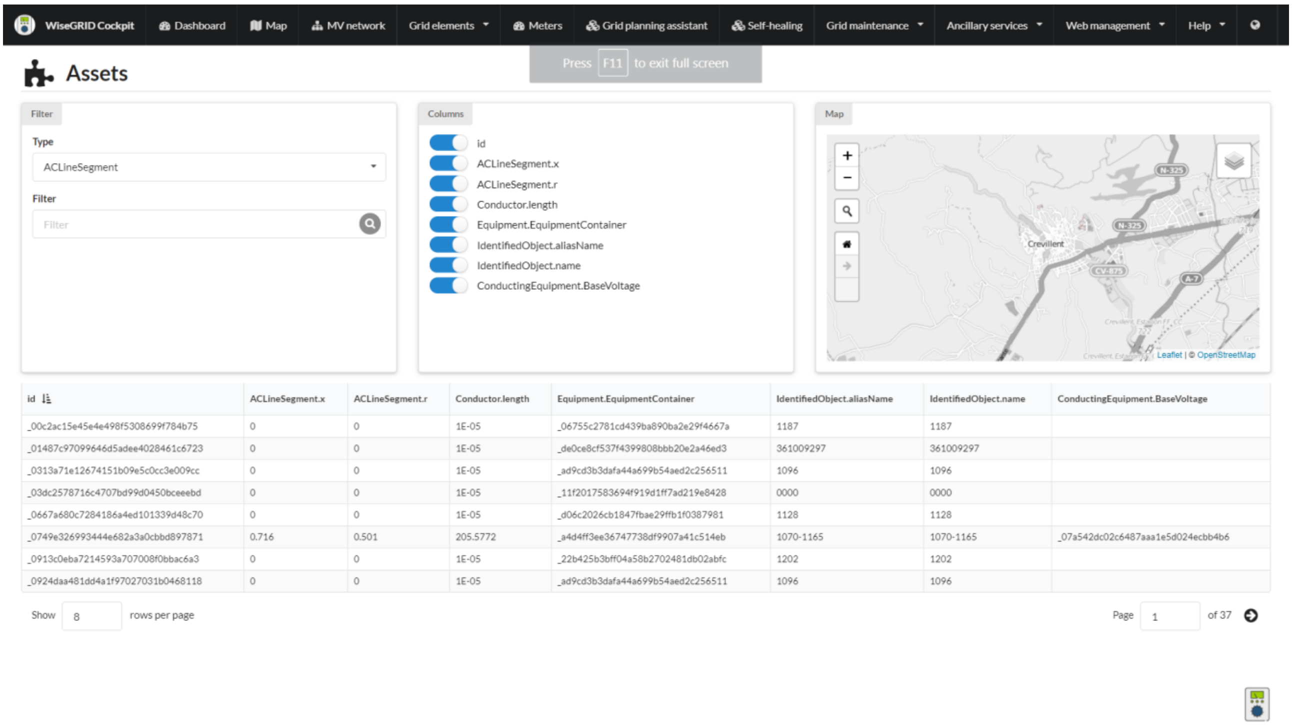



**Figure 47 – Grid elements, assets**

### 3.3.6 Grid elements – Usage point locations

This section provides an insight of all the usage point locations managed by the DSO. A filter can be used to search for a specific usage point location by typing its id, name or address. A small map shows the location of the usage point locations that match the filter.

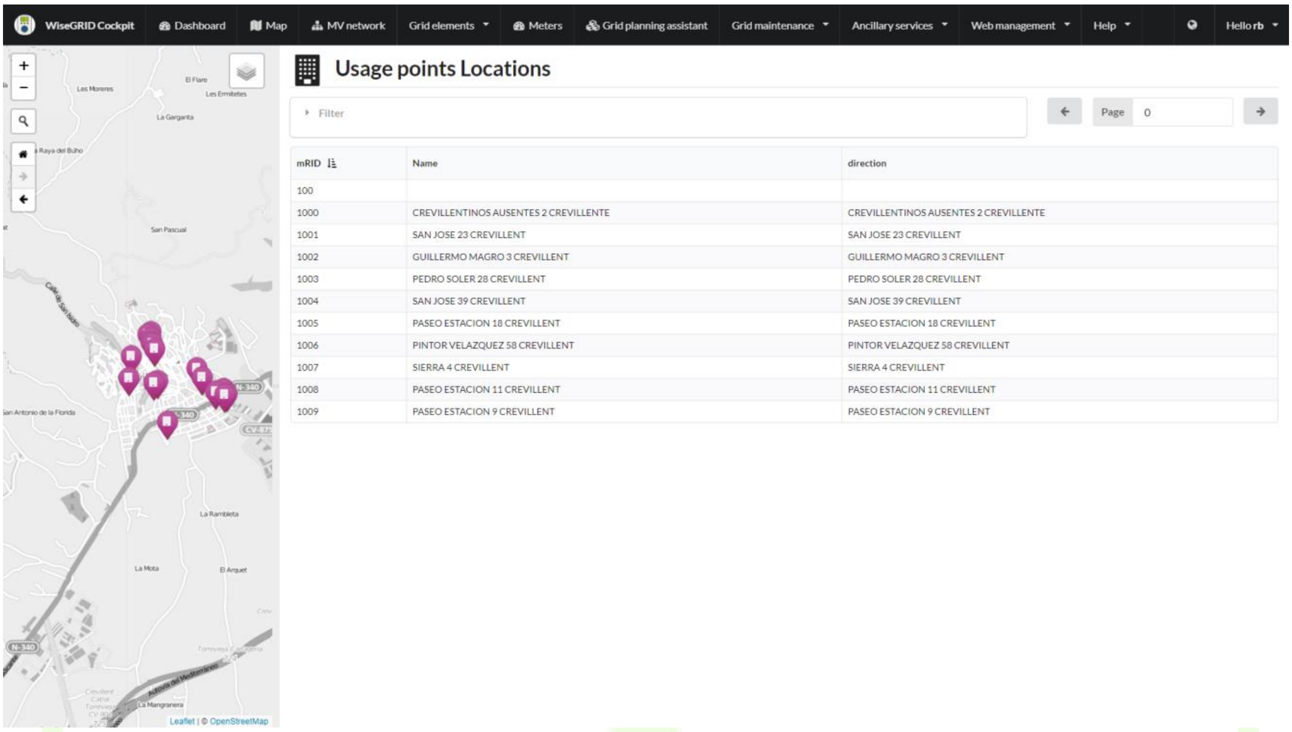



**Figure 48 – Grid elements**

By clicking in any of the usage point locations presented, the UI shows further details on that particular element, namely:

- Properties according to the CIM model of the usage point location (id, access method, direction, remarks, site access problems, type)
- Location of the usage point location on a map
- List of usage points located at this location, as well as the meter ID linked to those usage points.

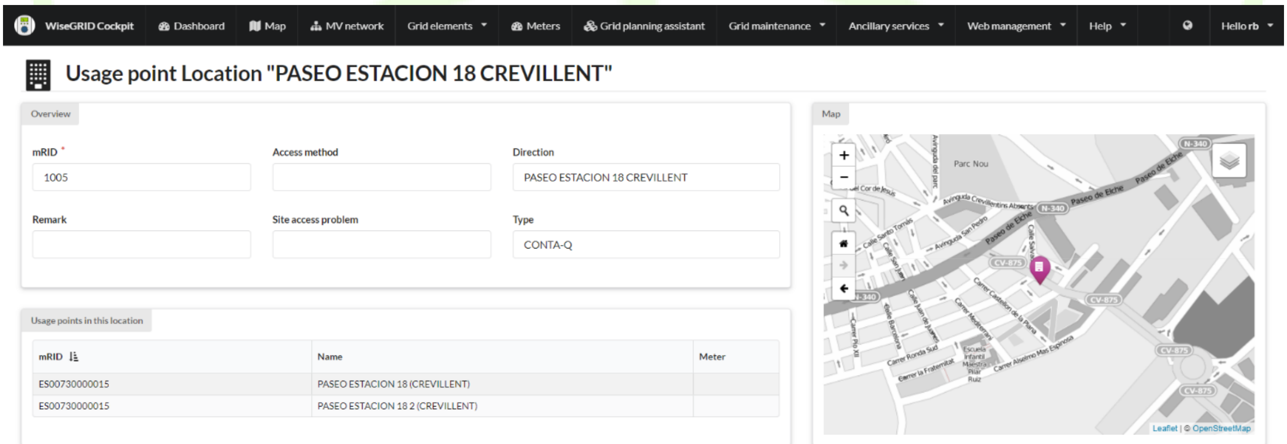



**Figure 49 – Usage point location details**

By clicking on any of the usage points, the UI shows the registered details for the usage point, accordingly to the CIM data model. Those details include:

- Overview: ID, nominal voltage, outage region, rated power…
- Location
- Contact details: electronic address, main and secondary phone details

- Associated meter

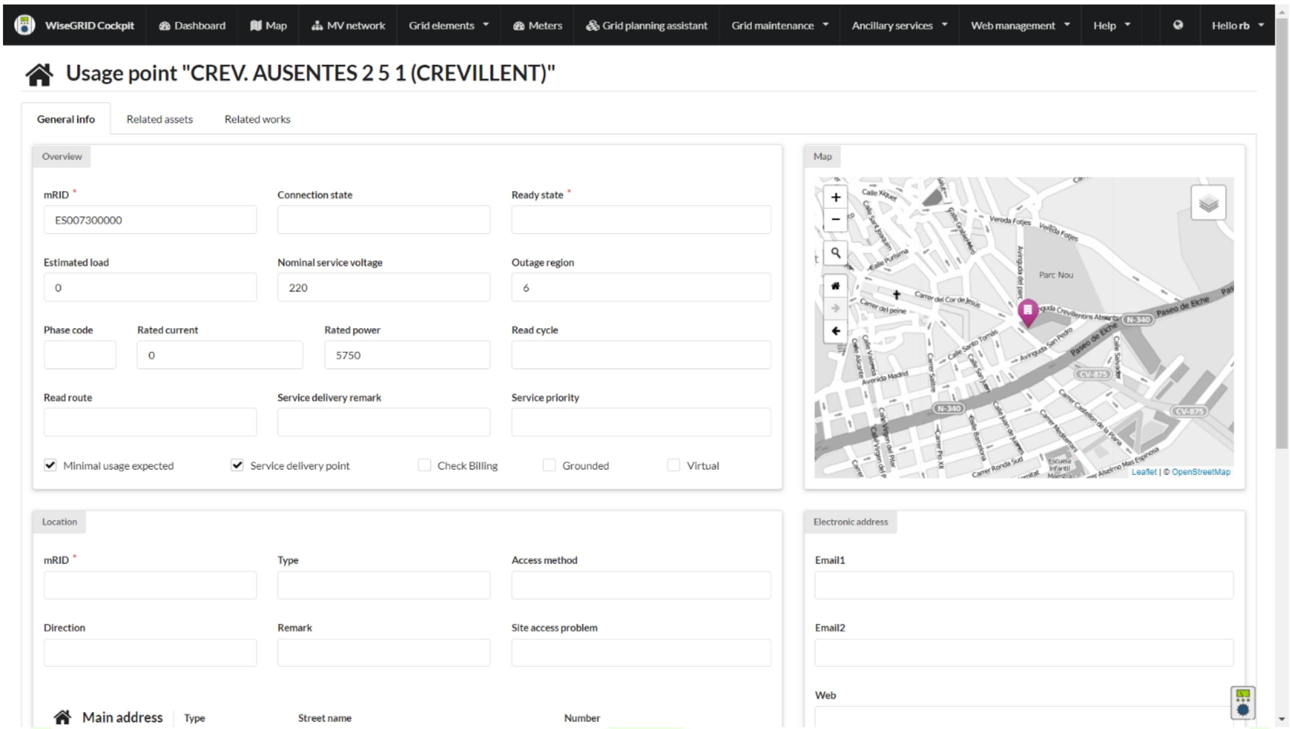
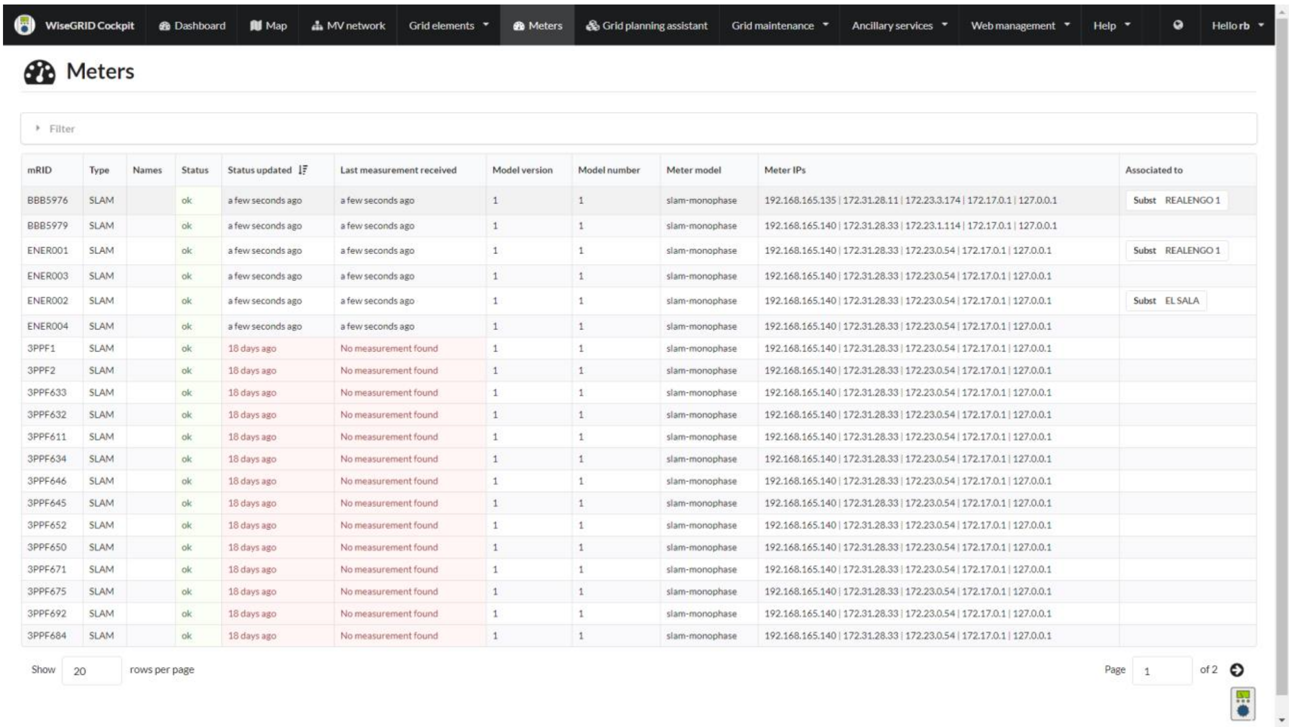



**Figure 50 – Usage point details**

### 3.3.7 Meters

The meters site provides access to an overview of the sensors deployed in the grid and monitored by the WiseGRID Cockpit. These sensors can be divided in different categories:

- Real sensors: SMX or other sensors (such as those integrated with the SCADA system) that provide real-time information to the WiseGRID Cockpit
- Virtual sensors: WiseGRID Cockpit includes a State Estimation module, with the purpose of increasing the observability of the grid by periodically executing the state estimation analysis to provide relevant metrics in points with no measuring equipment installed. Those results are presented in the form of virtual meters, which can be attached to the relevant assets of the grid (buses or lines). Their results are therefore seamlessly integrated in the UI and other modules of the WiseGRID Cockpit requiring this data




| mRID | Type | Names | Status | Status updated | Last measurement received | Model version | Model number | Meter model | Meter IPs | Associated to |
|---|---|---|---|---|---|---|---|---|---|---|
| BBB5976 | SLAM | | ok | a few seconds ago | a few seconds ago | 1 | 1 | slam-monophase | 192.168.165.135 \| 172.31.28.11 \| 172.23.3.174 \| 172.17.0.1 \| 127.0.0.1 | Subst REALENGO 1 |
| BBB5979 | SLAM | | ok | a few seconds ago | a few seconds ago | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.1.114 \| 172.17.0.1 \| 127.0.0.1 | |
| ENER001 | SLAM | | ok | a few seconds ago | a few seconds ago | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | Subst REALENGO 1 |
| ENER003 | SLAM | | ok | a few seconds ago | a few seconds ago | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| ENER002 | SLAM | | ok | a few seconds ago | a few seconds ago | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | Subst EL SALA |
| ENER004 | SLAM | | ok | a few seconds ago | a few seconds ago | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF1 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF2 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF633 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF632 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF611 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF634 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF646 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF645 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF652 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF650 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF671 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF675 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF692 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF684 | SLAM | | ok | 18 days ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |

**Figure 51 – Meters**

By clicking in any of the meters, a dialog is shown displaying further details accordingly to the CIM data model:

- Current status
- General data (ID, AMR system, smart meter type, serial number…)
- Asset model (rated voltage, rated current, model version…)
- Electronic address
- Seals

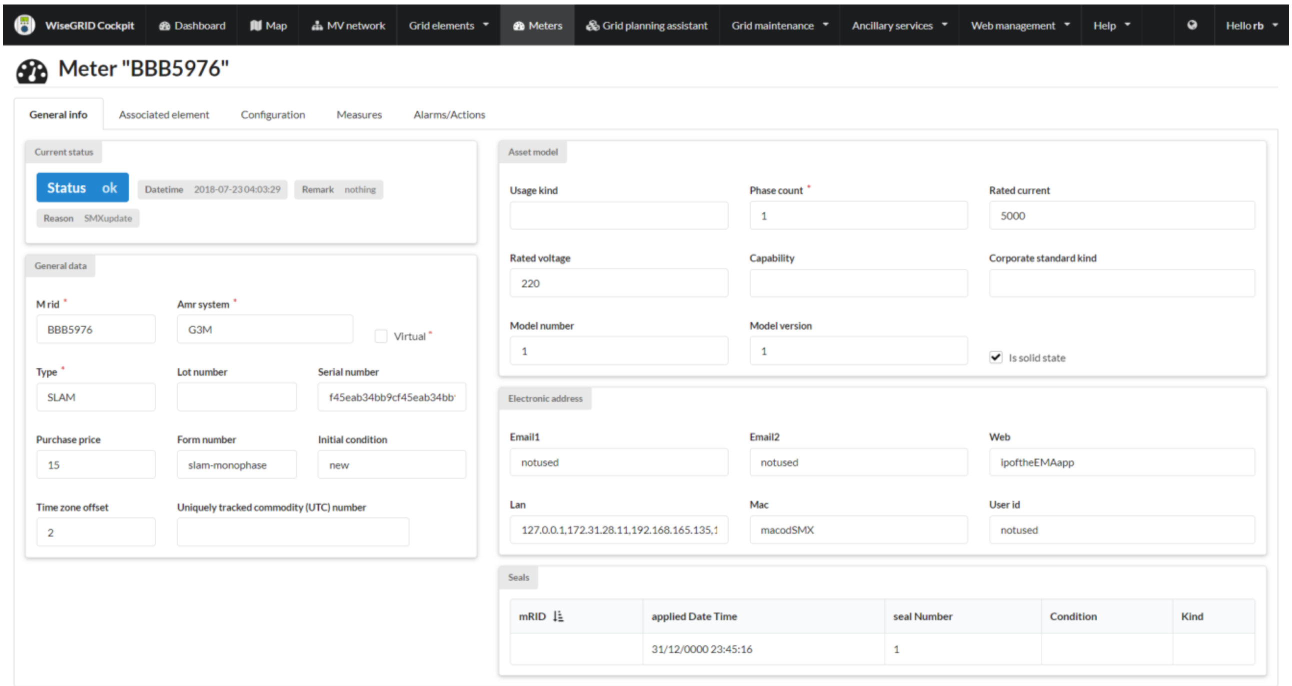



Figure 52 – Meters detail

The associated element tab allows to bind the selected meter to any of the assets defined in the WiseGRID Cockpit (line, bus or Renewable Energy Source). Once the association has been performed, the subsection displays further details on the element that is bound to the meter.

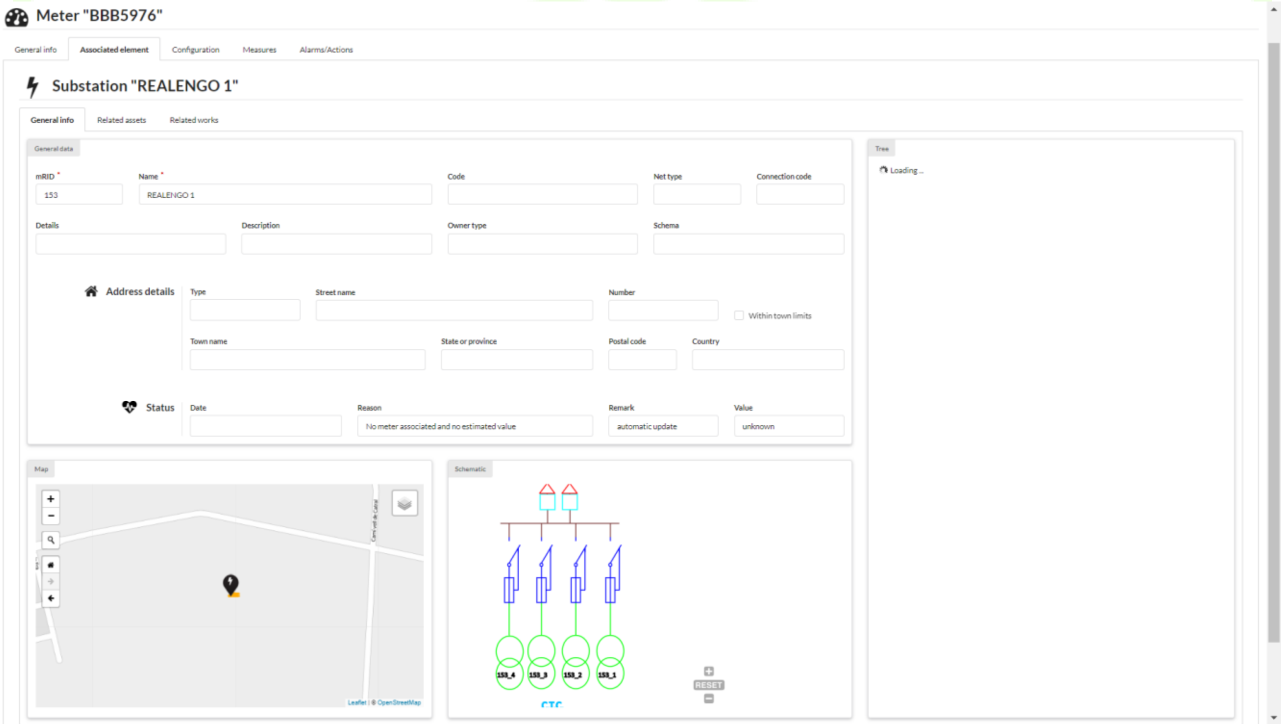



Figure 53 – Meter association to a substation

By selecting the measures tab, access is given to the historical data retrieved from the meter (instant metrics over last hour, hourly aggregations over last day and daily aggregations over last month).

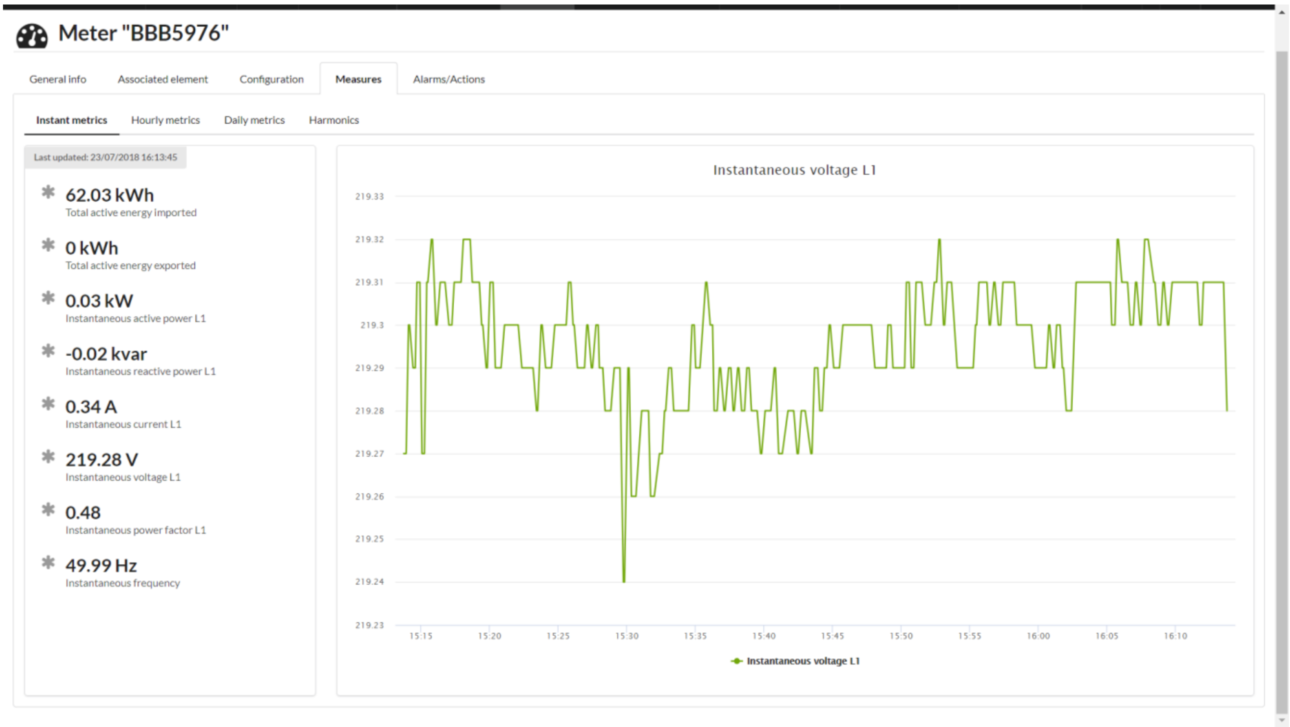



**Figure 54 – Meter measures**

### 3.3.8 Grid planning assistant

The grid planning assistant section allows the DSO operator to define and simulate different scenarios, allowing for instance to assess the impact in the grid of plausible scenarios of extra demand (e.g. simulating a certain penetration ratio of Electric Vehicles), extra production (e.g. simulating effect of a PV panel installation in a certain location of the grid) or installation of DSO-controlled storage systems. Additionally, flexibility provided by different aggregators can also be considered in the simulations, thus allowing to assess the role that demand-response campaigns could potentially have to solve certain issues on the grid.

The first subsection allows to define all the parameters of the simulation, namely:

- Generation per node
- Load per node
- Flexibility provided by aggregators on generation in specific areas
- Flexibility provided by aggregators on load in specific areas
- Controllable wind units: capacity, equivalent hours, price, min/max (re)active power
- Controllable PV units: capacity, equivalent hours, price, min/max (re)active power
- Controllable hydro units: capacity, equivalent hours, price, min/max (re)active power
- Controllable CHP units: ramp up, ramp down, min/max (re)active power
- Controllable storage units: capacity, max. (dis)charging power, (dis)charging efficiency, (dis)charging price

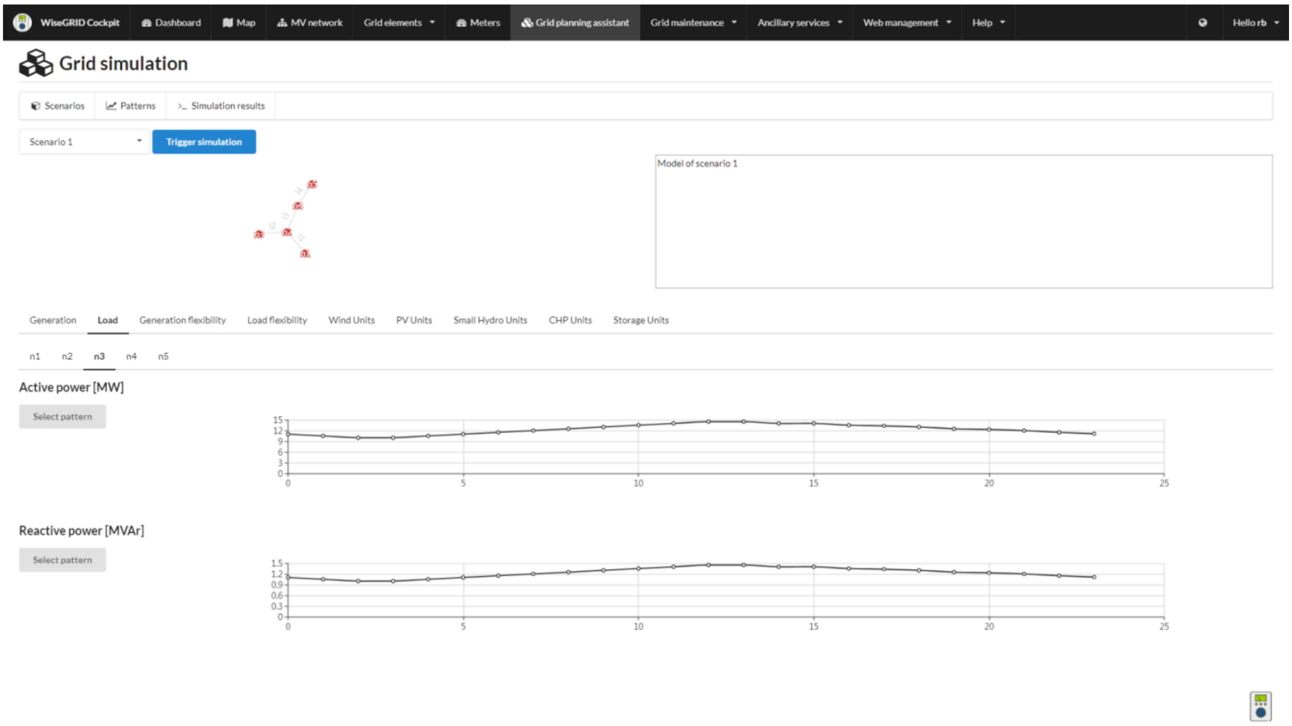



Figure 55 – Grid simulation, scenario definition

In order to facilitate the modelling of recurring scenarios, the concept of patterns is introduced. The DSO operator can insert different patterns for (re)active power or flexibility that can be easily reused among different scenarios.

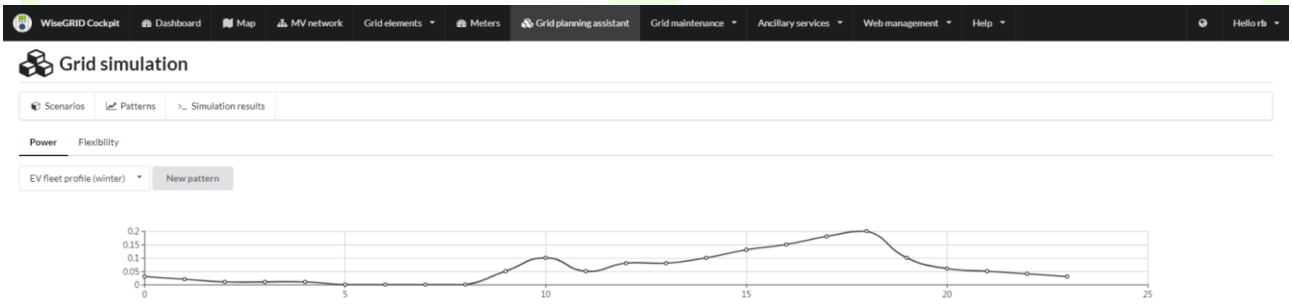



Figure 56 – Grid simulation, patterns

Finally, once the simulation is executed, the results are displayed under the Simulation results tab. The results of the simulation include:

- Estimated values for voltages per node and time step
- Estimated (re)active power flows in the lines of the grid per time step
- Summary with the identified operation limits that are surpassed (voltage or power flow)

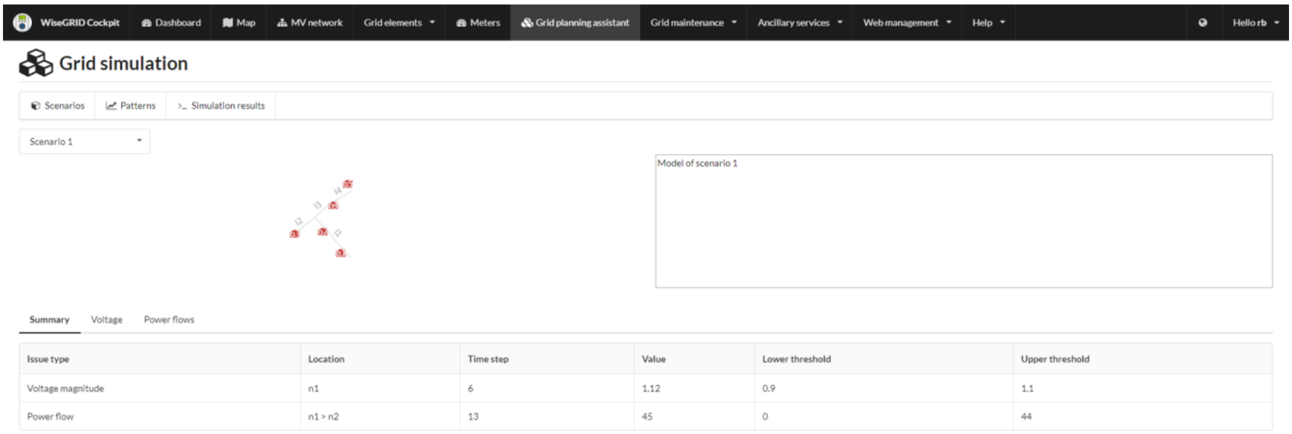



Figure 57 – Grid simulation, results

### 3.3.9 Self healing

This section displays information coming from the FLISR module, with regards to the reconfiguration instructions to be followed after a problem in the grid – a change in the status of a Switch-disconnector - is detected. The FLISR module gets automatically triggered in order to calculate the optimum set of steps to reestablish service in an optimum manner.

The section presents a list with the events detected in the grid that required the execution of the FLISR module, ordered by occurrence timestamp.

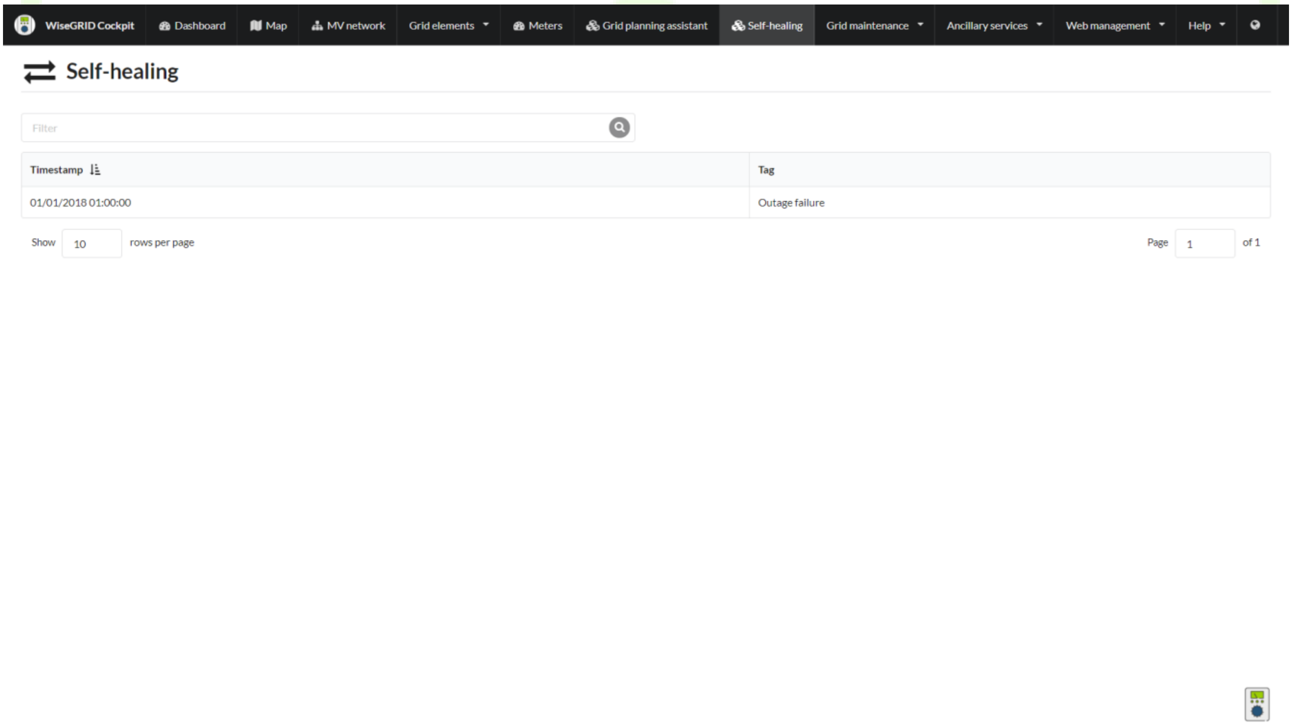



Figure 58 – Self-healing, list of events

By selecting an event, the results of the FLISR module are displayed in an understandable way to the DSO operator. These results include the optimum list of steps – as calculated by the FLISR module – to be carried out in order to re-establish the service with the minimum possible impact on the customers. Each one of the steps represents an operation of opening or closing a switch, which may be performed manually or automatically if possible.

By selecting one of the steps, further assistance is provided by:

- Displaying the foreseen status and the affected link in the schematic view of the distribution grid. The affected link is represented with a red blinking arrow. If the switch is meant to be closed, a continuous arrow is displayed. A dashed arrow represents a switch meant to be open. The other arrows represent the expected status of the links (switches) when the selected step is reached.
- Displaying the geographical location of the switch to be operated in the map, assisting the DSO operator to plan the operations to re-establish the service

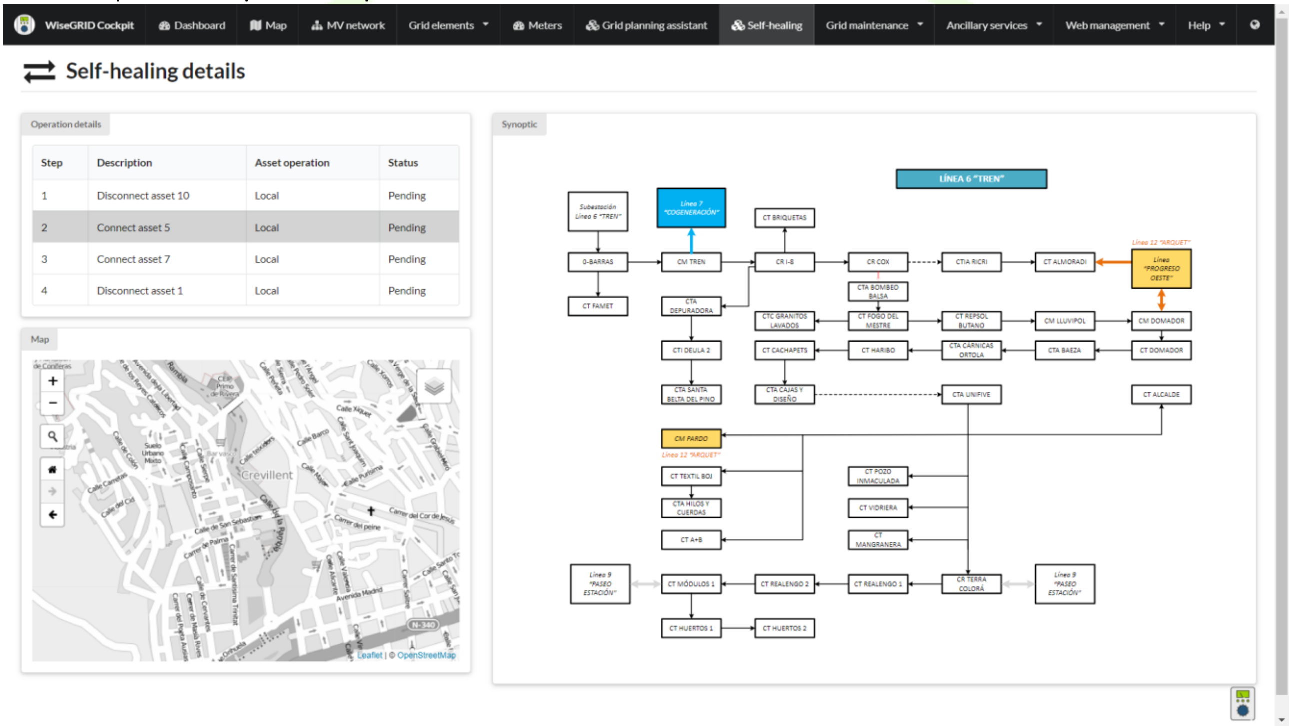



**Figure 59 – Self-healing, details on reconnection procedure**

### 3.3.10 Ancillary Services Market

This section displays information on the active and finished campaigns initiated by the DSO, providing the following details:

- Congestion point where the demand response campaign is triggered
- Starting timestamp
- End timestamp
- Required flexibility: requested power reduction (or increase) in the area of the congestion point for the given period of time
- Status: current status of the demand-response campaign (*Request posted*, *Offer sent*, *Order poster*, *DR campaign started*, *DR campaign finished*)

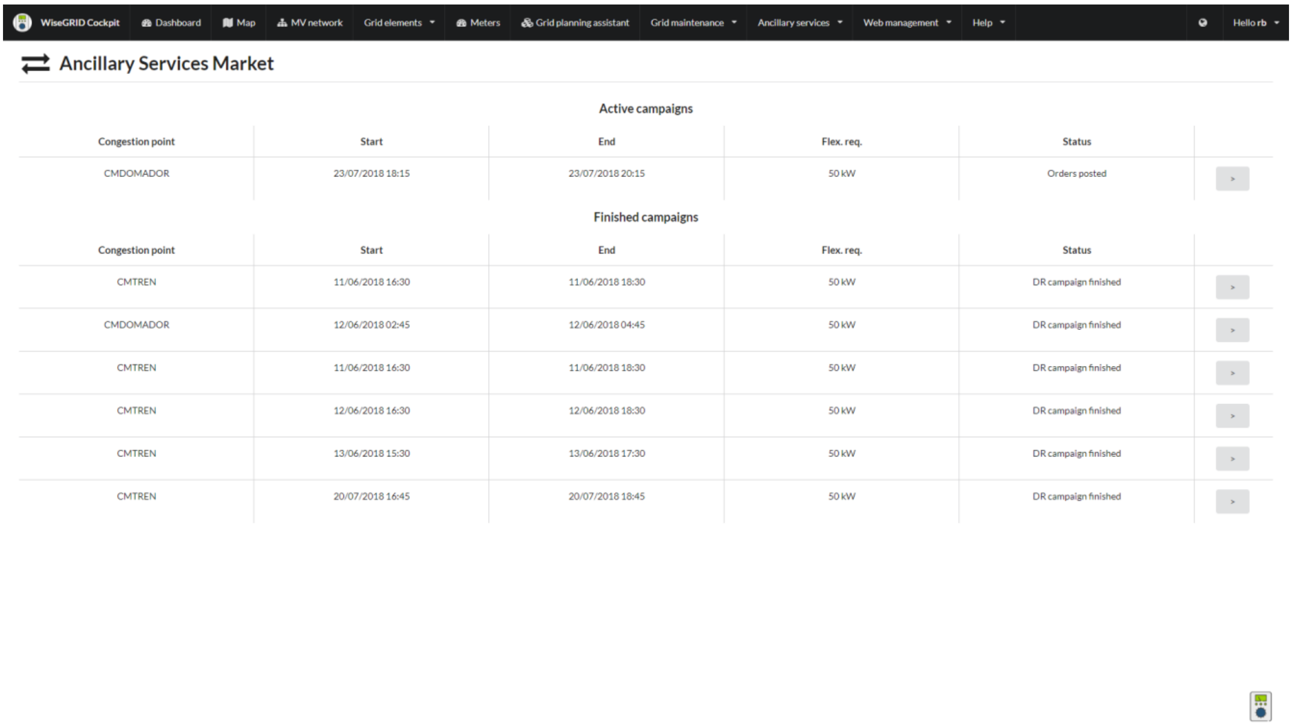



**Figure 60 – List of active and finished explicit demand response campaigns**

By selecting one of the campaigns, the corresponding details are displayed, including:

- Starting timestamp
- Duration of the campaign
- Required flexibility
- History of events for this campaign: shows all events that happened in the flexibility market related to this campaign.

**Table 24 – List of possible status for explicit demand response campaigns**

| | Timestamp | Status | Sender | Recipient | Flexibility | Price |
|---|---|---|---|---|---|---|
| **Flex. request** | Indicates when the request was posted by DSO | Request posted | DSO Operator | | Indicates the amount of flexibility requested | |
| **Offer sent** | Indicates when the offer was sent to the DSO | Offer sent | Name of the aggregator | DSO Operator | Indicates the amount of flexibility offered | Indicates the price requested for the offered flexibility |
| **Order posted** | Indicates when the order was posted by DSO | Order posted | DSO Operator | Name of the selected aggregator | Indicates the amount of flexibility ordered (the amount offered by the aggregator) | Indicates the price agreed with the aggregator (the price requested by the aggregator) |

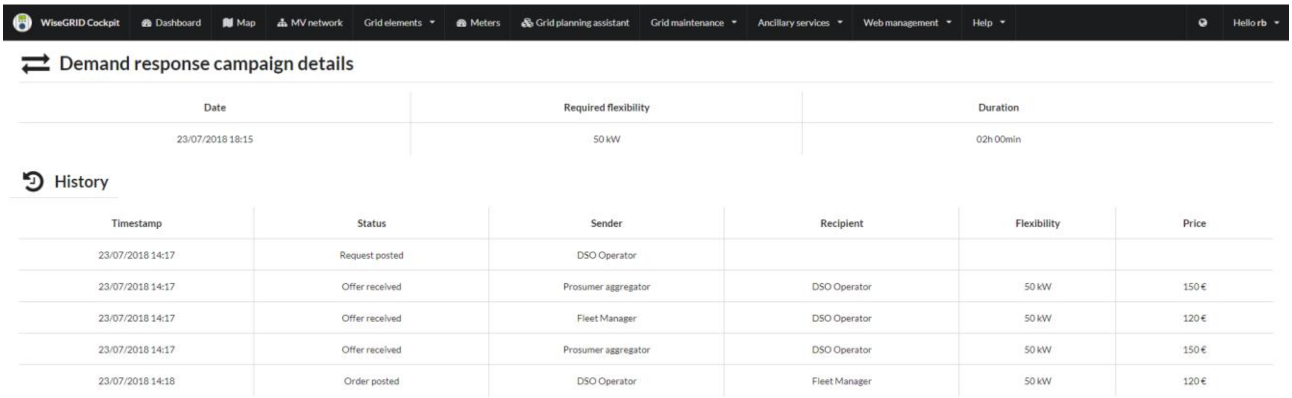



**Figure 61 – Details of an explicit demand response campaigns**

# 4 LAB-TESTING RESULTS

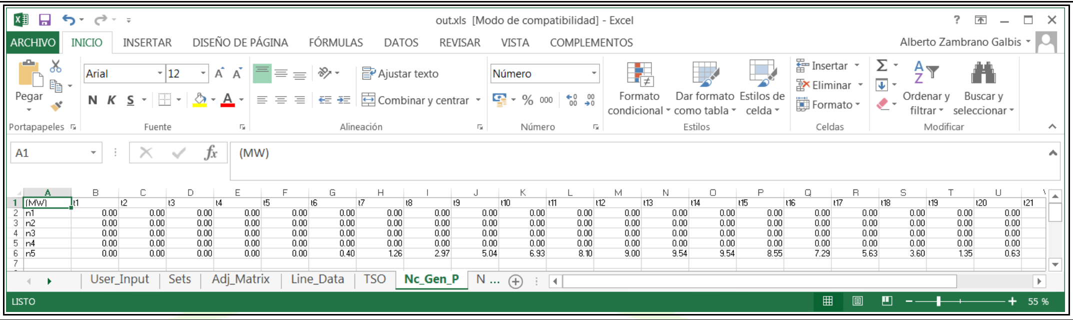
This section contains a set of templates with the definition, objectives, steps and results of all tests executed during this period on the different modules of the tool.

## 4.1 GRID PLANNING ASSISTANT

| | | | |
|---|---|---|---|
| **Name** | GPA001. Import topology from CIM model | | |
| **Module under test** | Grid planning assistant | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_4_Grid planning analysis | | |
| **Test environment** | CIM Topology model of a grid<br>SPARQL server hosting the model up and running<br>Internal ESB up and running | | |
| **Features to be tested** | Static data needed for the simulation (buses, lines, adjacency matrix) can be extracted from topology model | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | | | |
| **Steps** | 1. Create a new scenario simulation | | |
| **Pass criteria** | UI correctly displays:<br>- Structure of the grid<br>- Static properties of the buses<br>- Static properties of the lines | | |
| **Suspension criteria** | | | |
| **Results** | Test successful. | | |

| | |
|---|---|
| | Necessary information for the scenario simulation can be successfully imported by querying the CIM topology model |

| | | | |
|---|---|---|---|
| **Name** | GPA002. Simulation model is composed in the UI | | |
| **Module under test** | Grid planning assistant | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_4_Grid planning analysis | | |
| **Test environment** | CIM Topology model of a grid<br>SPARQL server hosting the model up and running<br>Internal ESB up and running | | |
| **Features to be tested** | Excel file required to simulate the scenario can be successfully created from the UI | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | GPA001. Import topology from CIM model | | |
| **Steps** | 1. Complete the definition of the scenario to be simulated<br>2. Trigger the simulation<br>3. Manually inspect the Excel file created in temporary folder | | |
| **Pass criteria** | Excel file contains all required information in the correct format:<br>- Static properties of the buses<br>- Static properties of the lines<br>- Adjacency matrix<br>- Generation and load P/Q per bus and time step<br>- Flexibility curves and prices per bus and time step<br>- Information related to RES (Wind, PV, Hydro, CHP)<br>- Information related to storage | | |
| **Suspension criteria** | | | |
| **Results** | Test successful.<br>After introducing all values in the UI and selecting the “Trigger simulation” button, an Excel file with the format expected by the GAMS simulation module is pushed to the internal ESB. | | |

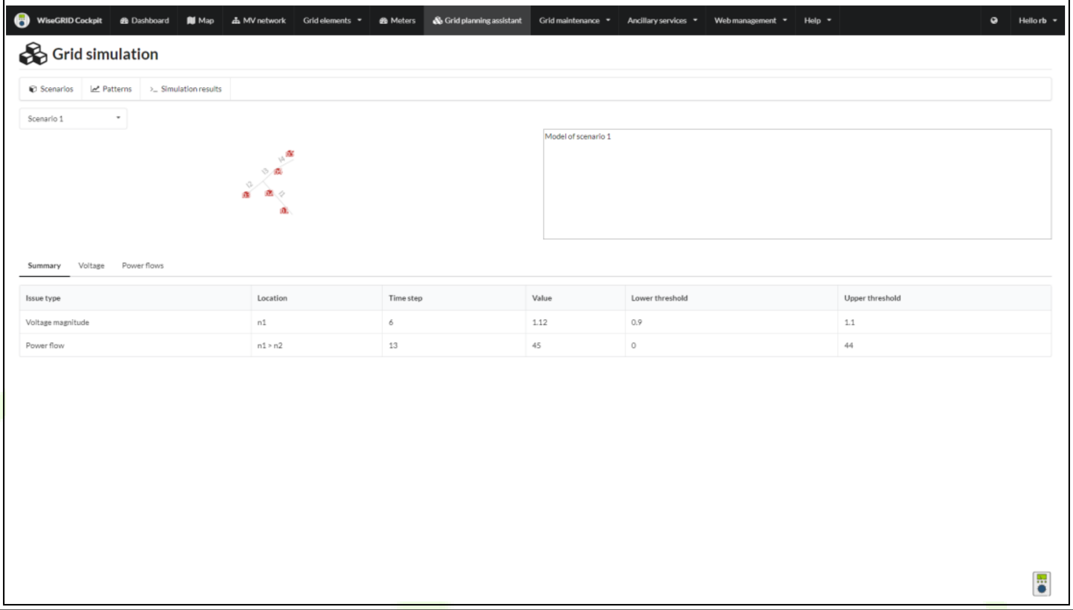
| | |
|---|---|
| |  |

| | | | |
|---|---|---|---|
| **Name** | GPA003. Server receives simulation request from internal ESB. | | |
| **Module under test** | Grid planning assistant | **Resp.** | ICCS |
| **Module requirement** | HL-UC 1_PUC_4_Grid planning analysis | | |
| **Test environment** | SPARQL server hosting the model up and running<br>Internal ESB up and running<br>Server hosting the GAMS solver up and running | | |
| **Features to be tested** | GAMS solver successfully loads the input file (.xlsx format). | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | GPA002. Simulation model is composed in the UI. | | |
| **Steps** | 1. Simulation is triggered.<br>2. Server receives the input file.<br>3. GAMS solver is triggered.<br>4. GAMS loads input file automatically without error. | | |
| **Pass criteria** | GAMS is triggered and loads the input file automatically. | | |
| **Suspension criteria** | | | |
| **Results** | Test successful | | |

| | | | |
|---|---|---|---|
| **Name** | GPA004. Server computes and delivers simulation results to internal ESB | | |
| **Module under test** | Grid planning assistant | **Resp.** | ICCS |
| **Module requirement** | HL-UC 1_PUC_4_Grid planning analysis | | |
| **Test environment** | Internal ESB up and running<br>Server hosting the GAMS solver up and running | | |

| | |
|---|---|
| **Features to be tested** | GAMS solver successfully runs, produces results and delivers them to internal ESB. |
| **Features not to be tested** | |
| **Preparation** | |
| **Dependencies** | GPA003. Server receives simulation request from internal ESB. |
| **Steps** | 1. GAMS solver is executed.<br>2. GAMS produces the results and stores them in Excel file.<br>3. Manually evaluate the results.<br>4. Output file is delivered to internal ESB. |
| **Pass criteria** | GAMS produces output file and delivers it to internal ESB. |
| **Suspension criteria** | |
| **Results** | Test successful |

| | | | |
|---|---|---|---|
| **Name** | GPA005. Simulation results are shown in the UI | | |
| **Module under test** | Grid planning assistant | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_4_Grid planning analysis | | |
| **Test environment** | CIM Topology model of a grid<br>SPARQL server hosting the model up and running<br>Internal ESB up and running | | |
| **Features to be tested** | Response of the GAMS simulation is properly shown in the UI | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | GPA004. Server computes and delivers simulation results to internal ESB | | |
| **Steps** | The execution of this test must happen automatically upon delivery of the results in the internal ESB | | |
| **Pass criteria** | UI correctly displays:<br>- Summary of issues<br>- Voltage curves<br>- Power flow curves | | |
| **Suspension criteria** | | | |
| **Results** | Test successful<br>Upon reception of an Excel file through the internal ESB, its contents are correctly processed and stored in the operational database, which allows them to be displayed in | | |

| | |
|---|---|
| | the UI   |

## 4.2 THERMAL AND RES PLANNING

| | | | |
|---|---|---|---|
| **Name** | TRP001. Static grid data is retrieved from structural database. | | |
| **Module under test** | Thermal and RES planning | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_4_Grid planning analysis. | | |
| **Test environment** | Structural database up and running.<br>Internal ESB up and running. | | |
| **Features to be tested** | Required data from structural database is collected and request by ESB is published. | | |
| **Features not to be tested** | | | |
| **Preparation** | This module requires a wide range of data concerning grid and asset characteristics. It is necessary to have filled the database with all the required data. | | |
| **Dependencies** | | | |
| **Steps** | 1. Ensure data availability in structural database.<br>2. RabbitMQ collects required data from database and fills the input file.<br>3. RabbitMQ publishes a request. | | |
| **Pass criteria** | Static data of the input file have been correctly gathered and placed in the input file. Manual cross checking is necessary. Response has been published correctly. | | |

| | |
|---|---|
| **Suspension criteria** | |
| **Results** | *Pending* |

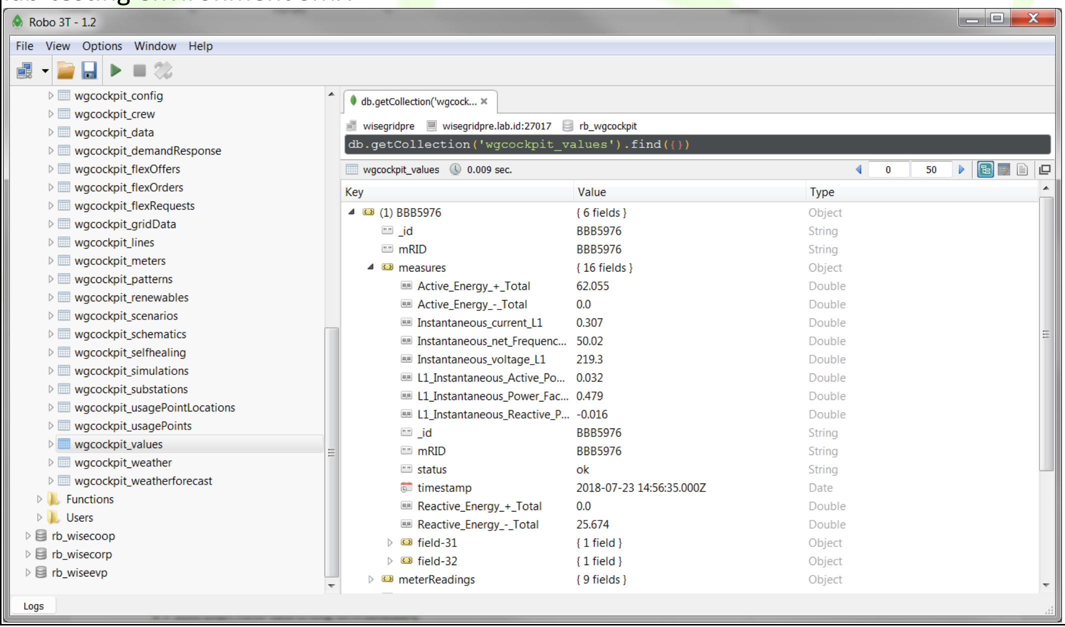
| | | | |
|---|---|---|---|
| **Name** | TRP002. Data is retrieved from long-term database. | | |
| **Module under test** | Thermal and RES planning | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_4_Grid planning analysis. | | |
| **Test environment** | Long-term database up and running.<br>Internal ESB up and running. | | |
| **Features to be tested** | Required data from long-term database are collected and request by ESB is published. | | |
| **Features not to be tested** | | | |
| **Preparation** | The submodule requires historical data (measurements) from the long-term database. This data should be available for the evaluation of this functionality. | | |
| **Dependencies** | | | |
| **Steps** | 1. Ensure data availability in long-term database.<br>2. RabbitMQ collects required data from database and fills the input file of the submodule.<br>3. RabbitMQ publishes a request. | | |
| **Pass criteria** | Data of the input file have been correctly gathered and placed in the input file. Manual cross checking is necessary. Response has been published correctly. | | |
| **Suspension criteria** | | | |
| **Results** | *Pending* | | |

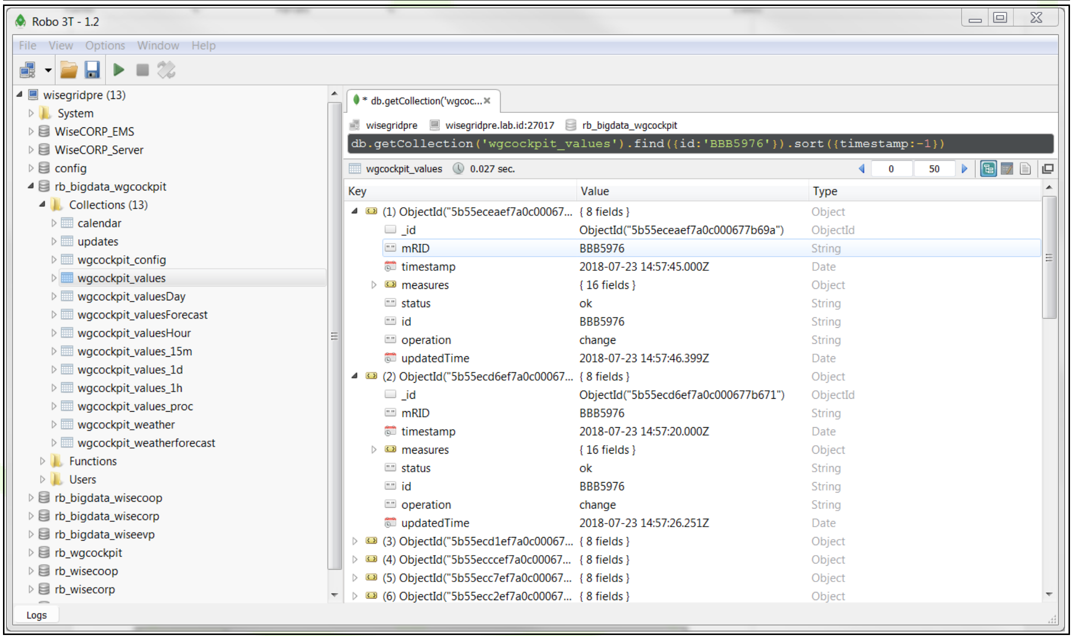
| | | | |
|---|---|---|---|
| **Name** | TRP003. Server receives requests from internal ESB. | | |
| **Module under test** | Thermal and RES planning | **Resp.** | ICCS |
| **Module requirement** | HL-UC 1_PUC_4_Grid planning analysis. | | |
| **Test environment** | Internal ESB up and running<br>Server hosting the module Thermal and RES planning and the submodule Demand/Peak prediction up and running. | | |

| | |
|---|---|
| **Features to be tested** | Module and submodule successfully receive requests and the corresponding input files. |
| **Features not to be tested** | |
| **Preparation** | |
| **Dependencies** | TRP001. Static grid data is retrieved from structural database.<br>TRP002. Data is retrieved from long-term database. |
| **Steps** | 1. Internal ESB requests Thermal and RES planning.<br>2. Server responds to requests and receives input files.<br>3. Module and submodule input files are received with no errors. |
| **Pass criteria** | Module and submodule are triggered and load the input files automatically. |
| **Suspension criteria** | |
| **Results** | *Pending* |

| | | | |
|---|---|---|---|
| **Name** | TRP004. Server computes and delivers results to internal ESB. | | |
| **Module under test** | Thermal and RES planning | **Resp.** | ICCS |
| **Module requirement** | HL-UC 1_PUC_4_Grid planning analysis. | | |
| **Test environment** | Internal ESB up and running.<br>Server hosting the module Thermal and RES planning up and running. | | |
| **Features to be tested** | Module successfully runs, produces results and delivers them to internal ESB. | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | TRP003. Server receives requests from internal ESB. | | |
| **Steps** | 1. Module TRP is executed.<br>2. Module TRP produces the results and stores them in output file.<br>3. Manually cross-check the results.<br>4. Output file is delivered to internal ESB. | | |
| **Pass criteria** | Module produces output file and delivers it to internal ESB successfully. | | |
| **Suspension criteria** | | | |
| **Results** | *Pending* | | |

## 4.3 RT MONITOR

| | | | |
|---|---|---|---|
| **Name** | RTM001. Read smart meter data from IOP | | |
| **Module under test** | RT Monitor | **Resp.** | ETRA |
| **Module requirement** | HL-UC 2_PUC_1_Distribution network real-time monitoring | | |
| **Test environment** | SMX running and sending information to IOP<br>IOP platform up and running | | |
| **Features to be tested** | Data from SMX is properly collected in the operational database of WiseGRID Cockpit | | |
| **Features not to be tested** | | | |
| **Preparation** | Configure one SMX to send data to the lab-testing IOP environment | | |
| **Dependencies** | | | |
| **Steps** | The execution of this test must happen automatically upon publication of data in the IOP | | |
| **Pass criteria** | Data from the SMX is correctly updated in the operational database. Operational database keeps a register of the last values sent by the SMX. | | |
| **Suspension criteria** | | | |
| **Results** | Test successful.<br>The following screenshot shows how the operational DB is populated with data from lab-testing environment SMX<br> | | |

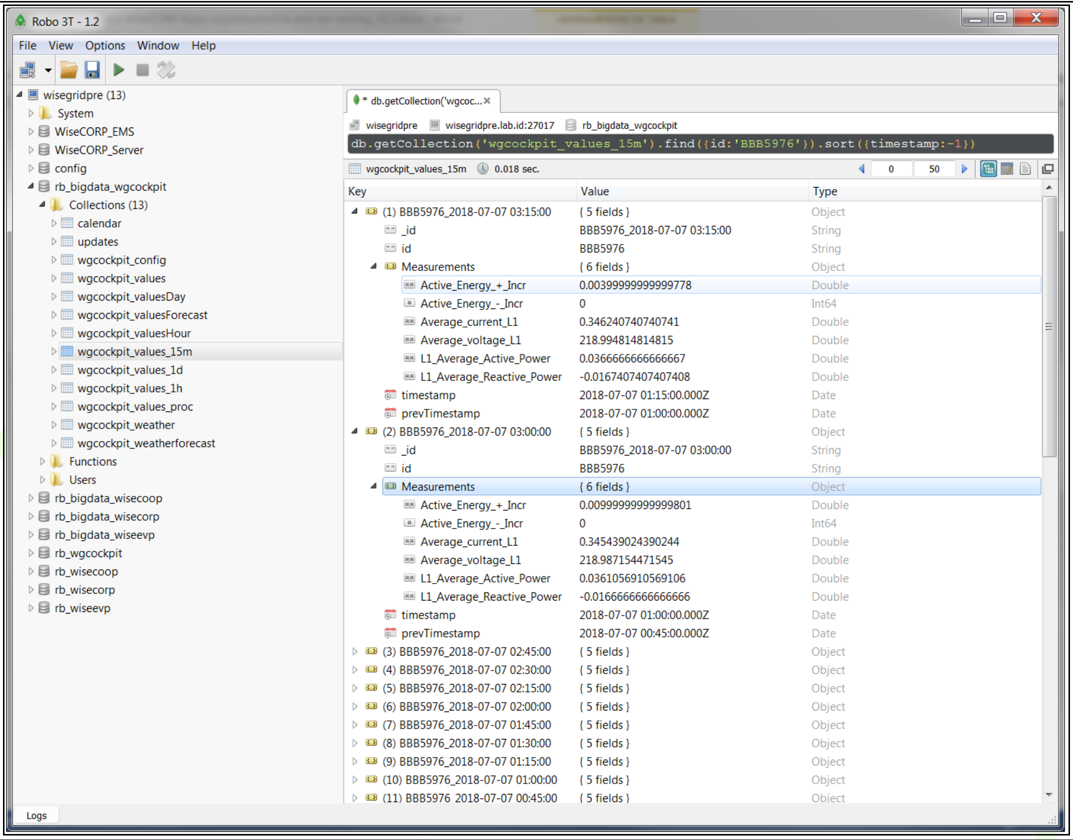
| | | | |
|---|---|---|---|
| **Name** | RTM002. Store smart meter data to Long-term DB | | |
| **Module under test** | RT Monitor | **Resp.** | ETRA |
| **Module requirement** | HL-UC 2_PUC_1_Distribution network real-time monitoring | | |
| **Test environment** | SMX running and sending information to IOP<br>IOP platform up and running | | |
| **Features to be tested** | Data from SMX is properly collected in the long-term database of WiseGRID Cockpit (big data) | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | RTM001. Read smart meter data from IOP | | |
| **Steps** | The execution of this test must happen automatically upon publication of data in the IOP | | |
| **Pass criteria** | Data from the SMX is correctly appended to the historic registry held in the long-term database | | |
| **Suspension criteria** | | | |
| **Results** |  | | |



## 4.4 KPI ENGINE

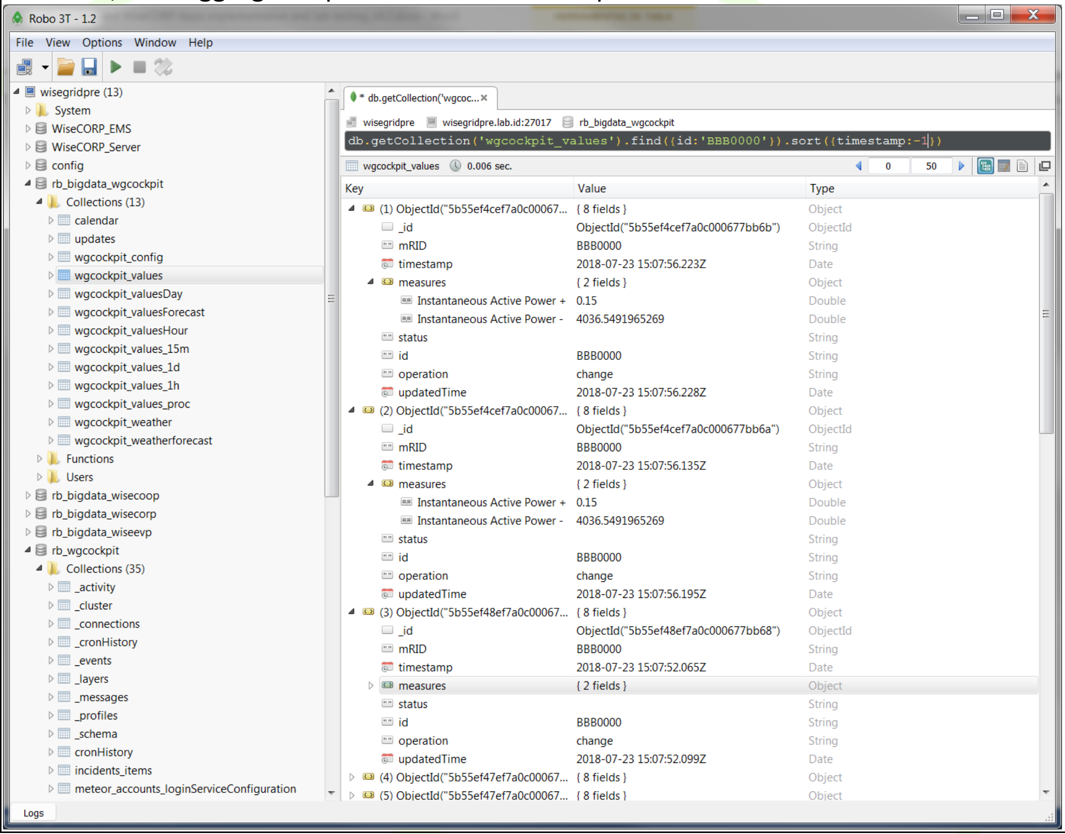
| | |
|---|---|
| **Name** | KPI001. Energy delta calculation |

| | | | |
|---|---|---|---|
| **Module under test** | KPI engine | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_1_Network monitoring | | |
| **Test environment** | SMX running and sending information to IOP<br>IOP platform up and running<br>RT monitor storing smart meter readings in long-term database<br>Spark server up and running | | |
| **Features to be tested** | Smart meters provide information of the total accumulated energy demand/production. The system therefore needs to calculate the energy deltas across consecutive readings in order to properly monitor the energy demand/production profiles.<br>Three different aggregation of the deltas are considered: quarterly, hourly and daily. | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | | | |
| **Steps** | 1. Execute Spark job for quarterly delta calculation<br>2. Execute Spark job for hourly delta calculation<br>3. Execute Spark job for daily delta calculation<br>4. Manually inspect long-term database collections with processed data | | |
| **Pass criteria** | - The long-term database contains 3 collections with the quarterly, hourly and daily aggregations<br>- Each collection contains documents that represent the energy deltas for the given period | | |
| **Suspension criteria** | | | |
| **Results** | Test successful.<br>The following screenshot shows the three created collections (wgcockpit_values_15m\|1h\|1d) as well as an example of the active and reactive energy deltas calculated. | | |

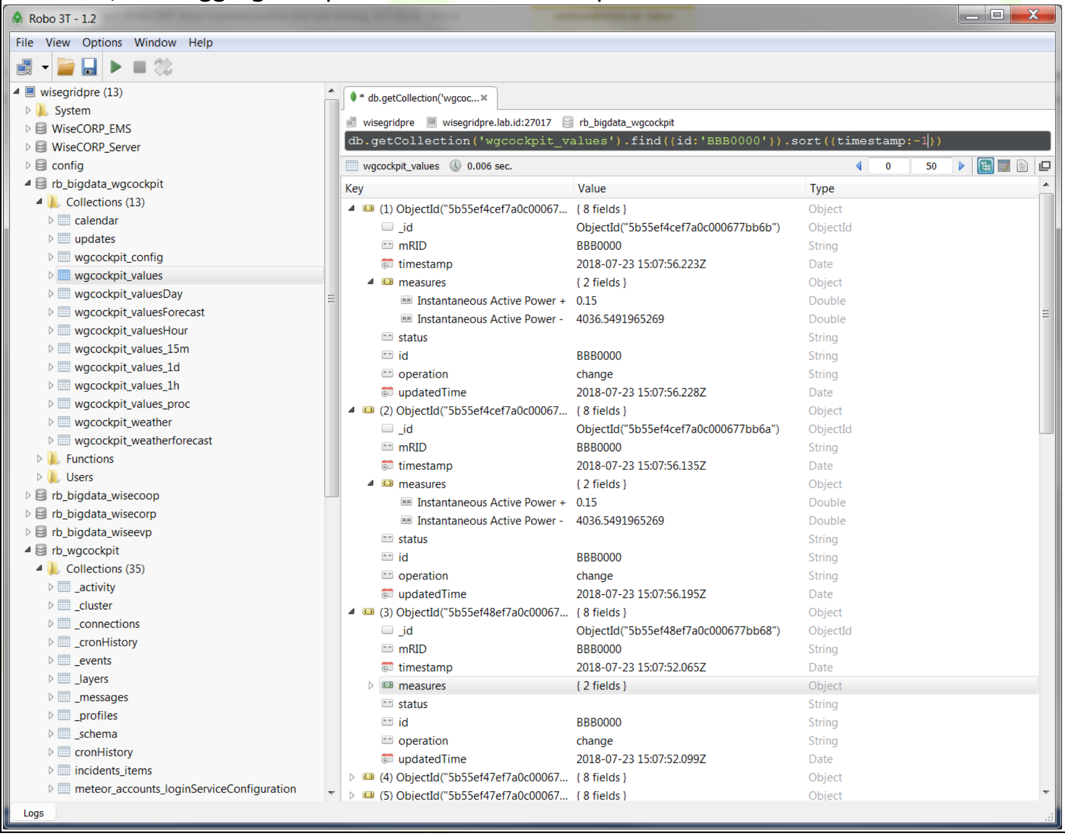



| | | | |
|---|---|---|---|
| **Name** | KPI002. Aggregated production | | |
| **Module under test** | KPI engine | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_1_Network monitoring | | |
| **Test environment** | SMX running and sending information to IOP<br>IOP platform up and running<br>RT monitor storing smart meter readings in long-term database<br>Spark server up and running | | |
| **Features to be tested** | Smart meters linked to buses of the grid with big RES installations provide information of the total production of the RES installation. There is a need to aggregate all of those in order to get an overview of the total energy production in the grid | | |
| **Features not to be tested** | | | |
| **Preparation** | Associate one smart meter to each one of the buses with production | | |
| **Dependencies** | | | |
| **Steps** | 1. Enable *Aggregation* characteristic to *Surveillance* module<br>2. Execute *Surveillance* module | | |

| | |
|---|---|
| | 3. Manually inspect long-term database collections, looking for the special Virtual Meter BBB0000 |
| **Pass criteria** | A new Virtual Meter appears in the database, whose readings on production data equal to the aggregation of all production identified in the grid. |
| **Suspension criteria** | |
| **Results** | Test successful<br>The long-term database contains information about the special virtual smart meter BBB0000, with aggregated power of demand and production |




| | | | |
|---|---|---|---|
| **Name** | KPI003. Aggregated demand | | |
| **Module under test** | KPI engine | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_1_Network monitoring | | |
| **Test environment** | SMX running and sending information to IOP<br>IOP platform up and running<br>RT monitor storing smart meter readings in long-term database<br>Spark server up and running | | |
| **Features to be** | Smart meters linked to buses of the grid provide information of the total demand of the | | |

| | |
|---|---|
| **tested** | loads connected to them. There is a need to aggregate all of those in order to get an overview of the total energy demand in the grid |
| **Features not to be tested** | |
| **Preparation** | Associate one smart meter to each one of the buses with loads |
| **Dependencies** | |
| **Steps** | 1. Enable *Aggregation* characteristic to *Surveillance* module<br>2. Execute *Surveillance* module<br>3. Manually inspect long-term database collections, looking for the special Virtual Meter BBB0000 |
| **Pass criteria** | A new Virtual Meter appears in the database, whose readings on demand data equal to the aggregation of all loads identified in the grid. |
| **Suspension criteria** | |
| **Results** | Test successful<br>The long-term database contains information about the special virtual smart meter BBB0000, with aggregated power of demand and production<br> |



| | |
|---|---|
| **Name** | KPI004. Voltage deviation |

| | | | |
|---|---|---|---|
| **Module under test** | KPI engine | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_1_Network monitoring | | |
| **Test environment** | SMX running and sending information to IOP<br>IOP platform up and running<br>RT monitor storing smart meter readings in long-term database | | |
| **Features to be tested** | The module enables the early detection of significant deviations of voltage on a given bus. This is performed by automatically evaluating a control chart on the voltage magnitude of each phase per bus. Alerts are triggered whenever percentiles 1 or 99 are surpassed | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | | | |
| **Steps** | 1. Execute outlier detector module | | |
| **Pass criteria** | - The operational database contains information on the voltage magnitude percentiles 1, 50 and 99 per bus and phase<br>- Whenever percentiles 1 or 99 are surpassed, an incident of type “Voltage deviation” is opened in the incidents collection of the operational database | | |
| **Suspension criteria** | | | |
| **Results** | Test successful<br>The outlier detector keeps track of the voltage percentiles per bus. Simulations of significant voltage deviations result in a new item written to the incident collection of the operational database<br>{<br>"_id" : "BBB5976\|0.0.0.0.0.1.54.0.0.0.0.0.0.0.128.0.29.0",<br>"timestamp" : ISODate("2018-02-23T07:02:21.464Z"),<br>"q01" : 241.57,<br>"q50" : 242.53,<br>"q99" : 243.34,<br>"digest" : …<br>} | | |

| | |
|---|---|
| **Name** | KPI005. Frequency deviation |

| | | | |
|---|---|---|---|
| **Module under test** | KPI engine | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_1_Network monitoring | | |
| **Test environment** | SMX running and sending information to IOP<br>IOP platform up and running<br>RT monitor storing smart meter readings in long-term database | | |
| **Features to be tested** | The module enables the early detection of significant deviations of frequency on a given bus. This is performed by automatically evaluating a control chart on the frequency magnitude per bus. Alerts are triggered whenever percentiles 1 or 99 are surpassed | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | | | |
| **Steps** | 1. Execute outlier detector module | | |
| **Pass criteria** | - The operational database contains information on the voltage magnitude percentiles 1, 50 and 99 per bus and phase<br>- Whenever percentiles 1 or 99 are surpassed, an incident of type "Voltage deviation" is opened in the incidents collection of the operational database | | |
| **Suspension criteria** | | | |
| **Results** | Test successful<br>The outlier detector keeps track of the voltage percentiles per bus. Simulations of significant voltage deviations result in a new item written to the incident collection of the operational database<br>{<br>"_id" : "BBB5976\|0.0.0.12.0.1.15.0.0.0.0.0.0.0.224.0.33.0",<br>"timestamp" : ISODate("2018-02-23T07:02:21.461Z"),<br>"q01" : 51.32,<br>"q50" : 51.52,<br>"q99" : 51.69,<br>"digest" : …<br>} | | |

## 4.5 POWER FLOW CALCULATOR

| | | | |
|---|---|---|---|
| **Name** | PFM001. Import topology from CIM model | | |
| **Module under test** | 3 phase power flow module | **Resp.** | ETRA |
| **Module requirement** | HL-UC 2_PUC_2_Real-time distribution system awareness | | |
| **Test environment** | CIM Topology model of a grid<br>SPARQL server hosting the model up and running<br>Internal ESB up and running | | |
| **Features to be tested** | Static data needed for the power flow calculation (buses, lines, adjacency matrix) can be extracted from topology model | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | | | |
| **Steps** | Analysis of required inputs and crosscheck with CIM topology model | | |
| **Pass criteria** | Required inputs are found | | |
| **Suspension criteria** | | | |
| **Results** | Test successful<br>Part of the information required by the Power Flow Calculator is not contained in the CIM topology model: nominal voltage, system frequency, earth resistance, ground/connection/type per bus, status/connection type per branch, conductor list. The considered solution implies setting up this information in the operational database, as part of the commissioning of the WiseGRID Cockpit in the different pilot sites. | | |

| | | | |
|---|---|---|---|
| **Name** | PFM002. Measurements from buses are read from long-term database | | |
| **Module under test** | 3 phase power flow module | **Resp.** | ETRA |
| **Module requirement** | HL-UC 2_PUC_2_Real-time distribution system awareness | | |
| **Test environment** | CIM Topology model of a grid<br>SPARQL server hosting the model up and running<br>Internal ESB up and running<br>SMX running and sending information to IOP | | |

| | |
|---|---|
| | IOP platform up and running |
| **Features to be tested** | Real-time measurements required as an input for power flow are available in the long-term database |
| **Features not to be tested** | |
| **Preparation** | |
| **Dependencies** | |
| **Steps** | Analysis of required inputs and crosscheck with long-term database |
| **Pass criteria** | Required inputs are found |
| **Suspension criteria** | |
| **Results** | Test successful<br>The bus data section of the Excel file required by the Power Flow Calculation module can be read from the operational database. |

| | | | |
|---|---|---|---|
| **Name** | PFM003. Calculation request is read from internal ESB. | | |
| **Module under test** | Power flow calculator | **Resp.** | ICCS |
| **Module requirement** | HL-UC 2_PUC_2_Real-time distribution system awareness | | |
| **Test environment** | Internal ESB up and running<br>Server hosting power flow calculator up and running | | |
| **Features to be tested** | Power flow calculator receives input data file (.xlsx format) and algorithm is executed. | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | PFC002. Measurements are read from long-term database | | |
| **Steps** | 1. Internal ESB requests power flow calculations.<br>2. Server responds to request and receives input file.<br>3. Power flow algorithm is triggered and executed.<br>4. Algorithm stores results in an Excel file.<br>5. Manually cross check the results. | | |
| **Pass criteria** | Power flow algorithm is triggered, receives input data and executed with no errors. | | |
| **Suspension criteria** | | | |
| **Results** | Test successful | | |

| | | | |
|---|---|---|---|
| **Name** | PFM004. Results are delivered to internal ESB. | | |
| **Module under test** | Power flow calculator | **Resp.** | ICCS |
| **Module requirement** | HL-UC 2_PUC_2_Real-time distribution system awareness | | |
| **Test environment** | Internal ESB up and running<br>Server hosting power flow calculator up and running | | |
| **Features to be tested** | Server delivers to internal ESB the output file (.xlsx format) with the power flow results. | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | PFC003. Calculation request is read from internal ESB. | | |
| **Steps** | 1. Manually cross check the output file created by the power flow algorithm.<br>2. Server responds to initial request and sends the output file.<br>3. Internal ESB receives the output file uncorrupted. | | |
| **Pass criteria** | Internal ESB receives uncorrupted the output results of the power flow algorithm. | | |
| **Suspension criteria** | | | |
| **Results** | Test successful | | |

| | | | |
|---|---|---|---|
| **Name** | PFM005. Generation of results as virtual meters | | |
| **Module under test** | 3 phase power flow module | **Resp.** | ETRA |
| **Module requirement** | HL-UC 2_PUC_2_Real-time distribution system awareness | | |
| **Test environment** | CIM Topology model of a grid<br>SPARQL server hosting the model up and running<br>Internal ESB up and running<br>SMX running and sending information to IOP<br>IOP platform up and running<br>3PPF results available | | |
| **Features to be tested** | Results of the 3PPF module are published to the IOP in the form of virtual smart meters, becoming therefore available to the any other module of the WiseGRID Cockpit requiring them. | | |

| | |
|---|---|
| **Features not to be tested** | |
| **Preparation** | 3PPF algorithm implementation generates the results (Excel file) and pushes them to the internal ESB |
| **Dependencies** | |
| **Steps** | 1. Execute 3PPF orchestrator module |
| **Pass criteria** | - Excel file is received from internal ESB<br>- Results from Excel file are extracted<br>- New virtual smart meters (prefix 3PPF_) appear in the system, providing the following information<br>o Per bus: voltage magnitude and angle, current, active and reactive power<br>o Per line: current, active and reactive power, active and reactive power losses |
| **Suspension criteria** | |
| **Results** | Test successful<br>Upon reception of results, the Excel file is correctly analysed. Results are formatted to DLMS format and published to the IOP. Other modules of WiseGRID Cockpit see these results as if they were delivered by a special set of Smart Meters with 3PPF prefixed ID |

| mRID | Type | Names | Status | Status updated | Last measurement received | Model version | Model number | Meter model | Meter IPs | Associated to |
|---|---|---|---|---|---|---|---|---|---|---|
| 3PPF1 | SLAM | | no comm | 24 minutes ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF1_2 | SLAM | | no comm | 24 minutes ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF2 | SLAM | | no comm | 24 minutes ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF2_671 | SLAM | | no comm | 24 minutes ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF611 | SLAM | | no comm | 24 minutes ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |
| 3PPF632 | SLAM | | no comm | 24 minutes ago | No measurement found | 1 | 1 | slam-monophase | 192.168.165.140 \| 172.31.28.33 \| 172.23.0.54 \| 172.17.0.1 \| 127.0.0.1 | |

## 4.6 STATE ESTIMATOR

| | | | |
|---|---|---|---|
| **Name** | SE001. Import topology from CIM model | | |
| **Module under test** | State Estimation module | **Resp.** | ETRA |
| **Module requirement** | HL-UC 2_PUC_2_Real-time distribution system awareness | | |
| **Test environment** | CIM Topology model of a grid<br>SPARQL server hosting the model up and running<br>Internal ESB up and running | | |
| **Features to be tested** | Static data needed for the state estimation calculation (buses, lines, adjacency matrix) can be extracted from topology model | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | | | |

| | |
|---|---|
| **Steps** | Analysis of required inputs and crosscheck with CIM topology model |
| **Pass criteria** | Required inputs are found |
| **Suspension criteria** | |
| **Results** | Test successful<br>Part of the information required by the Power Flow Calculator is not contained in the CIM topology model: nominal voltage, system frequency, earth resistance, ground/connection/type per bus, status/connection type per branch, conductor list. The considered solution implies setting up this information in the operational database, as part of the commissioning of the WiseGRID Cockpit in the different pilot sites. |

| | | | |
|---|---|---|---|
| **Name** | SE002. Measurements from buses are read from long-term database | | |
| **Module under test** | State Estimation module | **Resp.** | ETRA |
| **Module requirement** | HL-UC 2_PUC_2_Real-time distribution system awareness | | |
| **Test environment** | CIM Topology model of a grid<br>SPARQL server hosting the model up and running<br>Internal ESB up and running<br>SMX running and sending information to IOP<br>IOP platform up and running | | |
| **Features to be tested** | Historic measurements required as an input for state estimation calculation are available in the long-term database | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | | | |
| **Steps** | Analysis of required inputs and crosscheck with long-term database | | |
| **Pass criteria** | Required inputs are found | | |
| **Suspension criteria** | | | |
| **Results** | Real-time measurements required as an input for power flow are available in the long-term database | | |

| | |
|---|---|
| **Name** | SE003. Calculation request is read from internal ESB. |

| | | | |
|---|---|---|---|
| **Module under test** | State Estimator | **Resp.** | ICCS |
| **Module requirement** | HL-UC 2_PUC_2_Real-time distribution system awareness | | |
| **Test environment** | Internal ESB up and running<br>Server hosting state estimator up and running | | |
| **Features to be tested** | State estimator receives input data file and algorithm is executed. | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | SE002. Measurements are read from long-term database. | | |
| **Steps** | 1. Internal ESB requests state estimator.<br>2. Server responds to request and receives input file.<br>3. State estimator is triggered and executed.<br>4. Algorithm stores results in an output file. | | |
| **Pass criteria** | State estimator is triggered, receives input data and is executed with no errors. | | |
| **Suspension criteria** | | | |
| **Results** | Test successful | | |

| | | | |
|---|---|---|---|
| **Name** | SE004. Results are delivered to internal ESB. | | |
| **Module under test** | State Estimator | **Resp.** | ICCS |
| **Module requirement** | HL-UC 2_PUC_2_Real-time distribution system awareness | | |
| **Test environment** | Internal ESB up and running<br>Server hosting state estimator up and running | | |
| **Features to be tested** | Server delivers to internal ESB the output file (.xlsx format) with the power flow results. | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | STE003. Calculation request is read from internal ESB. | | |
| **Steps** | 1. Manually inspect the output file created by the state estimator.<br>2. Server responds to initial request and sends the output file. | | |
| **Pass criteria** | Internal ESB receives uncorrupted the output results of the state estimator. | | |
| **Suspension** | | | |

| | |
|---|---|
| **criteria** | |
| **Results** | Test successful |

| | | | |
|---|---|---|---|
| **Name** | SE005. Generation of results as virtual meters | | |
| **Module under test** | State Estimation module | **Resp.** | ETRA |
| **Module requirement** | HL-UC 2_PUC_2_Real-time distribution system awareness | | |
| **Test environment** | CIM Topology model of a grid<br>SPARQL server hosting the model up and running<br>Internal ESB up and running<br>SMX running and sending information to IOP<br>IOP platform up and running<br>3PPF results available | | |
| **Features to be tested** | Results of the state estimation module are published to the IOP in the form of virtual smart meters, becoming therefore available to the any other module of the WiseGRID Cockpit requiring them. | | |
| **Features not to be tested** | | | |
| **Preparation** | SE algorithm implementation generates the results (Excel file) and pushes them to the internal ESB | | |
| **Dependencies** | | | |
| **Steps** | 1. Execute SE orchestrator module | | |
| **Pass criteria** | - Excel file is received from internal ESB<br>- Results from Excel file are extracted<br>- New virtual smart meters (prefix SE_) appear in the system, providing the following information<br>  ○ Per bus: voltage magnitude and angle, current, active and reactive power<br>  ○ Per line: current, active and reactive power, active and reactive power losses | | |
| **Suspension criteria** | | | |
| **Results** | Test successful<br>Upon reception of results, the Excel file is correctly analysed. Results are formatted to DLMS format and published to the IOP. Other modules of WiseGRID Cockpit see these results as if they were delivered by a special set of Smart Meters with SE prefixed ID | | |

## 4.7 FORECAST MODULES

| | | | |
|---|---|---|---|
| **Name** | FOR001. Demand/production forecasting training | | |
| **Module under test** | WG Cockpit forecast module | **Resp.** | ITE |
| **Module requirement** | HL-UC 1_PUC_1_Network monitoring | | |
| **Test environment** | WG Cockpit forecast module up and running<br>Historical data available in long-term DB | | |
| **Features to be tested** | WG Cockpit forecast module is trained | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | RTM002. Store smart meter data to Long-term DB | | |
| **Steps** | Perform WG Cockpit forecast training | | |
| **Pass criteria** | Training MAPE below pre-defined threshold | | |
| **Suspension criteria** | | | |
| **Results** | WG Cockpit forecast model trained | | |

| | | | |
|---|---|---|---|
| **Name** | FOR002. Demand/Production forecasting | | |
| **Module under test** | WG Cockpit forecast module | **Resp.** | ITE |
| **Module requirement** | HL-UC 1_PUC_1_Network monitoring | | |
| **Test environment** | WG Cockpit forecast module up and running<br>Historical data available in long-term DB | | |
| **Features to be tested** | WG Cockpit forecast module performs demand/production forecasting training | | |
| **Features not to be tested** | | | |
| **Preparation** | Train WG Cockpit demand/production forecast module | | |
| **Dependencies** | FOR001. Demand/production forecasting training<br>RTM002. Store smart meter data to Long-term DB | | |
| **Steps** | WG Cockpit forecast module | | |
| **Pass criteria** | Prediction MAPE below pre-defined threshold | | |

| | |
|---|---|
| **Suspension criteria** | |
| **Results** | 24 hours hourly aggregated load/generation power prediction |

| | | | |
|---|---|---|---|
| **Name** | FOR003. Request message parsing test of WG Cockpit forecast module | | |
| **Module under test** | WG Cockpit forecast module | **Resp.** | ITE |
| **Module requirement** | HL-UC 1_PUC_1_Network monitoring | | |
| **Test environment** | Development RabbitMQ environment<br>WG Cockpit forecast module up and running<br>Historical data available in long-term DB | | |
| **Features to be tested** | Performance of WG Cockpit forecast module, at parsing forecast queries. | | |
| **Features not to be tested** | | | |
| **Preparation** | Enable RabbitMQ queues, and run WG Cockpit forecast module | | |
| **Dependencies** | | | |
| **Steps** | 1. Receipt of request<br>2. Request parsing<br>3. DB request according to the requested data<br>4. Treatment of the retrieved data | | |
| **Pass criteria** | - The forecast module is able to decode the queries properly<br>- The forecast module is able to retrieve information from the long-term DB with the parsed information | | |
| **Suspension criteria** | | | |
| **Results** | The module is able to parse the request messages and process it to retrieve information from the long-termDB. | | |

| | | | |
|---|---|---|---|
| **Name** | FOR004. Forecast response message generation test of WG Cockpit forecast module | | |
| **Module under test** | WG Cockpit forecast module | **Resp.** | ITE |
| **Module requirement** | HL-UC 1_PUC_1_Network monitoring | | |
| **Test environment** | RabbitMQ environment<br>WG Cockpit forecast module up and running | | |

| | |
|---|---|
| | Historical data available in long-term DB |
| **Features to be tested** | Performance of WG Cockpit forecast module, at generating and submitting the forecast response. |
| **Features not to be tested** | |
| **Preparation** | Enable RabbitMQ queues<br>Run the demand forecast module |
| **Dependencies** | |
| **Steps** | 1. Parsing of the forecasting algorithm output<br>2. Generating forecast response message |
| **Pass criteria** | - The forecast module is able to analyze properly the output provided by the forecasting algorithm<br>- The forecast module is able to generate properly the forecast response message |
| **Suspension criteria** | |
| **Results** | The module is able to analyze the information provided by the forecast algorithm, and generates the response. |

| | | | |
|---|---|---|---|
| **Name** | FOR005. Forecast is periodically triggered | | |
| **Module under test** | Forecast orchestrator | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_1_Network monitoring | | |
| **Test environment** | Internal ESB up and running<br>Historical data available in long-term DB | | |
| **Features to be tested** | WiseGRID Cockpit periodically posts a demand and a production forecast request per bus to the corresponding queue of the internal ESB | | |
| **Features not to be tested** | | | |
| **Preparation** | Open RabbitMQ monitor, monitor demand and production forecast queues | | |
| **Dependencies** | | | |
| **Steps** | 1. Execute forecast orchestrator module | | |
| **Pass criteria** | - Periodically, every hour, one request per smart meter appears in the demand and production forecast queues<br>- Requests claim next 24 hours hourly prediction | | |
| **Suspension criteria** | | | |
| **Results** | Test successful | | |

| | |
|---|---|
| | The following extract of logs of the docker container wgcockpit_forecastbridge_1 shows that one forecast query for each asset is being posted every hour.<br>`etraid@wisegridpre:~$ docker logs -t --tail 100 wgcockpit_forecastbridge_1 \| grep BBB5976 \| grep querying`<br>`2018-07-24T13:11:06.585842488Z [BBB5976] querying...`<br>`2018-07-24T14:11:06.448221362Z [BBB5976] querying...`<br>`2018-07-24T15:11:06.446890277Z [BBB5976] querying...` |

| | | | |
|---|---|---|---|
| **Name** | FOR006. Forecast results are saved to operational DB | | |
| **Module under test** | Forecast orchestrator | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_1_Network monitoring | | |
| **Test environment** | Internal ESB up and running<br>Historical data available in long-term DB<br>Demand and production forecast modules up and running | | |
| **Features to be tested** | WiseGRID Cockpit receives the results of the forecast module, formats them following the same format used to store real-time data, and stores the in the operational database | | |
| **Features not to be tested** | | | |
| **Preparation** | Open operational database, query next 24 hours of demand/production forecasts | | |
| **Dependencies** | | | |
| **Steps** | 1. Execute forecast orchestrator module | | |
| **Pass criteria** | Periodically, every hour, next 24 hours forecast metrics get updated in the operational database | | |
| **Suspension criteria** | | | |
| **Results** | Test successful<br>The following extract of logs of the docker container wgcockpit_forecastbridge_1 shows that one forecast query for each asset is being posted every hour.<br>`etraid@wisegridpre:~$ docker logs -t --tail 100 wgcockpit_forecastbridge_1 \| grep BBB5976 \| grep forecast`<br>`2018-07-24T13:58:11.305854065Z [BBB5976] forecast received...`<br>`2018-07-24T14:11:07.250519621Z [BBB5976] forecast received...`<br>`2018-07-24T15:11:08.679537937Z [BBB5976]` | | |

| | |
|---|---|
| | `forecast received...` |

## 4.8 CONGESTION FORECAST

| | | | |
|---|---|---|---|
| **Name** | CF001. Congestion forecast is periodically triggered | | |
| **Module under test** | Congestion Forecast orchestrator | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_3_Voltage support and congestion management | | |
| **Test environment** | Internal ESB up and running<br>Historical data available in long-term DB<br>Congestion forecast module up and running<br>3PPF module up and running<br>Demand and production forecast modules up and running | | |
| **Features to be tested** | WiseGRID Cockpit periodically posts the necessary inputs to the corresponding topics of the internal ESB (MQTT, Congestion_Forecast/+ topics) | | |
| **Features not to be tested** | | | |
| **Preparation** | Open MQTTFX to inspect MQTT messages in the internal ESB<br>Subscribe to Congestion_Forecast/+ topic | | |
| **Dependencies** | PFM*<br>FOR* | | |
| **Steps** | 1. Execute congestion forecast orchestrator module | | |
| **Pass criteria** | Periodically, every hour, next 24 hours congestion forecast analysis is triggered by publishing the corresponding messages to the following topics:<br>- Congestion_Forecast/GTP-DCM (topology info.)<br>- Congestion_Forecast/FcP-DCM (forecasted measurements)<br>- Congestion_Forecast/PQG-DCM (real-time measurements) | | |
| **Suspension criteria** | | | |
| **Results** | Test successful<br>The following extract of logs of the docker container wgcockpit_congestionforecastorchestrator_1 shows that the all necessary information to process the congestion forecast gets periodically published to the MQTT.<br>`etraid@wisegridpre:~$ docker logs -t wgcockpit_congestionforecastorchestrator_1`<br>`2018-07-24T07:52:58.835078284Z => Starting meteor app on port:80`<br>`2018-07-24T07:53:08.810599412Z >> [Congestion_Forecast/GTP-DCM] {"Nodes":[{"Node_Number":"650...` | | |

| | |
|---|---|
| | 2018-07-24T07:53:08.846103181Z >> [Congestion_Forecast/FcP-DCM] {"Nodes":[{"Node_Number":"650"...<br>2018-07-24T07:53:08.860761820Z >> [Congestion_Forecast/PQG-DCM]<br>{"Nodes":[{"Node_Number":"650","Name":"650","U1":2401.776832... |

## 4.9 THRESHOLD MONITOR

| | | | |
|---|---|---|---|
| **Name** | TM001. Read smart meter data from internal ESB | | |
| **Module under test** | Threshold Monitor | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_3_Voltage support and congestion management | | |
| **Test environment** | SMX running and sending information to IOP<br>IOP platform up and running | | |
| **Features to be tested** | Data published to the IOP (MQTT protocol) can be collected by the module in real-time | | |
| **Features not to be tested** | | | |
| **Preparation** | Configure one SMX to send data to the lab-testing IOP environment | | |
| **Dependencies** | | | |
| **Steps** | The execution of this test must happen automatically upon publication of data in the IOP | | |
| **Pass criteria** | Data from the SMX is correctly received by the Threshold monitor | | |
| **Suspension criteria** | | | |
| **Results** | Test successful<br>Subscription to IOP is successful, callback is executed upon reception of new data from smart meters | | |

| | | | |
|---|---|---|---|
| **Name** | TM002. Detection of threshold surpass on voltage | | |
| **Module under test** | Threshold Monitor | **Resp.** | ETRA |
| **Module** | HL-UC 1_PUC_3_Voltage support and congestion management | | |

| | |
|---|---|
| **requirement** | |
| **Test environment** | SMX running and sending information to IOP<br>IOP platform up and running |
| **Features to be tested** | Data published to the IOP (MQTT protocol) can be collected by the module in real-time |
| **Features not to be tested** | |
| **Preparation** | Configure one SMX to send mock data, with eventual voltage measurements surpassing ±10% of nominal voltage |
| **Dependencies** | |
| **Steps** | The execution of this test must happen automatically upon publication of data in the IOP |
| **Pass criteria** | The incidents collection contains a registry per voltage deviation detected, detailing timestamp, nominal voltage, detected deviation, bus and phase |
| **Suspension criteria** | |
| **Results** | Test successful<br>Upon (over/under)voltage threshold surpass, an incident object with the details gets written in the incidents collection |

| | | | |
|---|---|---|---|
| **Name** | TM003. Detection of current surpass on line | | |
| **Module under test** | Threshold Monitor | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_3_Voltage support and congestion management | | |
| **Test environment** | SMX running and sending information to IOP<br>IOP platform up and running | | |
| **Features to be tested** | Data published to the IOP (MQTT protocol) can be collected by the module in real-time | | |
| **Features not to be tested** | | | |
| **Preparation** | Configure one Virtual Meter from State Estimation module to send mock data, with eventual current measurements surpassing 90% operational limits of current | | |
| **Dependencies** | SE* | | |
| **Steps** | The execution of this test must happen automatically upon publication of data in the IOP | | |
| **Pass criteria** | The incidents collection contains a registry per current surpass detected, detailing timestamp, operational limit, detected measurement, line and phase | | |

| | |
|---|---|
| **Suspension criteria** | |
| **Results** | Test successful<br>Upon power threshold surpass in a line, an incident object with the details gets written in the incidents collection |

## 4.10 OUTLIER DETECTOR

| | | | |
|---|---|---|---|
| **Name** | OD001. Read smart meter data from internal ESB | | |
| **Module under test** | Outlier detector | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_3_Voltage support and congestion management | | |
| **Test environment** | SMX running and sending information to IOP<br>IOP platform up and running | | |
| **Features to be tested** | Data published to the IOP (MQTT protocol) can be collected by the module in real-time | | |
| **Features not to be tested** | | | |
| **Preparation** | Configure one SMX to send data to the lab-testing IOP environment | | |
| **Dependencies** | | | |
| **Steps** | The execution of this test must happen automatically upon publication of data in the IOP | | |
| **Pass criteria** | Data from the SMX is correctly received by the Outlier Detector | | |
| **Suspension criteria** | | | |
| **Results** | Test successful<br>Threshold monitor successfully subscribes to CIM messages published in the IOP, receiving all updates.<br>The following log extract shows how the outlier detector is triggered whenever new data is sent by the smart meters<br>`etraid@wisegridpre:~ $ docker logs -f wgcockpit_outlierdetector_1`<br>`>> ENER001`<br>`>> ENER003`<br>`>> ENER004`<br>`>> ENER002`<br>`>> BBB5976`<br>`>> BBB5976`<br>`>> ENER001`<br>`>> ENER002` | | |

| | |
|---|---|
| | >> ENER003<br>>> ENER004<br>>> BBB5976 |

| | | | |
|---|---|---|---|
| **Name** | OD002. Detection of outlier on voltage | | |
| **Module under test** | Outlier detector | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_3_Voltage support and congestion management | | |
| **Test environment** | SMX running and sending information to IOP<br>IOP platform up and running | | |
| **Features to be tested** | Module detects outliers on measured voltage | | |
| **Features not to be tested** | | | |
| **Preparation** | Configure one SMX to send mock data, with eventual voltage measurements surpassing detected threshold in voltage | | |
| **Dependencies** | | | |
| **Steps** | The execution of this test must happen automatically upon publication of data in the IOP | | |
| **Pass criteria** | The outlier detector publishes to the internal ESB the detected event | | |
| **Suspension criteria** | | | |
| **Results** | Test successful<br>The following extract of the log shows the message published to the internal ESB whenever an outlier detection in voltage is found<br>>> BBB5976<br>ALERT! {"mRID":"BBB5976", "timeStamp":"2018-07-30T11:06:03.000Z", "readingType":"0.0.0.0.0.1.54.0.0.0.0.0.0.0.128.0.29.0", "value":219.25, "lowerThreshold":219.17000000000002, "upperThreshold":219.2455} | | |

| | | | |
|---|---|---|---|
| **Name** | OD003. Detection of threshold surpass on frequency | | |
| **Module under test** | Outlier detector | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_3_Voltage support and congestion management | | |

| | |
|---|---|
| **Test environment** | SMX running and sending information to IOP<br>IOP platform up and running |
| **Features to be tested** | Module detects outliers on measured frequency |
| **Features not to be tested** | |
| **Preparation** | Configure one SMX to send mock data, with eventual frequency measurements surpassing detected threshold in frequency |
| **Dependencies** | |
| **Steps** | The execution of this test must happen automatically upon publication of data in the IOP |
| **Pass criteria** | The outlier detector publishes to the internal ESB the detected event |
| **Suspension criteria** | |
| **Results** | Test successful<br>The following extract of the log shows the message published to the internal ESB whenever an outlier detection in voltage is found<br>`>> BBB5976 ALERT! {"mRID":"BBB5976", "timeStamp":"2018-07-30T11:05:45.000Z", "readingType":"0.0.0.12.0.1.15.0.0.0.0.0.0.0.224.0.33.0", "value":49.96, "lowerThreshold":49.9641, "upperThreshold":50}` |

## 4.11 POWER QUALITY MODULE

| | | | |
|---|---|---|---|
| **Name** | PQM001. Measurements are read from long-term database. | | |
| **Module under test** | Power Quality Module | **Resp.** | ETRA |
| **Module requirement** | HL-UC 2_SUC_1.3_Monitoring power quality in the grid. | | |
| **Test environment** | Measurement data available in long-term database.<br>Long-term database up and running.<br>Internal ESB up and running. | | |
| **Features to be tested** | Measurement data required for the calculation of the power quality of a node are extracted from the long-term database by the internal ESB. | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | - | | |

| | |
|---|---|
| **Steps** | 1. Internal ESB requests data from long-term database.<br>2. Data is extracted from long-term database.<br>3. Internal ESB requests the calculation of power quality from power quality module. |
| **Pass criteria** | Measurement data are extracted from long term database and internal ESB sends request to module. |
| **Suspension criteria** | |
| **Results** | *Pending* |

| | | | |
|---|---|---|---|
| **Name** | PQM002. Calculation request is read from internal ESB. | | |
| **Module under test** | Power Quality module | **Resp.** | ICCS |
| **Module requirement** | HL-UC 2_SUC_1.3_Monitoring power quality in the grid. | | |
| **Test environment** | Internal ESB up and running<br>Server hosting power quality module up and running | | |
| **Features to be tested** | Power quality module receives input data file and algorithm is executed. | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | PQM001. Measurements are read from long-term database. | | |
| **Steps** | 1. Internal ESB requests power quality module execution.<br>2. Server responds to request and receives input file.<br>3. Power quality module is triggered and executed.<br>4. Algorithm stores results in an output file. | | |
| **Pass criteria** | Power quality module is triggered, receives input data and is executed with no errors. | | |
| **Suspension criteria** | | | |
| **Results** | *Pending* | | |

| | | | |
|---|---|---|---|
| **Name** | PQM003. Results are delivered to internal ESB. | | |
| **Module under test** | Power Quality Module | **Resp.** | ICCS |
| **Module requirement** | HL-UC 2_SUC_1.3_Monitoring power quality in the grid. | | |

| | |
|---|---|
| **Test environment** | Internal ESB up and running<br>Server hosting power quality module up and running |
| **Features to be tested** | Server delivers to internal ESB the output file (.xlsx format) with the power quality results. |
| **Features not to be tested** | |
| **Preparation** | |
| **Dependencies** | PQM002. Calculation request is read from internal ESB. |
| **Steps** | 1. Manually inspect the output file created by the power quality module.<br>2. Server responds to initial request and sends the output file. |
| **Pass criteria** | Internal ESB receives uncorrupted the output results of the power quality module. |
| **Suspension criteria** | |
| **Results** | *Pending* |

## 4.12 EMS FOR ISLANDS

| | | | |
|---|---|---|---|
| **Name** | EMS001. Read and publish data from structural database and forecast module. | | |
| **Module under test** | EMS for islands | **Resp.** | ETRA |
| **Module requirement** | HL-UC 1_PUC_2_Control strategies for reducing RES curtailment. | | |
| **Test environment** | Structural database up and running.<br>Internal ESB up and running.<br>Forecast module up and running. | | |
| **Features to be tested** | Required data from structural database and forecast module are collected and request by ESB is published. | | |
| **Features not to be tested** | | | |
| **Preparation** | It is necessary to have filled the database with all the required data. Forecast module is required to be running properly, meaning that all its dependencies should have been met. | | |
| **Dependencies** | | | |
| **Steps** | 1. Ensure data availability in structural database.<br>2. Ensure data availability from forecast module.<br>3. RabbitMQ collects required data from database and forecast module and fills the input file.<br>4. RabbitMQ publishes a request. | | |
| **Pass criteria** | Structural data of the input file have been correctly gathered and placed in the input | | |

| | | | |
|---|---|---|---|
| | file. Manual cross checking is necessary. Response has been published correctly. | | |
| **Suspension criteria** | | | |
| **Results** | *Pending* | | |

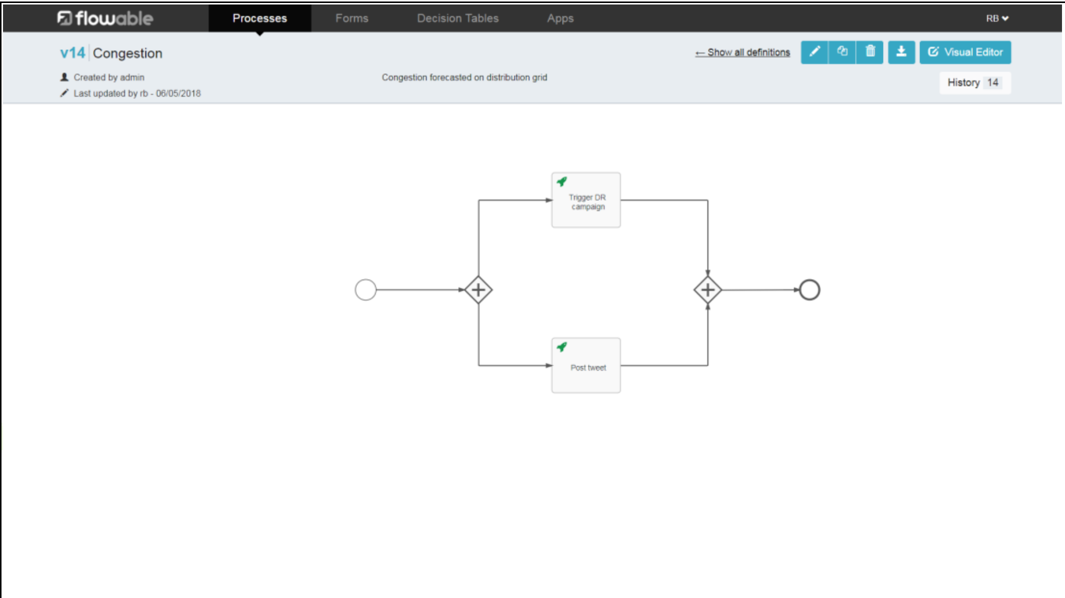
| | | | |
|---|---|---|---|
| **Name** | EMS002. Calculation request is read from internal ESB. | | |
| **Module under test** | EMS for islands | **Resp.** | ICCS |
| **Module requirement** | HL-UC 1_PUC_2_Control strategies for reducing RES curtailment. | | |
| **Test environment** | Internal ESB up and running<br>Server hosting EMS for islands module up and running. | | |
| **Features to be tested** | EMS for islands module receives input data file and algorithm is executed. | | |
| **Features not to be tested** | | | |
| **Preparation** | | | |
| **Dependencies** | EMS001. Read and publish data from structural database and forecast module. | | |
| **Steps** | 1. Internal ESB requests EMS for islands module execution.<br>2. Server responds to request and receives input file.<br>3. EMS for islands is triggered and executed.<br>4. Algorithm stores results in an output file. | | |
| **Pass criteria** | EMS for islands module is triggered, receives input data and is executed with no errors. | | |
| **Suspension criteria** | | | |
| **Results** | *Pending* | | |

| | | | |
|---|---|---|---|
| **Name** | EMS003. Results are delivered to internal ESB. | | |
| **Module under test** | EMS for islands | **Resp.** | ICCS |
| **Module requirement** | HL-UC 1_PUC_2_Control strategies for reducing RES curtailment. | | |
| **Test environment** | Internal ESB up and running<br>Server hosting EMS for islands module up and running | | |
| **Features to be** | Server delivers to internal ESB the output file with the results. | | |

| | |
|---|---|
| **tested** | |
| **Features not to be tested** | |
| **Preparation** | |
| **Dependencies** | EMS002. Calculation request is read from internal ESB. |
| **Steps** | 1. Manually inspect the output file created by the state estimator.<br>2. Server responds to initial request and sends the output file. |
| **Pass criteria** | Internal ESB receives uncorrupted the output results of the power quality module. |
| **Suspension criteria** | |
| **Results** | *Pending* |

## 4.13 GRID FAULT MANAGER

| | | | |
|---|---|---|---|
| **Name** | GFM001. Configuration of custom workflow | | |
| **Module under test** | Grid Fault Manager | **Resp.** | ETRA |
| **Module requirement** | HL-UC 2_PUC_3_Grid control | | |
| **Test environment** | Grid fault manager up and running | | |
| **Features to be tested** | New workflows can be defined from the UI | | |
| **Features not to be tested** | | | |
| **Preparation** | Grid fault manager up and running<br>WG Cockpit UI up and running | | |
| **Dependencies** | | | |
| **Steps** | 1. Access to the incident workflow definition section<br>2. Define a new BPMN workflow | | |
| **Pass criteria** | The new workflow can be defined and gets successfully stored | | |
| **Suspension criteria** | | | |
| **Results** | Test successful<br>New BPMN workflows can be created from Flowable UI. Workflows become automatically available and ready to be triggered from bound events/incidents | | |

| | |
|---|---|
| |  |

| | | | |
|---|---|---|---|
| **Name** | GFM002. Configuration of custom workflow | | |
| **Module under test** | Grid Fault Manager | **Resp.** | ETRA |
| **Module requirement** | HL-UC 2_PUC_3_Grid control | | |
| **Test environment** | Grid fault manager up and running | | |
| **Features to be tested** | Defined incident workflows are executed when the configured incident is detected | | |
| **Features not to be tested** | | | |
| **Preparation** | Setup congestion incidents to trigger Congestion workflow | | |
| **Dependencies** | GFM001 | | |
| **Steps** | 1. Manually simulate detection of a congestion incident | | |
| **Pass criteria** | The steps defined in the Congestion workflow get automatically executed | | |
| **Suspension criteria** | | | |
| **Results** | Test successful<br>The simulation of a congestion event successfully triggers the *Congestion* workflow, which activates a demand response campaign and publishes a tweet on the DSO social media account | | |

| | |
|---|---|
| **Name** | GFM003. Custom workflow triggers action on external module |

| | | | |
|---|---|---|---|
| **Module under test** | Grid Fault Manager | **Resp.** | ETRA |
| **Module requirement** | HL-UC 2_PUC_3_Grid control | | |
| **Test environment** | Grid fault manager up and running<br>Internal ESB up and running<br>Market hub module up and running | | |
| **Features to be tested** | Custom defined workflows have the ability to command the Market Hub module in order to initiate new demand-response campaigns | | |
| **Features not to be tested** | | | |
| **Preparation** | Setup Congestion Forecast workflow to start a demand-response campaign | | |
| **Dependencies** | GFM001 | | |
| **Steps** | 1. Manually simulate forecast of congestion on a line (output of Congestion Forecast module) | | |
| **Pass criteria** | The steps defined in the Congestion Forecast workflow get automatically executed<br>As part of those step, a demand response campaign is initiated<br>WiseGRID IOP market-related queues host the corresponding Flexibility Request triggered by the DSO | | |
| **Suspension criteria** | | | |
| **Results** | The simulation of a congestion event successfully triggers the *Congestion* workflow, which activates a demand response campaign and publishes a tweet on the DSO social media account | | |

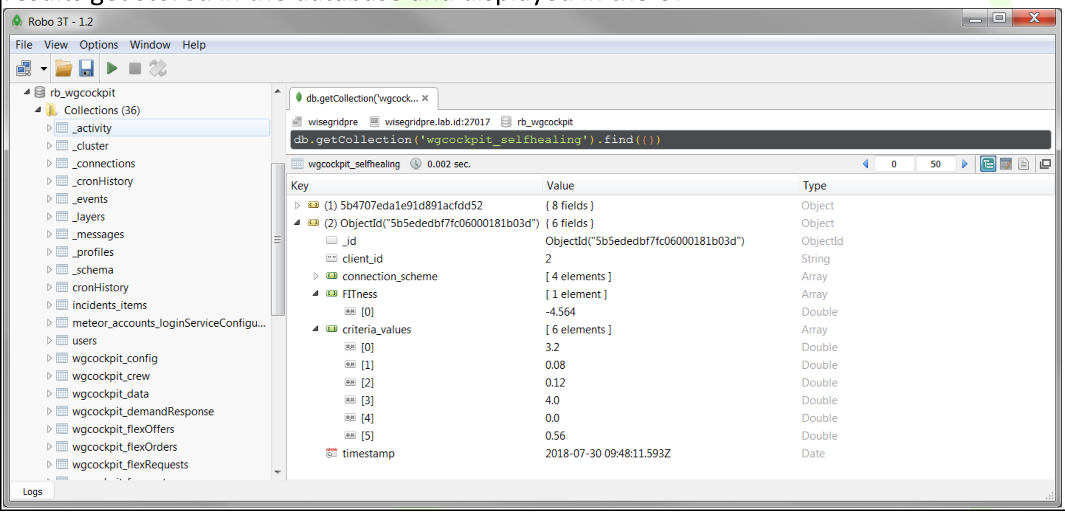
## 4.14 UNPLANNED OUTAGE TREATMENT (FLISR)

| | | | |
|---|---|---|---|
| **Name** | FLI001. Location of fault event on network | | |
| **Module under test** | Fault Location submodule | **Resp.** | ITE |
| **Module requirement** | HL-UC 2_PUC_3_Grid control | | |
| **Test environment** | Fault location submodule up and running | | |
| **Features to be tested** | Ability to identify the location of a fault event in the network lines | | |
| **Features not to be tested** | | | |
| **Preparation** | Import network topology, setup fault passage indicators, setup fault distance relays | | |

| | |
|---|---|
| **Dependencies** | |
| **Steps** | The execution of this test must happen automatically upon event fault detection |
| **Pass criteria** | The fault event is located on expected line segments and sectors expected according to the indications provided by DPF detectors and distance relays. |
| **Suspension criteria** | |
| **Results** | The fault event collection contains a registry per network line fault detected, detailing fault line ID and its sector |

| | | | |
|---|---|---|---|
| **Name** | FLI002. Isolation and Restoration of network after fault event | | |
| **Module under test** | Service Restoration submodule | **Resp.** | ITE |
| **Module requirement** | HL-UC 2_PUC_3_Grid control | | |
| **Test environment** | Service Restoration submodule up and running | | |
| **Features to be tested** | Ability to reconfigure network after a fault event in the network to restore maximum power load | | |
| **Features not to be tested** | | | |
| **Preparation** | Import network topology and parameters, get current network state | | |
| **Dependencies** | FLI001 | | |
| **Steps** | The execution of this test must happen automatically after fault event detection and location | | |
| **Pass criteria** | Non restored power load is smaller than after isolation | | |
| **Suspension criteria** | | | |
| **Results** | Optimal switching connection scheme and switching sequence. The switching sequence contains per registry a switching point ID and its state. Final Fitness and optimization criteria values. | | |

| | | | |
|---|---|---|---|
| **Name** | FLI003. Automatic execution of FLISR module | | |
| **Module under test** | Service Restoration submodule | **Resp.** | ETRA |
| **Module requirement** | HL-UC 2_PUC_3_Grid control | | |
| **Test** | Service Restoration submodule up and running | | |

| | |
|---|---|
| **environment** | |
| **Features to be tested** | Ability to trigger the FLISR module upon detection of a change on a switch status and retrieve the results of the FLISR module |
| **Features not to be tested** | |
| **Preparation** | Enable FLISR module communication with internal ESB |
| **Dependencies** | FLI001,FLI002 |
| **Steps** | The execution of this test must happen automatically after fault event detection and location |
| **Pass criteria** | Results of FLISR execution appear on the operational database |
| **Suspension criteria** | |
| **Results** | Test successful.<br>Integration of the FLISR module within the overall picture of the WGCockpit using the internal ESB has been tested. The FLISR module can be triggered when required, and results get stored in the database and displayed in the UI<br> |

# 5 CONCLUSIONS AND NEXT STEPS

The main conclusion of the work presented in this deliverable is that the methodology followed during the implementation and lab-testing phase was optimal for both tools. The standardization of a process and its explanation to the partners involved in this phase, allowed to avoid any misunderstanding and to follow the same steps so the final result is a coherent and homogeneous work.
All the tests and activities performed within this deliverable have been successful even if there has been necessity or repeating or refining some tests that allowed the involved partners to better understand the singularities of each module.

Although it has not been possible to make all the test that the partners would like to perform for this tool, the main ones and some complementary ones have been done. For Task 14.2 “WiseGRID integrated eco-system Lab-Testing” more tests will be performed (and the ones with pending results will be also documented) in order to prove the integration of the different tools together. During the deployment and demonstration phases, as all the tools will be integrated in real-life conditions and the consortium will have better knowledge of the particularities of each Pilot Site, the partners will be able to collect some feedback and continuing refining the tools and perform some more tests in order to develop the tools and optimally adapt them for the different Pilot Sites.

# 6 REFERENCES AND ACRONYMS

## 6.1 REFERENCES


[1] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Software_test_documentation. [Accessed 14 July 2016].

[2] The International Software Testing Standard, “ISO/IEC/IEEE 29119 Software Testing,” 10 September 2014. [Online]. Available: http://www.softwaretestingstandard.org/. [Accessed 14 July 2016].

[3] G. Arnold, D. Craciun, W. Heckmann and N. Schäfer , “Utility-scale PV installations and their challenges in grid-code compliance testing,” *storage & grids: Technical Briefing,* pp. 74-78, February 2015.

[4] “D13.1 WiseGRID Cockpit Design”.

[5] “akka,” [Online]. Available: https://akka.io/.

[6] [Online]. Available: https://github.com/tdunning/t-digest.

[7] [Online]. Available: https://www.flowable.org/.

[8] “D10.2 WiseGRID Flexibility-based DR Optimization Framework Specifications”.


## 6.2 ACRONYMS

Table 25 – Acronyms list

| **Acronyms List** | |
|---|---|
| AC | Alternative Current |
| AMI | Advanced Metering Infrastructure |
| AMR | Automatic Meter Reading |
| AMQP | Advanced Message Queuing Protocol |
| API | Application Programming Interface |
| CHP | Combined Heat Power |
| CIM | Common Information Model |
| DB | Data Base |
| DC | Direct Current |
| DER | Distributed Energy Resource |
| DLMS | Device Language Message Specification |
| DR | Demand Response |
| DSO | Distribution System Operator |

| | |
|---|---|
| EMS | Energy Management System |
| ESB | Enterprise Service Bus |
| FLISR | Fault Location, Isolation and Supply Restoration |
| GAMS | General Algebraic Modelling System |
| GUI | Graphic User Interface |
| HL-UC | High Level Use Case |
| HTTP | Hypertext Transfer Protocol |
| IOP | InterOperable Platform (another WiseGRID product) |
| KPI | Key Performance Indicator |
| LFC | Load Flow Calculation |
| MQTT | Message Queue Telemetry Transport |
| PV | Photovoltaic |
| RES | Renewable Energy Source |
| RPC | Remote Procedure Call |
| RT | Real-Time |
| SCADA | Supervisory Control And Data Acquisition |
| SMX | Smart Meter eXtension |
| TRP | Thermal and RES Planning |
| TSO | Transmission System Operator |
| UI | User Interface |
| USEF | Universal Smart Energy Framework |

# 7 ANNEX 1: CONGESTION FORECAST EXCHANGE FORMAT

## 7.1 GTP-DCM (GRID TOPOLOGY AND PARAMETERS)

Observation: The JSON examples below use a non-standard comment //; this is for the clarity, and will be removed on de-facto MQTT messages.

Topic: “Congestion_Forecast/GTP-DCM”

Payload example:


```
{
	"Nodes": {
		{
			“Node_Number”: “1”,
			“Name”: “MV_supply_point_01”,
			“Type”: “PQ”,
		},
		{
			“Node_Number”: “2”,
			“Name”: “MV_supply_point_02”,
			“Type”: “PQ”,
		},
		{
			“Node_Number”: “3”,
			“Name”: “DER with U control”,
			“Type”: “PU”,
			“Max_Q”: “130”
		},
		{
			“Node_Number”: “4”,
			“Name”: “MV_supply_point_01”,
// Slack (balancing) node
			“Type”: “Ut”,
			“P”: [“245”,’80’,”83”,”82”],
			“U”: [,”221’,”223”,”224”],
		}
	},
```

"Lines": {
{
“Line_Number”: “1”,
“Node_Connection”: [“1”, “3”],
“Name”: “Line_13”,
“R”: 12.1,
“X”: 5.6,
“d”: 490,
“Imax”: 23
},
{
“Line_Number”: “2”,
“Node_Connection”: [“2”, “3”],
“Name”: “Line_23”,
“R”: 12.1,
“X”: 5.6,
“d”: 490,
“Imax”: 23
}
}
}

## 7.2 FCP-DCM (FORECAST VALUES)

This data model format contains forecasted data generated by FcP (the external forecast program). FcP is connected to WiseGRID cockpit

Observations:

- The JSON examples below use a non-standard comment //; this is for the clarity, and will be removed on de-facto MQTT messages.

- to be noted that the forecast program FcP is agnostic of the fact that the node is PQ, PU or U-teta type; in this respect, FcP is receiving real-time data from the Wisegrid Cockpit and performs forecast algorithm on that data,for P, Q and U in each node

Topic: “Congestion_Forecast/FcP-DCM”

Payload example:

{
"Nodes": {
{
"Node_Number": "1",
"Name": "MV_supply_point_01",
"Date": "2018.05.30 00:00", // date for starting the forecast
"Interval": "60", // interval in minutes
"P1": [31, 27, 12, 14.5 …….. 34, 32.3],
"Q1": [31, 27, 12, 14.5 …….. 34, 32.3],
"U1": [231, 227, 212, 215 …….. 231, 242],
"P2": [31, 27, 12, 14.5 …….. 34, 32.3],
"Q2": [31, 27, 12, 14.5 …….. 34, 32.3],
"U2": [231, 227, 212, 215 …….. 231, 242],
"P3": [31, 27, 12, 14.5 …….. 34, 32.3],
"Q3": [31, 27, 12, 14.5 …….. 34, 32.3],
"U3": [231, 227, 212, 215 …….. 231, 242]
},
{
"Node_Number": "2",
"Name": "MV_supply_point_03",
"Date": "2018.05.30 00:00", // date for starting the forecast
"Interval": "60", // interval in minutes
"P1": [31, 27, 12, 14.5 …….. 34, 32.3]
"Q1": [31, 27, 12, 14.5 …….. 34, 32.3]
"U1": [231, 227, 212, 215 …….. 231, 242]
"P2": [31, 27, 12, 14.5 …….. 34, 32.3],
"Q2": [31, 27, 12, 14.5 …….. 34, 32.3],
"U2": [231, 227, 212, 215 …….. 231, 242],
"P3": [31, 27, 12, 14.5 …….. 34, 32.3],
"Q3": [31, 27, 12, 14.5 …….. 34, 32.3],
"U3": [231, 227, 212, 215 …….. 231, 242]
}
},
"Lines": {
{
"Line_Number": "1",

"Node_Connection": ["1", "3"],
"Name": "Line_13",
"Date": "2018.05.30 00:00", // date for starting the forecast
"Interval": "60", // interval in minutes
"P1": [31, 27, 12, 14.5 ........ 34, 32.3],
"Q1": [31, 27, 12, 14.5 ........ 34, 32.3],
"P2": [31, 27, 12, 14.5 ........ 34, 32.3],
"Q2": [31, 27, 12, 14.5 ........ 34, 32.3],
"P3": [31, 27, 12, 14.5 ........ 34, 32.3],
"Q3": [31, 27, 12, 14.5 ........ 34, 32.3]
},
{
"Line_Number": "2",
"Node_Connection": ["2", "3"],
"Name": "Line_13",
"Date": "2018.05.30 00:00", // date for starting the forecast
"Interval": "60", // interval in minutes
"P1": [31, 27, 12, 14.5 ........ 34, 32.3],
"Q1": [31, 27, 12, 14.5 ........ 34, 32.3],
"P2": [31, 27, 12, 14.5 ........ 34, 32.3],
"Q2": [31, 27, 12, 14.5 ........ 34, 32.3],
"P3": [31, 27, 12, 14.5 ........ 34, 32.3],
"Q3": [31, 27, 12, 14.5 ........ 34, 32.3]
}
},
}

To be observed that the active power series P1,P2, P3 are in fact the forecast of generated power for generation nodes ($P_g$), which is affected by weather (as we are considering mainly RES), or consumption power ($P_c$), for nodes which are consumer type.

FcP need to receive the series of produced and consumed powers, and by using the forecast :

$(P'_g) \rightarrow (P_g)$

$(P'_c) \rightarrow (P_c)$

## 7.3 GOPM-DCM

GOPM (Grid Operations Proposal module) has a list of preconfigured topology changes which are over-lapped over the current topology.

The basic situation is with current topology, meaning a list of lines which connects specific nodes in the grid.

By proposing a new operational mode, there are two possibilities:

- a new line appears in the line list, as a specific breaker is simulated now to be closed and a power flow now should exist through this line

- a line is disconnected, thus its contribution to the general power flow disappears.

GOPM is responsible for giving a list of such new situations, based on pre-programmed grid topologies.

Topic: “Congestion_Forecast/GOPM-DCM”

Payload example:


```
{
        “Date”: “2018.05.30 11:00”,     // date for forecast analysis
        "New_topology": “Sectionalised line 45”,
        "Lines": {
                {
                        “Line_Number”: “1”,
                        “Node_Connection”: [“1”, “3”],
                        “Name”: “Line_13”,
                        “Current_Status”: “OFF”
                        “New_Status”: “ON”
                        “R”: 12.1,
                        “X”: 5.6,
                        “d”: 490
                },
                {
                        “Line_Number”: “6”,
                        “Node_Connection”: [“5”, “12”],
                        “Name”: “Line_5_12”,
                        “Current_Status”: “ON”
                        “New_Status”: “OFF”
                        “R”: 12.1,
                        “X”: 5.6,
                        “d”: 2530
```

}
}

To be noted that the lines status is already known when the power flow is performed by LFC block, as being obtained from LFC-CAM interaction.

CAM (Constraints assessment module) is therefore looking in a configuration file with possible new topologies and generates the JSON listed before.

DCM is triggered to invoke new LFC operations, and before this, DCM is mixing/merging existing topology configuration obtained from the Cockpit (through the GTP-DCM file) with each proposal of new topology. By merging the basic topology with the proposal described in GOPM-DCM, the GOPM-DCM proposal has priority, meaning that a connected line is deleted from the connected lines and a disconnected line si forced to be connected.

The redundancy from the lines:

“Current_Status”: “ON”,

“New_Status”: “OFF”,

Of each line is introduced in order to have a more robust calculation flow, which can make also some checks on the run.

Also, the field below

“Date”: “2018.05.30 11:00”, // date for forecast analysis

Is needed in order to link the request with a specific time period of the forecast.

Finally, a label describing the proposed topology change is also provided:

"New_topology": “Sectionalised line 45”,

such that the new PFC includes also a description of the analysed situation.

## 7.4 DCM-LFC

DCM-LFC (Data collection for computing forecasted constraints Module → Load Flow Calculation) is a file which is covering the input data of the standard input data file 3PPF_input.xlsx.

The following sheets and columns are present in the 3PPF_input.xlsx file, which need a corresponding JSON structure:

**system_data**

Source bus i

Vs (p.u.)

Vnom (kV)

System frequency (Hz)

Earth resistance (Ohm.m)

Convergence limit

**bus_data**

bus i

ground_id

connection_id

Type

P_ph.A  (kW)

P_ph.B (kW)

P_ph.C (kW)

Q_ph.A (kvar)

Q_ph.B (kvar)

Q_ph.C (kvar)

Vnom(kV)

**capacitor_data**

bus i

ground_id

connection_id

Q_ph.A (kvar)

Q_ph.B (kvar)

Q_ph.C (kvar)

Vnom(kV)

**branch_data**

f_bus

t_bus

Status

type_id

Phase Conductor type_id

Neutral Conductor type_id

length (m)

Spacing_id

A

B

C

N

**transformer_data**

f_bus

t_bus

Winding Configuration Primary

Winding Configuration Secondary

Nominal Voltage (kV) Primary

Nominal Voltage (kV) Secondary

R (p.u.)

X (p.u.)

kVA Rating

**VR_data**

from

to

Phase monitoring A

Phase monitoring B

Phase monitoring C

Connection type

Npt

Bandwidth

PRIMARY CT RATING (A)

phase A Rcompensator (V)

phase A Xcompensator (V)

phase B Rcompensator (V)

phase B Xcompensator (V)

phase C Rcompensator (V)

phase C Xcompensator (V)

phase A Voltage level (V)

phase B Voltage level (V)

phase C Voltage level (V)

**switch_data**

f_bus

t_bus

Status

**Conductors**

Conductor type

Description

Diameter (mm)

GMR (mm)

R (Ω/km)

d_od (mm)

d_s (mm)

k

T (mm)

d_sh (mm)

**Spacing ID**

ID-500

ID-505

ID-510

ID-515

ID-520

This structure can be described as follows (comments // are only for clarity, will be not part of the message; double points show that more items can occur, they are also only for clarity, not part of the JSON):

Topic: “Congestion_Forecast/ DCM-LFC”

Payload example:

{
    "Date": "2018.05.30 11:00",
    "system_data": {
        "Source bus I": 650,
        "Vs (p.u.)": 1.00,
        "Vnom (kV)": 4.16,
        "System frequency (Hz)": 50.00,
        "Earth resistance (Ohm.m)": 100,
        "Convergence limit": 0.000001
    },
    "bus_data": [{
            "bus i": 650,
            "ground_id": 1.00,
            "connection_id": 0,
            "Type": 0,

"P_ph.A  (kW)": 0,
"P_ph.B  (kW)": 0,
"P_ph.C  (kW)": 0,
"Q_ph.A  (kW)": 0,
"Q_ph.B  (kW)": 0,
"Q_ph.C  (kW)": 0,
"Vnom(kV)": 4.16
},
{
"bus i": 632,
"ground_id": 1,
"connection_id": 0,
"Type": 0,
"P_ph.A  (kW)": 0,
"P_ph.B  (kW)": 0,
"P_ph.C  (kW)": 0,
"Q_ph.A  (kW)": 0,
"Q_ph.B  (kW)": 0,
"Q_ph.C  (kW)": 0,
"Vnom(kV)": 4.16
}
],
"capacitor_data": [],
"branch_data": [],
"transformer_data": [],
"VR_data": [],
"switch_data": [],
"Conductors": [],
"Spacing ID": []
}

If necessary, a translation program from JSON to XLS may be needed, to be agreed with ICCS.

## 7.5 LFC-CAM

LFC-CAM (Load Flow Calculation→Constraints Assessment module) is a file which is covering the output data of the standard output data file 3PPF_output.xlsx.

The following sheets and columns are present in the 3PPF_output.xlsx file, which need a corresponding JSON structure:

**bus voltages p.u.**

Bus

Phase A Magnitude (p.u.)

Phase A Angle (Deg.)

Phase B Magnitude (p.u.)

Phase B Angle (Deg.)

Phase C Magnitude (p.u.)

Phase C Angle (Deg.)

**bus voltages**

Bus

Phase A Magnitude (p.u.)

Phase A Angle (Deg.)

Phase B Magnitude (p.u.)

Phase B Angle (Deg.)

Phase C Magnitude (p.u.)

Phase C Angle (Deg.)

**Incoming Currents**

Bus

Phase A Magnitude (p.u.)

Phase A Angle (Deg.)

Phase B Magnitude (p.u.)

Phase B Angle (Deg.)

Phase C Magnitude (p.u.)

Phase C Angle (Deg.)

**Outgoing Currents**

f_bus

t_bus

Phase A Magnitude (p.u.)

Phase A Angle (Deg.)

Phase B Magnitude (p.u.)

Phase B Angle (Deg.)

Phase C Magnitude (p.u.)

Phase C Angle (Deg.)

**Incoming Power**

Bus

Phase A Active Power (kW)

Phase A Reactive Power (kvar)

Phase B Active Power (kW)

Phase B Reactive Power (kvar)

Phase C Active Power (kW)

Phase C Reactive Power (kvar)

**Outgoing Power**

f_bus

t_bus

Phase A Active Power (kW)

Phase A Reactive Power (kvar)

Phase B Active Power (kW)

Phase B Reactive Power (kvar)

Phase C Active Power (kW)

Phase C Reactive Power (kvar)

**Computational time**

Time Counters

Input data reading and process computational time

Output data process and writing computational time

Power flow computational time

Total computational time

This structure can be described as follows (comments // are only for clarity, will be not part of the message; double points show that more items can occur, they are also only for clarity, not part of the JSON):

Topic: “Congestion_Forecast/ LFC-CAM”

Payload example:

{
    “Date”: “2018.05.30 11:00”,     // date for forecast analysis
    " bus voltages p.u.": {
            “Bus”: 650,
            “Phase A Magnitude (p.u.)”:  1.00,
            “Phase A Angle (Deg.)”: 4.16,
            “Phase B Magnitude (p.u.)”:  1.00,
            “Phase B Angle (Deg.)”: 4.16,

“Phase C Magnitude (p.u.)”:  1.00,
“Phase C Angle (Deg.)”: 4.16
},
" bus voltages": {
“Bus”: 650,
“Phase A Magnitude (p.u.)”:  1.00,
“Phase A Angle (Deg.)”: 4.16,
“Phase B Magnitude (p.u.)”:  1.00,
“Phase B Angle (Deg.)”: 4.16,
“Phase C Magnitude (p.u.)”:  1.00,
“Phase C Angle (Deg.)”: 4.16
},
"Incoming Currents ": {
“Bus”: 650,
“Phase A Magnitude (p.u.)”:  1.00,
“Phase A Angle (Deg.)”: 4.16,
“Phase B Magnitude (p.u.)”:  1.00,
“Phase B Angle (Deg.)”: 4.16,
“Phase C Magnitude (p.u.)”:  1.00,
“Phase C Angle (Deg.)”: 4.16
},
"Outgoing Currents ": {
“f_bus”: 650,
“t_bus”: 650,
“Phase A Magnitude (p.u.)”:  1.00,
“Phase A Angle (Deg.)”: 4.16,
“Phase B Magnitude (p.u.)”:  1.00,
“Phase B Angle (Deg.)”: 4.16,
“Phase C Magnitude (p.u.)”:  1.00,
“Phase C Angle (Deg.)”: 4.16
},
"Incoming Power ": {
“Bus”: 650,
“Phase A Active Power (kW)”:  1.00,
“Phase A Reactive Power (kvar)”: 4.16,
“Phase B Active Power (kW)”:  1.00,

"Phase B Reactive Power (kvar)": 4.16,
"Phase C Active Power (kW)": 1.00,
"Phase C Reactive Power (kvar)": 4.16,
},
"Outgoing Power ": {
"f_bus": 650,
"t_bus": 650,
"Phase A Active Power (kW)": 1.00,
"Phase A Reactive Power (kvar)": 4.16,
"Phase B Active Power (kW)": 1.00,
"Phase B Reactive Power (kvar)": 4.16,
"Phase C Active Power (kW)": 1.00,
"Phase C Reactive Power (kvar)": 4.16,
},
" Computational time": {
"Time Counters": 650,
"Input data reading and process computational time": 650,
"Output data process and writing computational time": 1.00,
"Power flow computational time": 4.16,
"Total computational time": 1.00,
}
}

## 7.6 PQG-DCM

PQG-DCM (P and Q generated→ Data collection for computing forecasted constraints Module) is a file which is covering real-time output data of the WiseGRID cockpit which is sent to DCM.

Topic: "Congestion_Forecast/PQG-DCM"

Payload example:

{
"Date": "2018.05.27 11:43:35", // date stamp for the real-time values
"Nodes": {
{
"Node_Number": "1",
"Name": "MV_supply_point_01",
"P1": 31.2,

“Q1”: 56.4,
“U1”: 231.8,
“P2”: 31.2,
“Q2”: 56.4,
“U2”: 231.8,
“P3”: 31.2,
“Q3”: 56.4,
“U3”: 231.8,
},
{
“Node_Number”: “24”,
“Name”: “MV_supply_point_17”,
}
},
"Lines": {
{
“Line_Number”: “1”,
“Node_Connection”: [“1”, “3”],
“Name”: “Line_13”,
“P1”: 121.3,
“Q1”: -17.2,
“P2”: 121.3,
“Q2”: -17.2,
“P3”: 121.3,
“Q3”: -17.2,
},
{
“Line_Number”: “2”,
“Node_Connection”: [“2”, “3”],
“Name”: “Line_17”,
“P1”: -54.7,
“Q1”: 21.8,
“P2”: -37.9,
“Q2”: 17.3,
“P3”: -38.5,
“Q3”: 12.4,

}
}
}

## 7.7 DCM-CKP

DCM-CkP (Data collection for computing forecasted constraints Module → WiseGRID cockpit) is a file which sends to the WiseGRID cockpit Congestion forecast results.

Topic: “Congestion_Forecast/DCM-CkP”

Payload example:


```
{
"Congestions": {
    “Date”: “2018.05.30 12:45”,    // date stamp for the congestion values
    "Nodes": {
        {
            “Node_Number”: “1”,
            “Name”: “MV_supply_point_01”,
            // Limits for U in the node: [LIM_LOW_short_time, LIM_LOW_long_time,
            // U_nominal, LIM_HIGH_long_time, LIM_HIGH_short_time]
            “U_Limits”:  [195.0, 207.0, 230.0, 253.0, 255.0]
            “U1_Assessment”: “HIGH”,
            “U1”: 254.4,
            “U2_ Assessment”: “OK”,
            “U2”: 245.3,
            “U3_ Assessment”: “HIGH_HIGH”,
            “U3”: 255.9,
        },
        {
            “Node_Number”: “14”,
            “Name”: “MV_supply_point_01”,
            // Limits for U in the node: [LIM_LOW_short_time, LIM_LOW_long_time,
            // U_nominal, LIM_HIGH_long_time, LIM_HIGH_short_time]
            “U_Limits”:  [195.0, 207.0, 230.0, 253.0, 253.0]
            “U1_Assessment”: “LOW”,
            “U1”: 203.6,
            “U2_ Assessment”: “OK”,
```

“U2”: 207.3,
“U3_ Assessment”: “LOW_LOW”,
“U3”: 194.2,
},
},
"Lines": {
{
“Line_Number”: “1”,
“Node_Connection”: [“1”, “3”],
“Name”: “Line_13”,
// Limits for I on the line: [I_nominal, LIM_HIGH_long_time, LIM_HIGH_short_time]
“I1_ Assessment”: “HIGH”,
“I1”: 25.4,
“P1”: 121.3,
“Q1”: -17.2,
“I2_ Assessment”: “HIGH”,
“I2”: 25.4,
“P2”: 121.3,
“Q2”: -17.2,
“I2_ Assessment”: “HIGH”,
“I3”: 25.4,
“P3”: 121.3,
“Q3”: -17.2,
},
{
“Line_Number”: “7”,
“Node_Connection”: [“12”, “21”],
“Name”: “Line_42”,
// Limits for I on the line: [I_nominal, LIM_HIGH_long_time, LIM_HIGH_short_time]
“I1_ Assessment”: “HIGH”,
“I1”: 25.4,
“P1”: 121.3,
“Q1”: -17.2,
“I2_ Assessment”: “HIGH”,
“I2”: 25.4,
“P2”: 121.3,

"Q2": -17.2,
"I2_ Assessment": "HIGH",
"I3": 25.4,
"P3": 121.3,
"Q3": -17.2,
}
},
{
"Date": "2018.05.30 13:00", // date stamp for the congestion values
"Nodes": {
……
}
"Lines": {
……
},
{
"Date": "2018.05.30 13:15", // date stamp for the congestion values
"Nodes": {
……
}
"Lines": {
……
}
}

## 7.8 CAM-GOPM

CAM-GOPM (Constraints assessment Module → Grid Operations Proposal module) is a file which sends to GOPM the constraints results.

Topic: "Congestion_Forecast/CAM-GOPM"

Payload example:

{
"Congestions": {
"Date": "2018.05.30 12:45", // date stamp for the congestion values
"Nodes": {
{

"Node_Number": "1",
"Name": "MV_supply_point_01",
"U1_Assessment": "HIGH",
"U2_ Assessment": "OK",
"U3_ Assessment": "HIGH_HIGH",
},
{
"Node_Number": "14",
"Name": "MV_supply_point_01",
"U1_Assessment": "LOW",
"U2_ Assessment": "OK",
"U3_ Assessment": "LOW_LOW",
},
},
"Lines": {
{
"Line_Number": "1",
"Node_Connection": ["1", "3"],
"Name": "Line_13",
"I1_ Assessment": "HIGH",
"I2_ Assessment": "HIGH",
"I2_ Assessment": "HIGH",
},
{
"Line_Number": "7",
"Node_Connection": ["12", "21"],
"Name": "Line_42",
"I1_ Assessment": "HIGH",
"I2_ Assessment": "HIGH",
"I2_ Assessment": "HIGH"
}
},
{
"Date": "2018.05.30 13:00", // date stamp for the congestion values
"Nodes": {
……

}
"Lines": {
……
},
{
“Date”: “2018.05.30 13:15”,       // date stamp for the congestion values
"Nodes": {
……
}
"Lines": {
……
}
}