



e-balance

Deliverable D5.5

Integration of the energy management platform

Editor:	Marijn Jongerden and Marco Gerards (University of Twente)
Dissemination level: (Confidentiality)	PU
Suggested readers:	Consortium/Experts
Version:	1.0
Total number of pages:	54
Keywords:	smart grids, energy management platform, integration

Abstract

This document provides an overview of how the different modules of the e-balance energy management platform, which have been developed within Work Package 5, are integrated.

The integration of each software module has been evaluated through a set of functional tests. The tests are described in the form of test cards. The results of the tests are given in the corresponding result cards. The results show that the different modules perform as expected after integration.

Disclaimer

This document contains material, which is the copyright of certain e-balance consortium parties, and may not be reproduced or copied without permission.

All e-balance consortium parties have agreed to full publication of this document.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the e-balance consortium as a whole, nor a certain party of the e-balance consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

The information, documentation and figures available in this deliverable are written by the e-balance partners under EC co-financing (project number: 609132) and does not necessarily reflect the view of the European Commission.

Impressum

[Full project title] Balancing energy production and consumption in energy efficient smart neighbourhoods

[Short project title] e-balance

[Number and title of work-package] WP5 Energy Management Platform

[Document title] Integration of the energy management platform

[Editor: Name, company] Marijn Jongerden and Marco Gerards, University of Twente

[Work-package leader: Name, company] Marco Gerards, University of Twente

Copyright notice

© 2013-2017 Participants in project e-balance

Executive Summary

This deliverable gives an overview of how the different parts that have been developed within the various Work Package 5 tasks are integrated in the e-balance energy management platform.

In Work Package 5 we have developed the different software modules for the energy balancing and energy resilience features, as well as the security and privacy mechanisms, which ensure the proper data access for these modules. All these modules are integrated in the energy management platforms in order to obtain an operational system that can be deployed in the demonstrators.

This document provides the description of the interfaces between the different modules. Additionally, the tests that are performed to check the operation of the modules in the integrated system are listed in the form of test cards. The results of the tests are listed as well, in the form of corresponding result cards.

In Section 2 we discuss how the energy management platform is built upon the middleware that was developed in Work package 4. This section also describes how the mechanisms for secure data access of the energy balancing modules are integrated. In Section 3 the integration of the energy balancing modules is described. Section 4 describes the integration of the energy resilience modules. In all of these sections the corresponding tests and test results are listed. The document is concluded in Section 5.

List of authors

Company	Author
University of Twente	Marijn Jongerden Marco Gerards James Piggott
University of Malaga	Daniel Garrido Jaime Chen Eduardo Cañete
IHP	Krzysztof Piotrowski Ievgen Kabin Peter Langendörfer
EFACEC	Filipe Campos Francisco Basadre Eduardo Rodrigues Alberto Rodrigues Alberto Bernardo
INOV	Mário Nunes

Table of Contents

Executive Summary.....	3
List of authors.....	4
Table of Contents	5
List of Figures.....	6
Abbreviations	7
1 Introduction	9
2 Integration of energy management platform with communication platform middleware.....	10
2.1 Integration of the Java Wrapper and energy management services	10
2.2 Integration of the security and privacy module.....	11
2.2.1 Interface description	11
2.2.2 Tests and results.....	12
2.3 Integration of the balancing module with the communication platform middleware	14
2.3.1 Create and write to variables	14
2.3.2 Polling variables	14
2.3.3 Subscribe to variable event handlers	15
3 Integration of the energy balancing modules	17
3.1 Integration of balancing logic with device controller	17
3.1.1 Interface description	17
3.1.2 Tests and results.....	18
3.2 Integration of balancing logic with the prediction module	23
3.2.1 Interface description	23
3.2.2 Tests and results.....	24
4 Integration of the grid resilience modules.....	27
4.1 Integration of LV NH Power Flow, DER Power Flows and other components	27
4.1.1 LV NH Power flow – Interface.....	27
4.1.2 LV NH Power flow.....	32
4.1.3 LV NH Power flow – Tests and results.....	34
4.2 Integration of self-healing FDIR and communication platform.....	37
4.2.1 Interface description	37
4.2.2 Electrical network topology.....	38
4.2.3 Tests and results.....	38
4.3 Integration of VOS, OPF and communication platform	41
4.3.1 Interface description	41
4.3.2 Underlying Communication	42
4.3.3 Tests and results.....	43
4.4 Integration of LV Grid Resilience Modules.....	47
4.4.1 LV Fault Management - LV Fault Prevention - Dynamic Voltage Control	47
4.4.2 Integration of LV Fraud, LV Quality and LV Fault Management modules	48
5 Conclusions	53
References	54

List of Figures

Figure 1: Position of deliverable D5.5 within the e-balance project.	9
Figure 2: Sketch of the Java Wrapper and its placement in the software architecture	10
Figure 3: Energy management platform and communication middleware integration.	11
Figure 4: The security and privacy module within the Java Wrapper instance	11
Figure 5: UML diagram of the integration of the balancing logic with FPAI.....	18
Figure 6: Sequence diagram of prediction module interaction.....	24
Figure 7: NH Power Flow components architecture	27
Figure 8: LV Network Data – snapshot workflow	29
Figure 9: LV Network Data – snapshot data	29
Figure 10: Power flow execution using snapshots	30
Figure 11: LV Power flow results image.....	31
Figure 12: LVGMU application blocks.....	32
Figure 13: Power Flow C/Java Integration.....	33
Figure 14: Relevant electrical network for FDIR tests.....	38
Figure 15: Component architecture	43
Figure 16: Efacec’s “G Smart” extended with Dynamic Voltage Control software on a PC.....	47
Figure 17: Sequence diagram of the communication between the Voltage Control PC and the G Smart	48
Figure 18: Integration of grid resilience modules in the G Smart	49

Abbreviations

AMI	Advanced Metering Infrastructure
API	Application Programming Interface
CMU	Customer Management Unit
COM	Common Connector
DA	Distributed Automation
DBMS	Data Base Management System
DER	Distributed Energy Resource
DER-MU	Distributed Energy Resource Management Unit
DMU	Device Management Unit
DOPF	Distributed Optimal Power Flow
DoW	Description of work
DSM	Demand Side Management
DSO	Distribution System Operator
EAN	European Article Number
EC	European Commission
ES	Energy Supplier
ESCO	Energy Service Company
FDIR	Fault Detection, Isolation and Restoration
FTP	File Transfer Protocol
GMU	Grid Management Unit
GSS	Grid Support Service
GUI	Graphical User Interface
GW	Gateway
HV	High Voltage
ICT	Information and Communication Technologies
IEC	International Electrotechnical Commission
IED	Intelligent Energy Device
IP	Internet Protocol
KPI	Key Performance Indicator
LAN	Local Area Network
LV	Low Voltage
LV-FAN	Low Voltage Field Area Network
LVGMU	Low Voltage Grid Management Unit
MAIFI	Momentary Average Interruption Frequency Index
MGCC	Microgrid Central Controller
MV	Medium Voltage
MV-FAN	Medium Voltage Field Area Network
MVFDL	Medium Voltage Fault Detection and Location
MVGMU	Medium Voltage Grid Management Unit
NC	Normally Closed
NO	Normally Open
NPF	Neighbourhood Power Flow
OPF	Optimised Power Flow
PC	Personal Computer
PL	Power line
PLC	Power Line Communications
PS	Primary Substation
PV	Photovoltaic panel
QoE	Quality-of-Experience
QoS	Quality of Supply
RES	Renewable Energy Sources
RFMesh	Radiofrequency Mesh
RTU	Remote Terminal Unit
SAIDI	System Average Interruption Duration Index

SM	Smart Meter
SPDT	Single-pole Double Throw
SS	Secondary Substation
TL-GMU	Top Level Grid Management Unit
TSDB	Time Series Data Base
TSO	Transmission System Operator
UC	Use Case
VOS	Validation of Optimised Solutions
VPN	Virtual Private Network
WLAN	Wireless Local Area Network

1 Introduction

In order to get a step closer to the operating demonstrators, the different features that have been developed within Work package 5 need to be integrated. This document gives an overview of how the different software modules are integrated, and lists the tests that were performed to check that the modules work as expected.

Figure 1 shows the position of this deliverable within the e-balance project. This deliverable is part of work package WP 5 – Energy Management Platform.

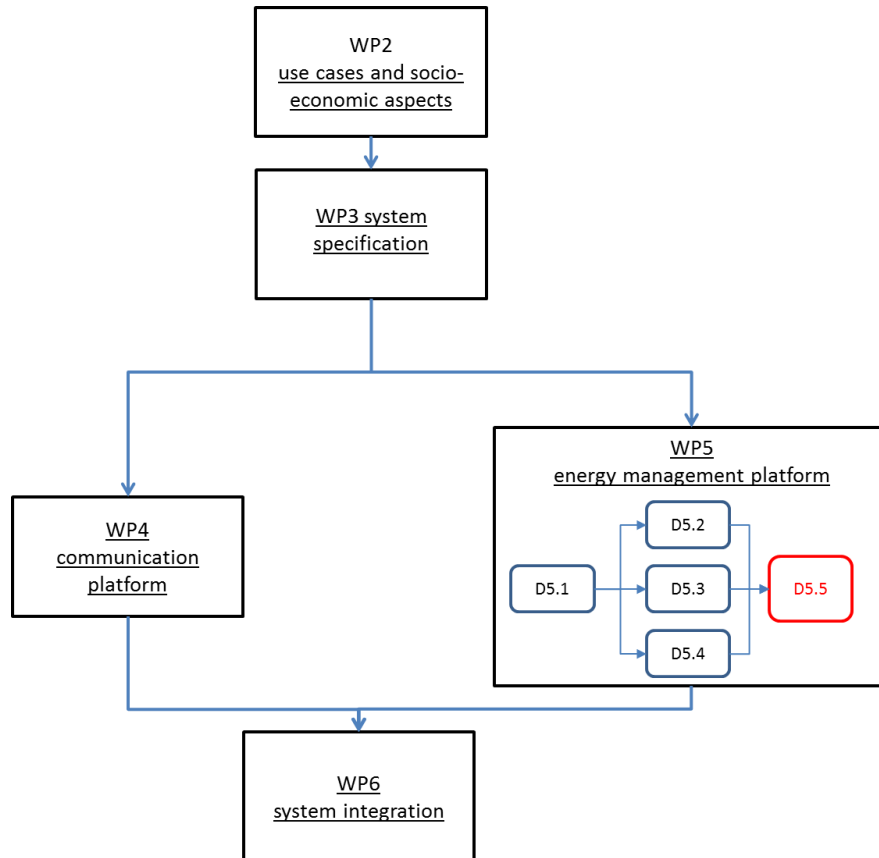


Figure 1: Position of deliverable D5.5 within the e-balance project.

Deliverable 5.2 [1] provides the algorithms for energy balancing and the algorithms for predicting the electricity production and consumption. Deliverable D5.3 [2] provides the algorithms for the various resilience and self-healing mechanisms. Deliverable D5.4 [3] provides the needed security and privacy mechanisms. In this deliverable (D5.5) all this work comes together and is integrated in the energy management platform. As this document describes integration work and results, it does not extend the state-of-the-art and due to this the beyond-state-of-the-art subsection is not included in his deliverable.

Moreover, as various algorithms make use of the middleware developed in WP4 [4], we decided not only to integrate modules developed in WP5 but to do first integration steps und tests also with the communication platform middleware. This goes beyond the originally planned scope of the integration work in WP5 but reduces effort of the integration of the demonstrators. In more clear words we anticipated part of tasks to be done in WP6.

The document is structured as follows. In Section 2 the interface to the middleware is described, as well as the security and privacy mechanism for accessing the data stored in the middleware. Sections 3 and 4, give an overview of how the different modules are integrated, for the Energy Balancing modules and Grid Resilience modules, respectively. Sections 2, 3 and 4, all list the various tests that have been performed to check the integration of the different modules, and show the results of the tests. Finally, the document is concluded in Section 5 with an outlook towards the integration of the different management units for the demonstrators.

2 Integration of energy management platform with communication platform middleware

This section describes the integration of the modules responsible for interactions between the communication platform and the energy management platform. This is an important part of the work done in Task T5.5, since the energy management platform actually relies on the communication platform that provides the glue for all the distributed energy management logic located on the different management units. The direct connection between the management logic and the communication platform middleware is realised using the Java Wrapper (see Figure 2).

Additionally, as part of the interaction between these two parts of the e-balance system, the security and privacy module is located within the Java Wrapper. Thus, this section also describes the integration of this module within the energy management platform.

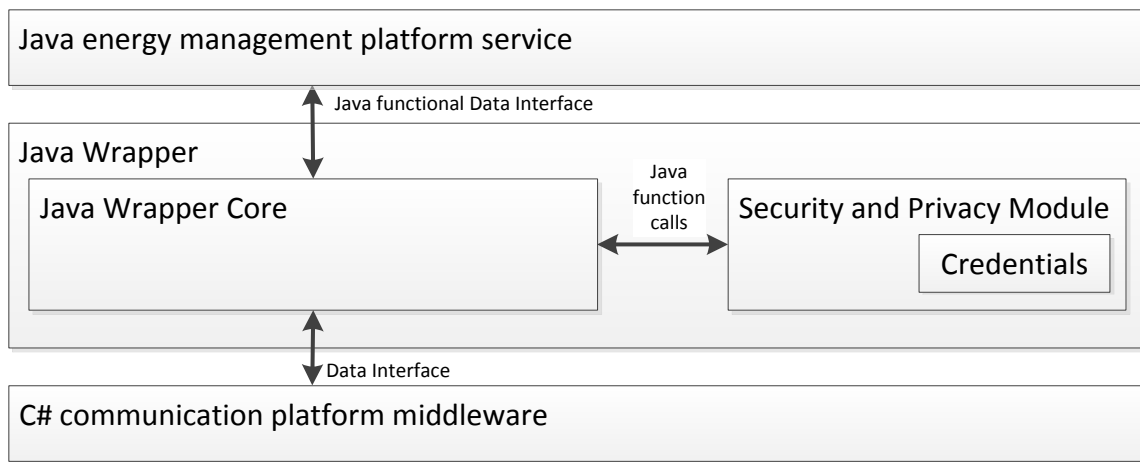


Figure 2: Sketch of the Java Wrapper and its placement in the software architecture

The Java Wrapper is instantiated for each energy management service running in the particular management unit. Thus, for each service there is also a new instance of the security and privacy module with the credentials corresponding to the service provider (stakeholder) and the service itself.

2.1 Integration of the Java Wrapper and energy management services

As described in deliverable D4.3, smart grids systems are complex systems with a heterogeneous set of devices communicating with each other. Furthermore, all these devices have to communicate within a hierarchical structure in a secure way and under different roles.

In order to facilitate the development of applications on the top of e-balance system a middleware has been designed and implemented to provide an abstraction layer that hides the communication related details. This middleware provides developers a simple way of programming applications for the e-balance system and at the same time it manages the complex underlying communication network.

Applications run on top of the middleware using the API provided to exchange information. Figure 3 describes the communication architecture that each management unit running the energy management platform application uses.

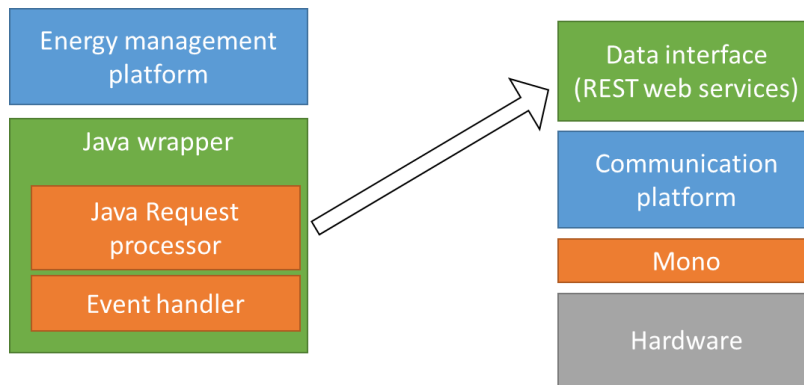


Figure 3: Energy management platform and communication middleware integration.

The communication platform middleware offers a simple API based on REST web services. As mentioned previously, the energy management platform has been implemented in Java. In order to simplify the communication between instances of the energy management platform running in different management units, a Java wrapper has been developed. The Java wrapper allows Java applications to use the communication middleware with functional operations rather than having to use web services. More details about the Java wrapper and its communication with the communication platform can be found in deliverable D4.4.

The data in the energy management platform is represented as variables. All the energy management platform instances manipulate variables either locally or remotely. The following sections summarise the different variables that the energy management platform instances exchange.

2.2 Integration of the security and privacy module

2.2.1 Interface description

The security and privacy module is implemented in Java and is realised as a Class with a given interface, defining the functions (methods) provided by the Class. These functions allow adding the security and privacy related aspects to the data access requests issued by the service (signing the request – for authentication and encrypting the values for confidentiality). This influences the internal structure of the request and is de facto transparent for the service itself.

The detailed description of the functions provided by the security and privacy module class are defined in deliverable D5.4 [3]. These functions are used by the Java Wrapper Core from the Java Wrapper instance created for the specific service (see Figure 4). Each energy management service has its own instance of the Java Wrapper and, as a result, also its individual instance of the security and privacy module. Thus, independent from the number of different services running on a management unit, each service has its own security and privacy module and its own security credentials and context.

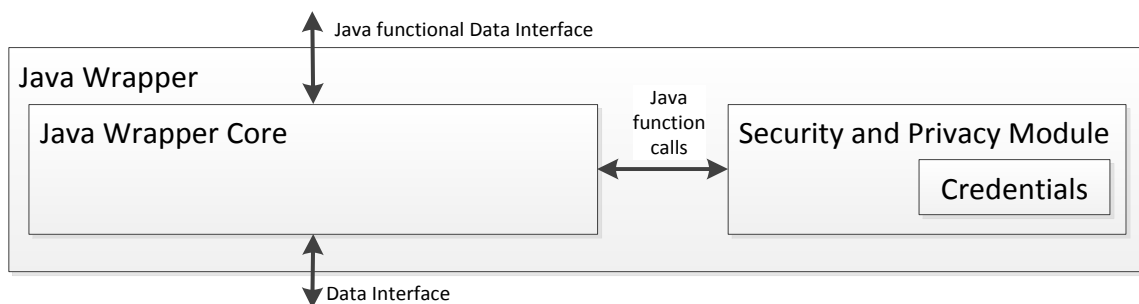


Figure 4: The security and privacy module within the Java Wrapper instance

2.2.2 Tests and results

The following tests were performed using test applications emulating the behaviour of a service – issuing a data access request (reading and writing a defined variable), as well as using a test application testing the right content of the request structure to be sent further to the communication platform middleware, according to the known context (security credentials).

2.2.2.1 Testing the signature generation for a data access request issued by a service

Test card	
Name/code/test case identifier	TestSignature
Objectives	Test the process of generating the signature for a data access request
Devices involved	Management Unit (CMU or LVGMU or MVGMU or TLGMU)
Pre-requirements	The keystore file with credentials used for the test service is created prior the test. Simple test application that instantiates the Java Wrapper and generates a test request using the Java Data Interface is available.
Steps	<ol style="list-style-type: none"> 1. Create the service (test application) with its instance of Java Wrapper. 2. The service issues the test read requests. 3. The request structures that are generated by the Java Wrapper are verified according to the known context (credentials).
Expected result	The request structure is correct and the signature can be positively verified for the provided certificate by the test application analysing the request structure.
Additional comments	

Result card	
Name/code/test case identifier	TestSignature
Results	Test positive for a series of 1000 requests for given credentials (keystores). In total 5 different credentials were used (200 requests per each).

2.2.2.2 Testing the data encryption

Test card	
Name/code/test case identifier	TestEncryption
Objectives	Test the data encryption (individual values of the variable) realised using the Java Wrappers with the security and privacy modules with individual credentials
Devices involved	Management Unit (CMU or LVGMU or MVGMU or TLGMU)
Pre-requirements	The keystore with credentials used for the test service is created prior the test. Simple test application that instantiates the Java Wrapper, initialises the data encryption block in the security and privacy module and generates a

	test write request using the Java Data Interface is available.
Steps	<ol style="list-style-type: none"> 1. Create the test service application with its instance of Java Wrappers. 2. The service initialises the data encryption block in the security and privacy module with known initialization data. 3. The services issue the test write requests. 4. The request structures that are generated by the Java Wrapper are verified according to the known context (credentials).
Expected result	The value written to the variable is correct according to the context (used initialization data) and can be decrypted by the test application analysing the request structure.

Result card	
Name/code/test case identifier	TestEncryption
Results	Test positive for a series of 1000 requests for given credentials (initialisation data). In total 5 different credentials were used (200 requests per each).

2.2.2.3 Testing the instantiation of the integrated secure Java Wrapper

Test card	
Name/code/test case identifier	TestJavaWrapperSecurity
Objectives	Test the data instantiation of the individual Java Wrappers with the security and privacy modules with individual credentials
Devices involved	Management Unit (LVGMU or MVGMU or TLGMU)
Pre-requirements	<p>The keystores with credentials used for the individual instantiations (services) are created prior the test and the keystore files are available.</p> <p>Simple test applications that instantiate the Java Wrapper pointing at the respective keystores that generate a test request using the Java Data Interface available.</p>
Steps	<ol style="list-style-type: none"> 1. Create two services (test applications) with their instances of Java Wrappers. 2. The services issue the test requests. 3. The request structures, which are generated by the Java Wrapper, are verified according to the known context (credentials).
Expected result	Two services are instantiated and the issued requests are correctly signed and the signatures can be verified by the test application analysing the request structure.

Result card	
Name/code/test case identifier	TestJavaWrapperSecurity
Results	Test positive for a series of 10 requests for the given service credentials (keystore).

2.3 Integration of the balancing module with the communication platform middleware

The communication platform middleware is required to support a number of complex interactions in order for the balancing logic to be able to carry out a balancing cycle. Three general tests were devised to check the validity of the Java wrapper (API) and its ability to allow the balancing cycle to communicate with the middleware.

2.3.1 Create and write to variables

Test card	
Name/code/test case identifier	testVariableCreationWrite
Objectives	Test if the balancing logic can create and write values to a variable in the middleware using the asynchronous API
Devices involved	CMU
Pre-requirements	An instance of the middleware and the CMU-Server is running
Steps	Start CMU-Server and middleware Start balancing module Balancing module creates new variable in the middleware through the java wrapper. If callback successful Write a series of values to the variable in the middleware through the java wrapper.
Expected result	If callback from both types of operations returns then test is successful

Result card	
Name/code/test case identifier	testVariableCreationWrite
Results	Test successful.

2.3.2 Polling variables

Test card	
Name/code/test case identifier	testRetrieveValues
Objectives	Test if the balancing module can retrieve values stored in the middleware. Test includes retrieving latest values, values corresponding to time interval or value condition

Devices involved	CMU
Pre-requirements	An instance of the middleware and the CMU-Server are running
Steps	Start CMU-Server and middleware Start balancing module Retrieve last value entered into the variable Retrieve all value between two timestamp dates Retrieve all values permitted by a condition that evaluates to true
Expected result	For each of the three tests the correct values are returned
Additional comments	This test makes use the synchronous API as only the local middleware instance is tested. As such there is no callback from asynchronous operation

Result card	
Name/code/test case identifier	testRetrieveValues
Results	Test successful.

2.3.3 Subscribe to variable event handlers

Test card	
Name/code/test case identifier	testVariableSubscription
Objectives	Test if the balancing module can subscribe to a variable in the middleware. After successful subscription the event handler in the Java wrapper needs to respond to changes in the middleware and activate balancing logic.
Devices involved	2 CMUs and 1 CMU-Server
Pre-requirements	2 instances of the middleware and the CMU-Server are running
Steps	Start CMU-Server and middleware Start balancing module Subscribe to variable and wait for callback Respond when subscription activates event handler
Expected result	Subscribe callback successful and event handler respond to new values being written to the middleware by another CMU
Additional comments	This test can be performed in parallel to testRetrieveValues

Result card	
Name/code/test case identifier	testVariableSubscription

Results	Test successful.
---------	------------------

3 Integration of the energy balancing modules

This section is based on [5].

3.1 Integration of balancing logic with device controller

The Flexiblepower Alliance Interface (FPAI) is a common interface for smart appliances. Several drivers that use this interface have been implemented by the Flexiblepower Alliance Network (FAN) and this implementation is also called FPAI. It is used as the device controller within e-balance since it is expected to become a standard in the near future. The FPAI aims to become the common language between Demand Side Management (DSM) software and appliance drivers. FPAI is an open, Java/OSGi-based platform, which distinguishes itself by the approach it uses to decouple the DSM application from the appliance drivers.

Rather than trying to come up with a single common intermediate representation that fits poorly everywhere, FPAI proposes to introduce four languages. FPAI does not describe specific device classes, such as washing machines, EVs and microCHPs, yet instead describes classes of flexibility. These flexibility classes fit conceptually between the DSM view of resources and the perspective of appliance developers. In the sense of flexibility, washing machines may be class-wise equal to EVs (both have to “run” within a certain time interval) and microCHPs may be equal to heat pumps (both describe a thermostatic control problem). The device driver developer chooses the most suitable flexibility, or control space class from the available set. For every device model, this binding needs to be developed once.

Energy applications using the control space classes should respond to a control space update with an allocation of the corresponding type. Together, the family of messages that describes the control spaces and allocations is FPAI’s Energy Flexibility Interface (EFI). The balancing module is an energy application that implements the FPAI interfaces. The balancing module (see D5.2) makes control decisions for the smart appliances that are controlled by FPAI.

3.1.1 Interface description

The balancing logic is split up into a core and a platform-specific adapter. The implementation of the balancing logic is decoupled from FPAI, and works without FPAI and OSGi (although it does need to be embedded into some environment). The balancing logic depends only on the Java standard library, and Google’s Protocol Buffers serialization library. Although there are many similarities between the set of devices which are supported by energy application platforms, the exact definition of the semantics of devices varies widely; at this moment, a shared model seems impractical. Consequently, we consider the control space specific code to be part of the platform adapter. In this way, we should be able to target the port to other (Java-based) energy application platforms and to multiple versions of FPAI, with limited risk of interference.

In Figure 5 we present the UML diagram of the balancing logic to FPAI.

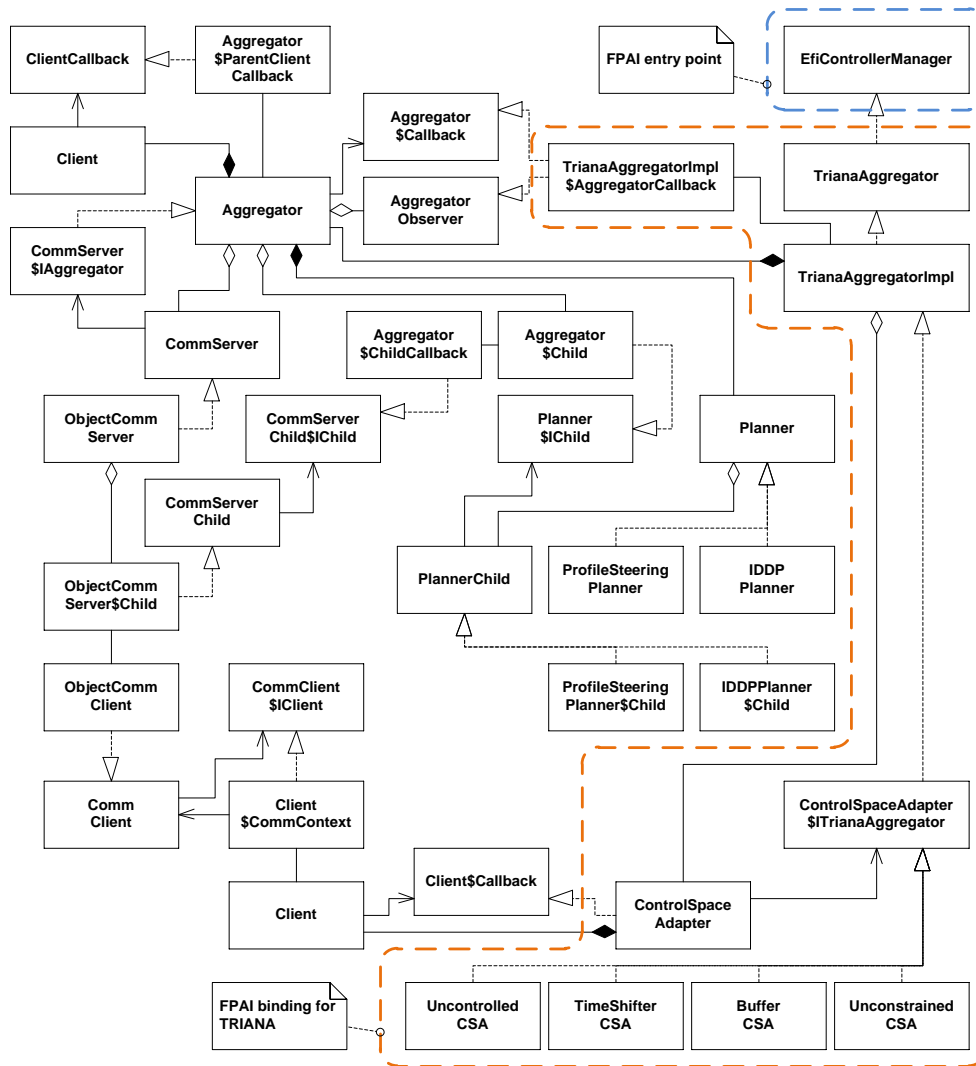


Figure 5: UML diagram of the integration of the balancing logic with FPAI

3.1.2 Tests and results

Tests were performed to test the integration of the balancing logic with FPAI. With these tests we check whether the correct FPAI control actions are given according to the optimal planning.

3.1.2.1 Testing control of an FPAI buffer device

Test card	
Name/code/test case identifier	testRunMinimum
Objectives	<p>Check if the balancing logic can create a planning for an FPAI Buffer device and can control it. Verify if the buffer reaches the minimum required on time.</p> <p>Configure FPAI with an FPAI Buffer device and use the balancing logic to make a planning. Verify if the planning reaches the minimum time the device should be on.</p>
Devices involved	CMU
Pre-requirements	FPAI is configured with a buffer device
Steps	1. The communication with FPAI is set up.

	<ol style="list-style-type: none"> 2. The balancing logic creates a planning for this device with several constraints (e.g., minimum required on time). 3. FPAI is controlled using this planning. 4. The control actions of FPAI are checked for feasibility.
Expected result	The FPAI control actions are feasible.

Result card	
Name/code/test case identifier	testRunMinimum
Results	Test successful.

3.1.2.2 **Testing planning for a FPAI buffer device**

Test card	
Name/code/test case identifier	testRunAtLowestCost
Objectives	<p>Configure FPAI with an FPAI Buffer device and use the balancing logic to make a planning. Verify if the planning is optimal.</p> <p>Check if the balancing logic can create an optimal planning for an FPAI Buffer device and can control it. Verify if the buffer is used such that the total costs are minimised.</p>
Devices involved	CMU
Pre-requirements	FPAI is configured with a buffer device
Steps	<ol style="list-style-type: none"> 1. The communication with FPAI is set up. 2. The balancing logic creates a planning for this device with several constraints (e.g., minimum required on time). 3. FPAI is controlled using this planning. 4. The control actions of FPAI are checked for optimality.
Expected result	The FPAI control actions are optimal.
Additional comments	

Result card	
Name/code/test case identifier	testRunAtLowestCost
Results	Test successful.

3.1.2.3 Testing start costs for FPAI buffer device

Test card	
Name/code/test case identifier	testStartCost
Objectives	Configure FPAI with an FPAI Buffer device and use the balancing logic to make a planning. Verify if the planning takes start-up costs (e.g., energy for starting the device) into account, such that longer runs are preferred above small runs.
Devices involved	CMU
Pre-requirements	FPAI is configured with a buffer device
Steps	<ol style="list-style-type: none"> 1. The communication with FPAI is set up. 2. The balancing logic creates a planning for this device with several constraints (e.g., minimum required on time). 3. FPAI is controlled using this planning. 4. The control actions of FPAI are checked for optimality.
Expected result	The FPAI control actions are optimal.
Additional comments	

Result card	
Name/code/test case identifier	TestStartCost
Results	Test successful

3.1.2.4 Testing planning for a FPAI timeshiftable and an increasing cost function

Test card	
Name/code/test case identifier	testPriceInc
Objectives	Configure FPAI with a device from the FPAI Timeshiftable class. Make a planning for an increasing cost function, and verify if the result is optimal.
Devices involved	CMU
Pre-requirements	FPAI is configured with a time-shiftable device.
Steps	<ol style="list-style-type: none"> 1. The communication with FPAI is set up. 2. The balancing logic creates a planning for this device with several constraints (e.g., deadline of the device) and uses a cost function that has a minimum at the beginning of the planning horizon. 3. FPAI is controlled using this planning. 4. The control actions of FPAI are checked for optimality.
Expected result	The FPAI control actions are optimal for the cost function given in this test.

Additional comments	This test and the tests below use different cost functions to avoid false positives.
---------------------	--

Result card	
Name/code/test case identifier	TestPriceInc
Results	Test successful.

3.1.2.5 Testing planning for a FPAI timeshiftable and a decreasing cost function

Test card	
Name/code/test case identifier	testPriceDec
Objectives	Configure FPAI with a device from the FPAI Timeshiftable class. Make a planning for a decreasing cost function, and verify if the result is optimal.
Devices involved	CMU
Pre-requirements	FPAI is configured with a time-shiftable device.
Steps	<ol style="list-style-type: none"> 1. The communication with FPAI is set up. 2. The balancing logic creates a planning for this device with several constraints (e.g., deadline of the device) and uses a cost function that has a minimum at the end of the planning horizon. 3. FPAI is controlled using this planning. 4. The control actions of FPAI are checked for optimality.
Expected result	The FPAI control actions are optimal for the cost function given in this test.
Additional comments	

Result card	
Name/code/test case identifier	TestPriceDec
Results	Test successful.

3.1.2.6 Testing planning for a FPAI timeshiftable and valley cost function

Test card	
Name/code/test case identifier	testPriceValley
Objectives	Configure FPAI with a device from the FPAI Timeshiftable class. Make a planning for a cost function with the minimum at the middle of the planning horizon, and verify if the result is optimal.
Devices involved	CMU
Pre-requirements	FPAI is configured with a time-shiftable device.
Steps	<ol style="list-style-type: none"> 1. The communication with FPAI is set up. 2. The balancing logic creates a planning for this device with several constraints (e.g., deadline of the device) and uses a cost function that has a minimum at the middle of the planning horizon. 3. FPAI is controlled using this planning. 4. The control actions of FPAI are checked for optimality.
Expected result	The FPAI control actions are optimal for the cost function given in this test.
Additional comments	

Result card	
Name/code/test case identifier	testPriceValley
Results	Test successful.

3.1.2.7 Testing the planning for a FPAI unconstrained device and constant cost function

Test card	
Name/code/test case identifier	testRunContinuously
Objectives	Configure FPAI with a device from the FPAI Unconstrained class. Make a planning for a constant cost, and verify if the result is optimal.
Devices involved	CMU
Pre-requirements	FPAI is configured with an Unconstrained device.
Steps	<ol style="list-style-type: none"> 1. The communication with FPAI is set up. 2. The balancing logic creates a planning for this device with a constant cost function. 3. FPAI is controlled using this planning.
Expected result	The FPAI control actions are optimal for the cost function given in this test.
Additional comments	

Result card	
Name/code/test case identifier	testRunContinuously
Results	Test successful.

3.1.2.8 Testing the planning for a FPAI unconstrained device and an intermittent cost function

Test card	
Name/code/test case identifier	testRunIntermittent
Objectives	Test if the balancing logic can plan an FPAI Unconstraint device with a cost function that toggles between high and low and control it accordingly.
Devices involved	CMU
Pre-requirements	FPAI is configured with an Unconstrained device.
Steps	<ol style="list-style-type: none"> 1. The communication with FPAI is set up. 2. The balancing logic creates a planning for this device with a cost function that toggles between high and low. 3. FPAI is controlled using this planning.
Expected result	The FPAI control actions are optimal for the cost function given in this test.
Additional comments	

Result card	
Name/code/test case identifier	testRunIntermittent
Results	Test successful.

3.2 Integration of balancing logic with the prediction module

The prediction module uses smart meter measurements to predict a power profile for the upcoming 24 hours. This power profile is used as input by the balancing logic in the CMU to determine where the flexibility needs to be deployed, and is aggregated by the LVGMU (and other management units) to determine if problems may occur in the electricity grid.

3.2.1 Interface description

At household level (CMU) smart meter power measurements are retrieved every 10 seconds via the P1 port. These values include both controllable and uncontrollable device loads. The information is used by the prediction module to predict a power profile.

The smart meter information is gathered by the balancing module and stored for a period of 24 hours in a sliding window array. This array holds 96 floating point values, each corresponding to a 15 minute time interval. After every 15 minutes the average power consumption is calculated and stored. Every 15 minutes

the balancing module sends out this array of 96 values to the prediction module through the Middleware and requests a forecast for the next 24 hours. The prediction module listens to any changes written to the Middleware through an event handler and then proceeds to create and return a new forecast. The Middleware stores this forecast. The balancing module is subscribed to changes in the Middleware through an event handler and retrieves the values. This forecast is used by the balancing algorithm to adjust the planning profile.

The integration of the balancing module and the communication through the Middleware with other modules is shown in the sequence diagram given in Figure 6.

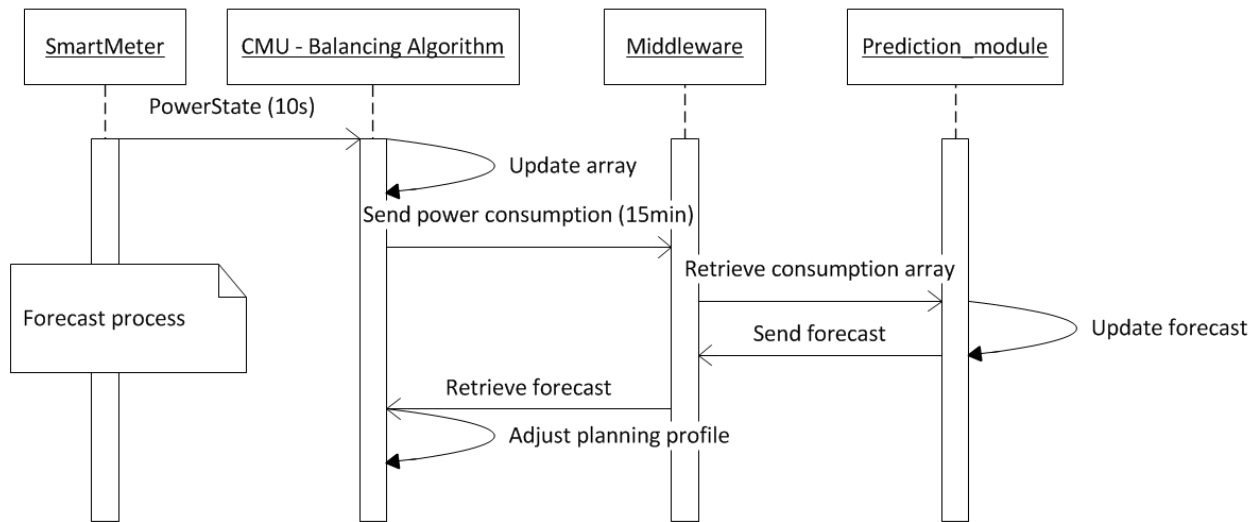


Figure 6: Sequence diagram of prediction module interaction

3.2.2 Tests and results

The following tests are used to check the integrity of the Prediction Module after integration.

In the first test below the prediction module is tested for the acceptance of 4 days’ worth of simulated consumption data. For each of the data points a prediction value should be returned

Test card	
Name/code/test case identifier	testPredictionValueAndVolume
Objectives	Tests if the Prediction Module returns 384 double values after 384 values have been input
Devices involved	1 CMU
Pre-requirements	Instance of prediction module is running, test is run independently of all other modules
Steps	Start the prediction module Select ‘Run JUnits’ when prompted. JUnits will respond whether tests were run correctly.
Expected result	For each of the 384 input doubles corresponding to four days of test data a double should be returned
Additional comments	This test does not check the value of the returned double itself

Result card	
Name/code/test case identifier	testPredictionValueAndVolume
Results	Test successful.

The additional test below checks the systems long-term integrity by having it loop over the 4 days' worth of test data until 10,000 predictions have been made. This corresponds to 104 days.

Test card	
Name/code/test case identifier	testLongTermIntegrity
Objectives	This test checks whether the prediction module continues to work over the long-term. As the neural network continues to learn with every new input value there is chance an unbalanced system occurs.
Devices involved	1 CMU
Pre-requirements	Instance of prediction module is running, test is run independently of all other modules.
Steps	Start the prediction module Select 'Run JUnits' when prompted. Run the test testPredictionValueAndVolume in a loop until 10.000 predictions have been made JUnits will respond whether tests were run correctly.
Expected result	An array of size 10.000 should be filled with prediction values
Additional comments	This test does not check the value of the returned double itself

Result card	
Name/code/test case identifier	testLongTermIntegrity
Results	Test successful.

Test Prediction Module will in practice need to learn from stored historical data and afterwards return a prediction value when a consumption value is sent. This switchover is a potential integrity problem and is checked with the following test.

Test card	
Name/code/test case identifier	testLearningAndPredicting
Objectives	This test checks whether the prediction module can learn from stored historical data stored in a CSV file and return a potentially unlimited number of prediction value (doubles)
Devices involved	1 CMU
Pre-requirements	Instance of prediction module is running, test is run independently of all

	other modules.
Steps	<p>Start the prediction module</p> <p>Select 'Run JUnits' when prompted.</p> <p>Load historical data from CSV file.</p> <p>Use this data to make the .nnet learn.</p> <p>Run the test testPredictionValueAndVolume in a loop until 10,000 predictions have been made</p> <p>JUnits will respond whether tests were run correctly.</p>
Expected result	<p>All historical examples should be used random is used.</p> <p>An array of size 10,000 should be filled with prediction values</p>
Additional comments	This test does not check the value of the returned double itself, only the integration of the software

Result card	
Name/code/test case identifier	testLearningAndPredicting
Results	Test successful.

4 Integration of the grid resilience modules

This section describes the integration of the grid resilience modules in the existing products.

4.1 Integration of LV NH Power Flow, DER Power Flows and other components

4.1.1 LV NH Power flow – Interface

The Neighbourhood Power Flows (NPF) module creates a characterisation of the LV grid operation state, in order to provide an overall grid state awareness, aiming at providing data for a GUI that is suitable for authorised users, toward improving LV grid resilience. To achieve this purpose the NPF module provides a synchronised snapshot of the LV grid conditions.

The involved architecture components are: LV Grid Management Unit (LVGMU), LV grid Sensors and Smart Meters – as described in Figure 7.

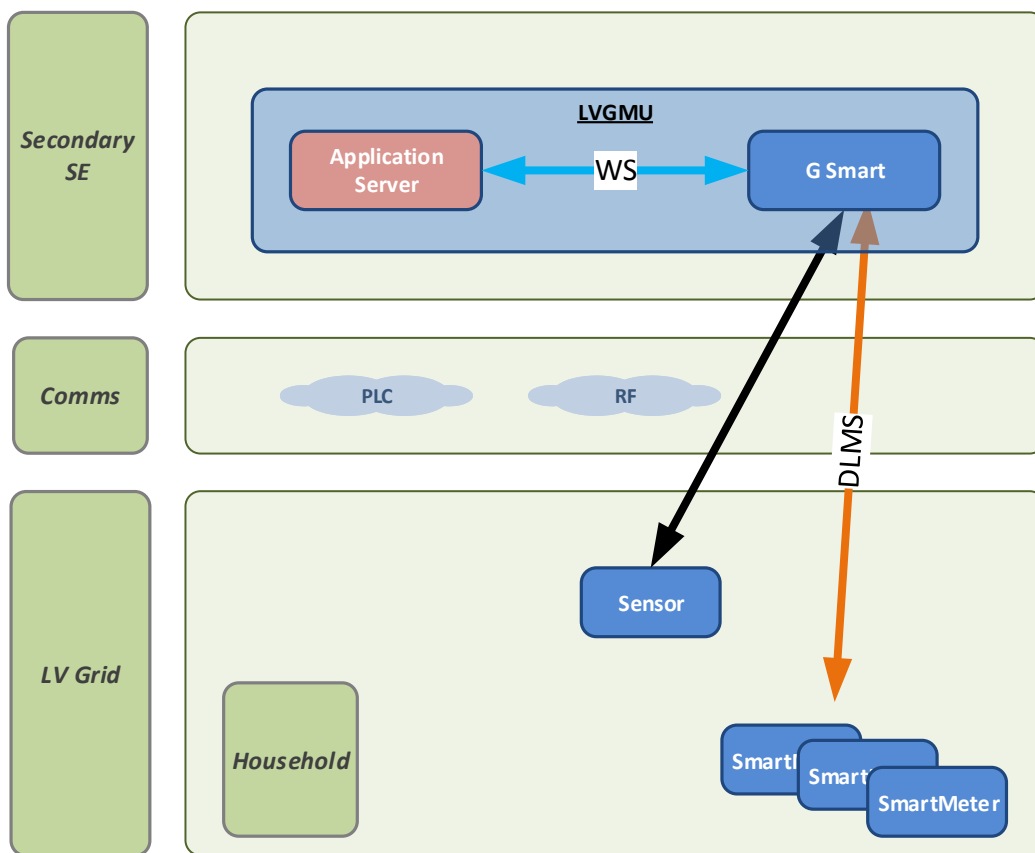


Figure 7: NH Power Flow components architecture

The LVGMU interfaces with the field interface module that is responsible for acquiring real-time or near real-time data at the field level.

The Smart Meters are geographically distributed along the feeder of the secondary substation, being deployed in a geographic neighbourhood. The communication between the Smart Meters and the secondary substation gateways are via various means of communication, like Radio Frequency (RF) or Power Line Carrier (PLC), among other communication means.

State of art Smart Meters are equipment with advanced measure capability, being capable of acquiring voltage, current, power, power factor and energy measures. Based on these measures, built-in implemented algorithms populate more complex data structures, like load diagrams, voltage diagrams, instantaneous values, events or daily energy closing reports. A load or voltage diagram is a mechanism that some Smart

Meters have included in their firmware, where measurement (energy, voltage) data can be stored cyclically (e.g. every 15 min, 30 min or 60 min). The data is stored in a non-volatile memory data device, like a FLASH RAM. The storage capability depends of the non-volatile memory size, capture cycle and number of measured channels to capture.

The LV grid Sensors spread through the LV grid can detect current alarms and transmit information data to the G Smart (communication module), which can trigger an alarm or action.

The integration between the G Smart and the Smart Meters uses the standardised Device Language Message Specification (DLMS) protocol. The integration between the G Smart and the application server uses WebServices. The G Smart receives data from the Smart Meters and transforms that data into another format according to Simple Object Access Protocol (SOAP)/eXtensible Markup Language (XML).

The application server WebServices (WS) has the capability to handle the following information data:

- Instantaneous values: measured values at the request time (Voltage, Current, Energy);
- Voltage or load diagrams: array of measures between a time internal (Voltage, Energy produced, Energy consumed).

The NPF on the LVGMU implements an Unbalanced Power Flow (UPF) algorithm, which consists of a generalised approach based on an efficient and robust three-phase branch-oriented backward-forward procedure.

The algorithm is able to deal with 4-wire unbalanced LV systems and needs time synchronised data, in order to compute results with precision. This means that the power flow cannot be executed with non-synchronised input data.

Real-time data from Smart Meters collection by the G Smart (by performing a poll over all Smart Meters) is infeasible. It is worth mentioning that the used communications, namely PLC, does not provide enough bandwidth so that synchronised time precision data could be assured for polling a large set of Smart Meters. Therefore, the used mechanism to gather data from Smart Meters is load diagrams or voltage diagrams. A load or voltage diagram comprises a specific time series of sampled data obtained or calculated by the Smart Meter, at precise intervals.

The Efacec LVGMU implements a unique algorithm that is a “LV Network Data Snapshot“, where the input is the target time and the output is a data snapshot for all the Smart Meters of the LV network.

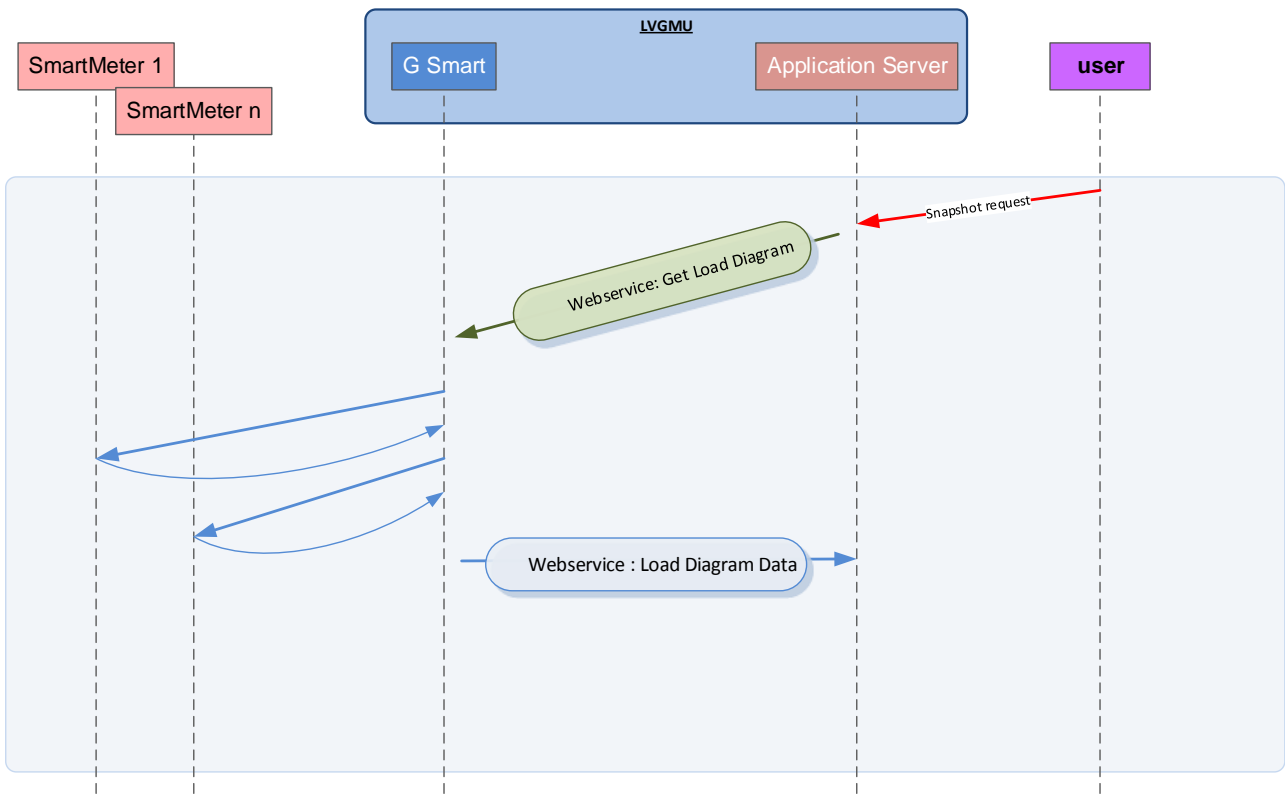


Figure 8: LV Network Data – snapshot workflow

This mechanism implemented on the LVGMU provides a snapshot with a resolution depending of the Smart Meter period cycle. For a Smart Meter configuration sample of 15 minutes, 4 synchronised snapshots per hour will be available, which can be requested anytime, as long as the necessary data remain stored on the Smart Meter. This feature represents an added advantage, related with the necessity of spending communication bandwidth only when the snapshots are requested. Figure 8 depicts the load/voltage diagram polling requests over all Smart Meters, which allow G Smart to gather all synchronised data.

A snapshot created on the LVGMU can be stored and persist for future query. These snapshots can be used just for data visualisation on a diagram or geo-referenced map, as shown in Figure 9.

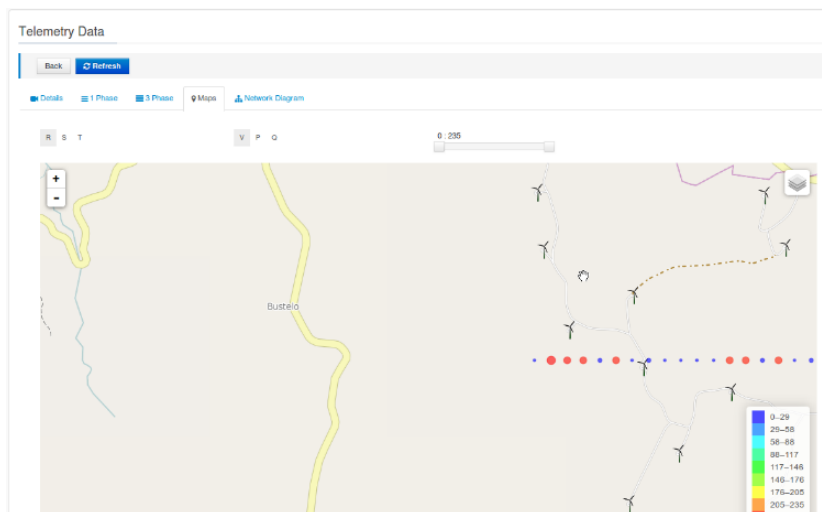


Figure 9: LV Network Data – snapshot data

Once a data snapshot is performed, as a result of received and gathered data from the Smart Meters, it can be used for enabling LV grid power flow calculations. Yet, to execute a power flow the network electric model is also necessary. The electric model describes the network configuration, at that time, identifying with detail the major components. The components capable of being configured to the power flow are:

- Secondary substation: used to specify the secondary substation considered as starting point of the electrical network;
- Transformer: used to specify a transformer in the electrical network;
- Bus-bar: used to specify a bus-bar in the electrical network;
- Node: used to specify a node in the electrical network;
- Line: used to specify a line in the electrical network, including its catalogue characteristics;
- Distribution cabinet: used to specify a distribution cabinet in the electrical network;
- Generator: used to specify a generator in the electrical network;
- Capacitor: used to specify a capacitor bank in the electrical network;
- Load: used to specify a load in the electrical network;
- Storage: used to specify an energy storage device in the electrical network;
- Switch: used to specify a circuit switch in the electrical network;
- Smart Meter: used to allocate Smart Meter equipment to a node/bus-bar/transformer /load/storage/capacitor.

The persisted snapshot data and model are the inputs to the power flow algorithm, which will be executed at the user's request or, alternatively, it can be requested by an external event trigger.

The outcome of the power flow is kept in persistent memory and can be queried for post analysis, upon user request, as described in Figure 10.

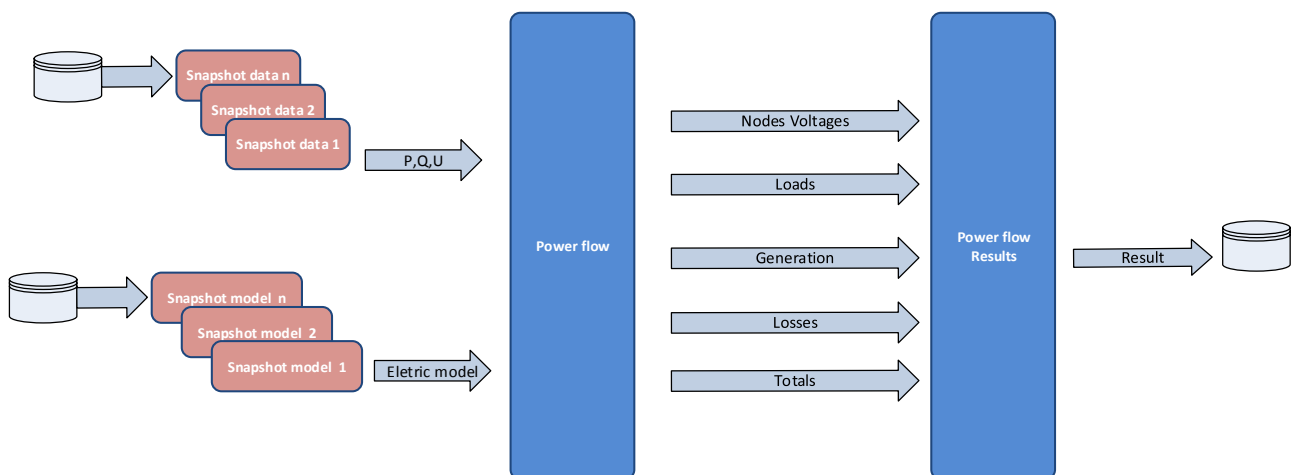


Figure 10: Power flow execution using snapshots

The output results can be visualised on the LVGMU as a data table, which can be exported to PDF format, but it also can be visualised in an electric diagram, as shown in Figure 11.

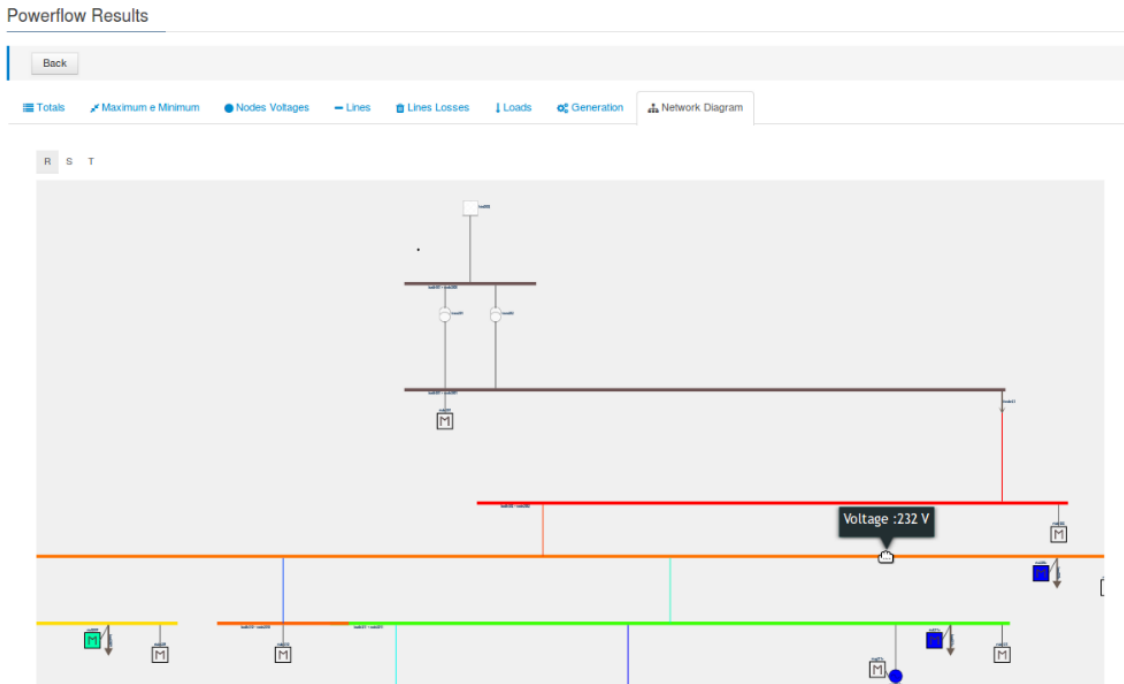


Figure 11: LV Power flow results image

The outcome results of the power flow are:

- Power flows calculation success status;
- Totals / transformer:
 - total active power (P) generation;
 - total reactive power (Q) generation;
 - total P consumption;
 - total Q consumption;
 - total P losses per-phase;
 - total Q losses per-phase;
 - maximum and minimum voltage magnitude values on phases R / S / T;
- Values on nodes / bus-bars:
 - voltage;
- Line results:
 - rating;
 - current;
 - P and Q losses;
 - voltage;

4.1.2 LV NH Power flow

The integration between the Application server and G Smart is done using WebServices, on a Hypertext Transfer Protocol (HTTP) using a client/server approach and FTP.

The building blocks of the LVGMU, as demonstrated in Figure 12, are composed by the following blocks:

- DataBase Management System (DBMS): is the relational database for data model persistence;
- Time Series DataBase (TSDB): is the time series database, where all the received field data is stored in persistent memory;
- APP: is the business logic layer, where all the logic is implemented;
- FTP: is the FTP server to receive the data reports from the G Smart;
- Webserver: HTTP web server that will implement services and the GUI;
- Webservice (WS) Client: Client WebServices to connect to the G Smart web server.

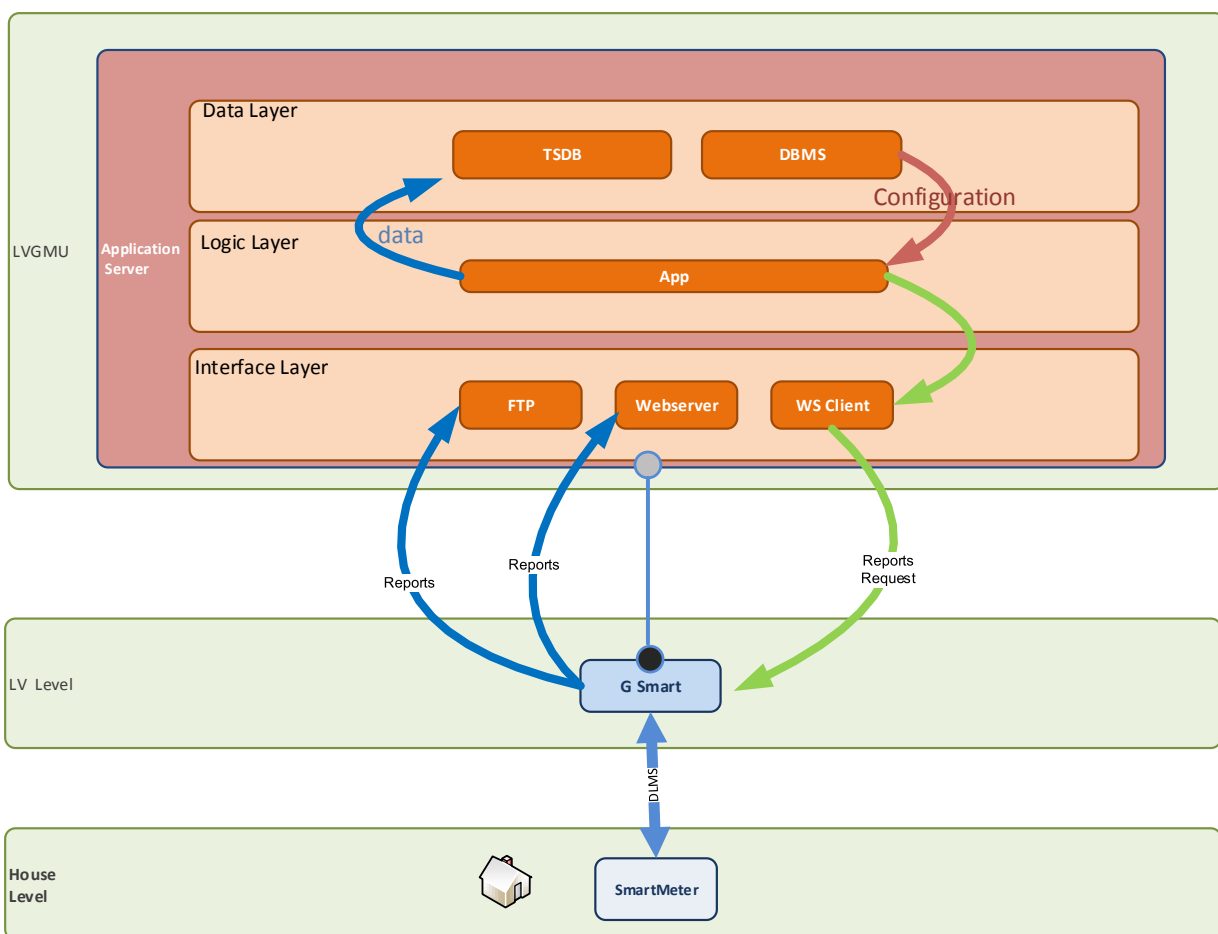


Figure 12: LVGMU application blocks

The DBMS module is the data store for the main configuration data, like the network model, and is also important for logging the system activity. The configuration data will have items such as:

- asset configuration;
- network connectivity configuration;
- relations between network topology elements;
- user activity audit log

The TSDB is the data store for time series data. The data points that are stored have origin on Smart Meters' load diagrams.

The APP block is the business logic implemented in the application server. An application server is a software framework that provides services and Application Programming Interface (API), in order to implement business logic. In this case, the business logic is implemented using the J2E API. The implemented business logic has the following services:

- Database access with persistency API;
- Database cache system;
- Internet Protocol version 4 (IPv4) high performance data memory cache;
- Application logic interface for web services;
- Application logic to compute the snapshot;
- Application logic to compute the power flow;

The power flow logic is implemented in C language and the APP block is developed in Java language, so it was necessary to bind the C language to Java, using the JNA (Java Native Access). Figure 13 depicts such integration.

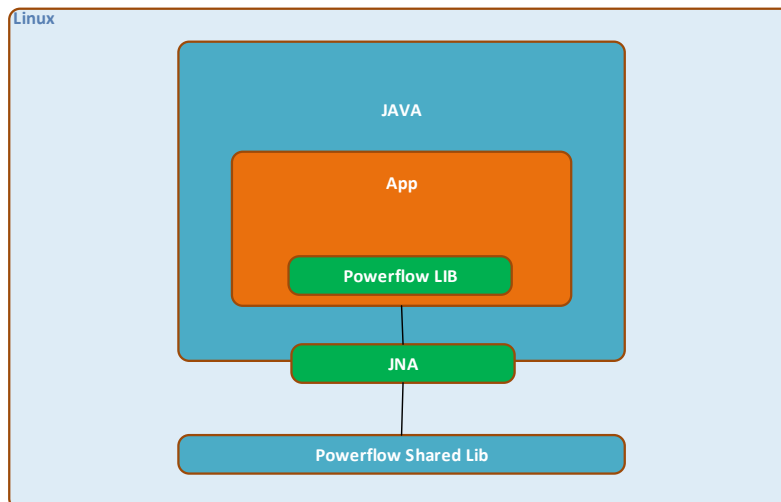


Figure 13: Power Flow C/Java Integration

The FTP block is a common FTP server. In this case, the FTP server is used on Linux OS and it will be used to receive XML reports from the G Smart. When the LVGMU requests data from the G Smart – the request can be asynchronous – the response will be sent to the FTP server repository.

The webserver block has two types of interfaces: GUI and interfaces to other components. For the GUI, the webserver implements the JSF web specification in order to implement and deliver the GUI web pages to the LV grid operator.

The webserver implements technical interfaces for:

- SOAP WebServices (WS): this technology is used to implement the interface between the core logic and the G Smart equipment. It implements an API to:
 - Report status;
 - Receive report.

The WS clients are the frameworks that provide data access to the G Smart webserver. For the G Smart, it implements the API to request a load diagram from a Smart Meter.

4.1.3 LV NH Power flow – Tests and results

The following tests were performed with the LV power flow on the LVGMU. The tests were executed with LV network schematic designed only for test purpose.

4.1.3.1 Testing the LV network import

Test card	
Name/code/test case identifier	ImportLVSchema
Objectives	Import a LV network schema for a LV test network
Devices involved	LVGMU
Pre-requirements	Have a network schema
Steps	On LVGMU: <ul style="list-style-type: none"> • Go to Energy -> Powerflow • Click on Import data, and import the schema
Expected result	Result on screen shall be success
Additional comments	-

Result card	
Name/code/test case identifier	ImportLVSchema
Results	Successful import of a network with : 8 Smart Meters, 14 nodes, 14 bus-bar, 2 transformers, 1x feeder, 11 lines, 17 loads, 8 generators, 1 energy storage

4.1.3.2 Testing the LV network Snapshot creation

Test card	
Name/code/test case identifier	CreateLVSnapshot
Objectives	To create a snapshot of the LV network, for the model and data
Devices involved	LVGMU, G Smart
Pre-requirements	LVGMU shall have the Secondary substations configured, the LV network imported and an established connection to a G Smart
Steps	On LVGMU: <ul style="list-style-type: none"> • Go to Energy -> Powerflow -> Secondary Sub. list • Choose the secondary substation, choose create snapshot and choose the snapshot time.
Expected result	Result on screen shall be success.
Additional comments	-

Result card	
Name/code/test case identifier	CreateLVSnapshot
Results	Success

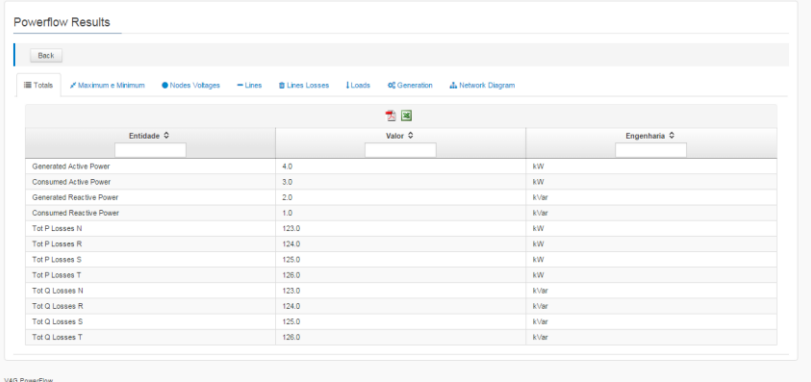
4.1.3.3 Testing the LV network Snapshot visualisation

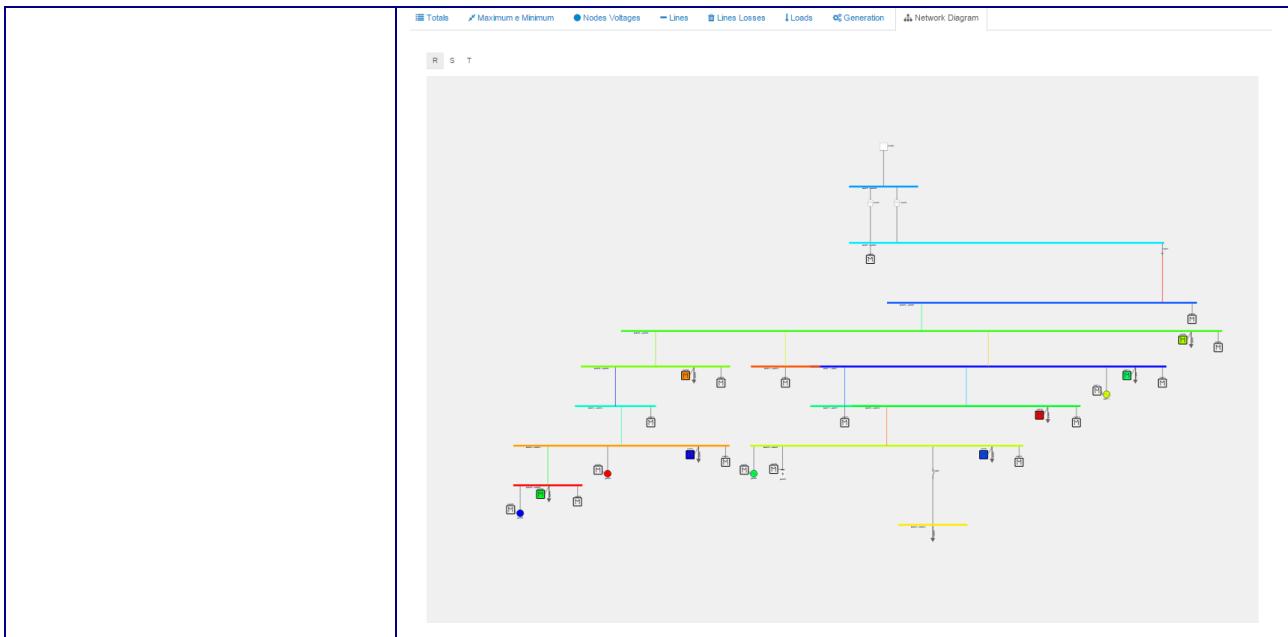
Test card	
Name/code/test case identifier	VisualiseSnapshot
Objectives	After a snapshot was created the data can be visualised on LVGMU
Devices involved	LVGMU
Pre-requirements	A snapshot shall be created with success
Steps	On LVGMU: <ul style="list-style-type: none"> • Go to Energy -> Powerflow -> Network snapshot • Choose a snapshot and click start. When the execution state shows it has finished, click on telemetry data
Expected result	Result on screen shall be success.
Additional comments	-

Result card	
Name/code/test case identifier	VisualiseSnapshot
Results	Success

4.1.3.4 Testing the LV network Power flow creation and visualisation

Test card	
Name/code/test case identifier	Snapshot4PowerFlow
Objectives	After a snapshot was created the snapshot can be the input for the power flow.
Devices involved	LVG MU
Pre-requirements	A snapshot shall be created with success
Steps	On LVGMU: <ul style="list-style-type: none"> Go to Energy -> Powerflow -> Network snapshot Choose a snapshot and click power flow.
Expected result	Result on screen shall be success. The power flow visualization shall have data for the power flow results in table format and in schematic format. The power flow shall converge.
Additional comments	-

Result card																																								
Name/code/test case identifier	Snapshot4PowerFlow																																							
Results	Success the power flow did converge. Image of a snapshot result in table format:  <p>The screenshot shows a table with the following data:</p> <table border="1"> <thead> <tr> <th>Entidade</th> <th>Valor</th> <th>Engenharia</th> </tr> </thead> <tbody> <tr><td>Generated Active Power</td><td>4.0</td><td>kW</td></tr> <tr><td>Consumed Active Power</td><td>3.0</td><td>kW</td></tr> <tr><td>Generated Reactive Power</td><td>2.0</td><td>kVar</td></tr> <tr><td>Consumed Reactive Power</td><td>1.0</td><td>kVar</td></tr> <tr><td>Tot P Losses N</td><td>123.0</td><td>kW</td></tr> <tr><td>Tot P Losses R</td><td>124.0</td><td>kW</td></tr> <tr><td>Tot P Losses S</td><td>125.0</td><td>kW</td></tr> <tr><td>Tot P Losses T</td><td>126.0</td><td>kW</td></tr> <tr><td>Tot Q Losses N</td><td>123.0</td><td>kVar</td></tr> <tr><td>Tot Q Losses R</td><td>124.0</td><td>kVar</td></tr> <tr><td>Tot Q Losses S</td><td>125.0</td><td>kVar</td></tr> <tr><td>Tot Q Losses T</td><td>126.0</td><td>kVar</td></tr> </tbody> </table>	Entidade	Valor	Engenharia	Generated Active Power	4.0	kW	Consumed Active Power	3.0	kW	Generated Reactive Power	2.0	kVar	Consumed Reactive Power	1.0	kVar	Tot P Losses N	123.0	kW	Tot P Losses R	124.0	kW	Tot P Losses S	125.0	kW	Tot P Losses T	126.0	kW	Tot Q Losses N	123.0	kVar	Tot Q Losses R	124.0	kVar	Tot Q Losses S	125.0	kVar	Tot Q Losses T	126.0	kVar
Entidade	Valor	Engenharia																																						
Generated Active Power	4.0	kW																																						
Consumed Active Power	3.0	kW																																						
Generated Reactive Power	2.0	kVar																																						
Consumed Reactive Power	1.0	kVar																																						
Tot P Losses N	123.0	kW																																						
Tot P Losses R	124.0	kW																																						
Tot P Losses S	125.0	kW																																						
Tot P Losses T	126.0	kW																																						
Tot Q Losses N	123.0	kVar																																						
Tot Q Losses R	124.0	kVar																																						
Tot Q Losses S	125.0	kVar																																						
Tot Q Losses T	126.0	kVar																																						
	Image of a snapshot result in schematic format:																																							



4.2 Integration of self-healing FDIR and communication platform

When a fault occurs in an MV feeder, an overcurrent protection commands the substation circuit breaker disconnecting the feeder in order to isolate the faulted area. Automatic reclosing functions at the substation are then triggered in order to eliminate temporary faults.

If the fault subsists after the reclosing cycle then the Fault Detection, Isolation and Restoration (FDIR) function should run in order to locate the fault, isolate the minimum area that encloses the fault, and then restore service to clients that are outside the isolated area.

This section describes the integration tests between Medium Voltage Grid Management (MVGGMU) and Top Level Grid Management Unit (TLGGMU) within the scope of FDIR functionality.

4.2.1 Interface description

The FDIR function is triggered at the end of the reclosing cycle, and runs only if the tripped feeder breaker remains open.

This function uses information coming from fault detectors deployed along the network, supplied to MVGGMU as telemetered data points. FDIR joins this information with the knowledge of the network topology in order to locate the fault. Then FDIR executes commands to open tele-controlled switches that will isolate the fault.

The upstream service restoration, accomplished by simply closing the tripped breaker, is always a safe operation since it is assumed that the load to be restored upstream the fault is smaller than the load that has been cut by the breaker trip.

Downstream restoration is a more complex operation as requires the estimation of the load downstream the fault and the knowledge of the availability of backup lines and transformers to supply energy to clients in that area. The relevant data, like active power (P) and reactive power (Q) at backup transformers, is delivered to the MVGGMU as telemetered data points. This information, together with the network data model knowledge, let the MVGGMU validate alternative configurations to supply energy to clients downstream the fault, by running a DOPF algorithm. Once the final network configuration is computed, MVGGMU commands available tele-controlled switches in order to reconfigure the network.

After running the FDIR algorithm and reconfiguring the network, the MVGGMU exports information about the performed actions to the TLGGMU, in order to let Dispatch operators know that the fault has been

automatically handled. This interface to the TLGMU supports the following features at the operator's workstation:

- In the alarm list, the alarm for a breaker trip event that has been handled by FDIR function at MVGMU shows a "FDIR" icon. This informs the operator that the fault that caused the breaker trip has been automatically located and isolated by the MVGMU and that the service has already been restored to clients outside the isolated area;
- From the "FDIR" icon in the alarm list, the operator is able to consult the actions that have automatically been performed by the FDIR function running at the MVGMU. This includes the switching order corresponding to the network reconfiguration and the identification of fault locators that became active after the fault.

4.2.2 Electrical network topology

The tests are performed on the 'Batalha' feeder of the 'S. Jorge' HV/MV Substation.

The network topology relevant for the tests is presented in Figure 14. In this figure the Normally Open switches are represented as red squares while Normally Closed switches are represented as green squares. All represented switches are tele-controlled and have fault detectors.

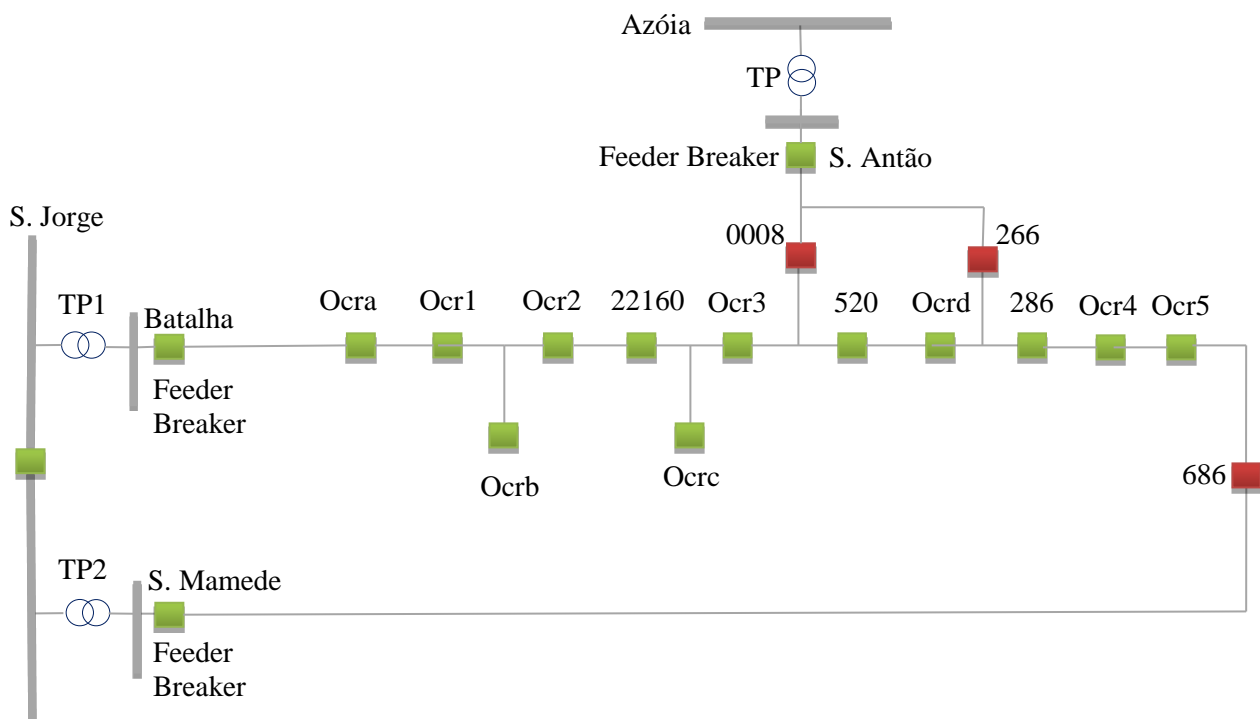


Figure 14: Relevant electrical network for FDIR tests

The system was modelled according to the electrical characteristics of the equipment existing in these substations and feeders. Here is present the most relevant modelled data:

- Transformers Load Capacity:
 - S. Jorge: TP1 = 40 MVA, TP2 = 40 MVA;
 - Azóia: TP = 20 MVA,

4.2.3 Tests and results

The tests are performed running scripts which simulate changes in telemetered data points.

Test card	
Name/code/test case identifier	OutOfService
Objectives	Verify that when the FDIR function is Out Of Service, no action is automatically performed by MVGMU after a breaker trip
Devices involved	MVGMU, TLGMU
Pre-requirements	Operator logged in at TLGMU workstation FDIR function initially out of service
Steps	1 - Open the “S. Jorge Automatismos” and the network schematic diagrams. 2 - Run a telemetry script which simulates (by this order) 1 - Begin of Batalha feeder breaker reclosing cycle 2 - Batalha breaker trip 3 - (close breaker / open breaker – recloser automatism) 4 - Active fault indicators: Ocra, Ocr1 5 - End of Batalha feeder breaker reclosing cycle
Expected result	- The FDIR status indication shows “Out of Service” - No FDIR icon is shown next to the alarm relative to the Batalha feeder breaker trip - No network reconfiguration took place after the breaker trip
Additional comments	

Result card	
Name/code/test case identifier	OutOfService
Results	Test successful.

Test card	
Name/code/test case identifier	In/OutOfServiceControl
Objectives	Verify that from a TLGMU workstation it is possible to control whether the FDIR function of MVGMU is in or out of service
Devices involved	MVGMU, TLGMU
Pre-requirements	Operator logged in at a TLGMU workstation FDIR function initially out of service
Steps	1 - Open the “S. Jorge automatismos” synoptic and identify the FDIR “Out of Service” status indication. 2 – Click with mouse right button on the status indication, choose the “In service” option and execute the action from the displayed command window.
Expected result	The MVGMU’s FDIR status indication shows “In Service”
Steps	Click with right button on the status indication, choose the “Out of service” option and execute the action from the displayed command window.
Expected result	The FDIR status indication shows “Out of Service”
Additional comments	

Result card	
Name/code/test case identifier	In/OutOfServiceControl
Results	Test successful.

Test card	
Name/code/test case identifier	Fault-Ocr1-Ocr2
Objectives	Verify correct fault location, isolation and restoration when a fault occurs in a line segment between Ocr1 and Ocr2 switches
Devices involved	MVGMU
Pre-requirements	<p>FDIR function In Service All feeder breakers closed. All switches in normal position. Feeders Load (telemetry data simulation): S.Jorge TP1: 11.9 MW, 3.6 MVar, S.Jorge TP2: 9.8 MW, 3.5 MVar Azóia TP: 6.1 MW, 3.9 MVar Batalha: 3.0 MW, 0.8 MVar, 56.5 A S.Mamede: 5.2 MW, 1.4 MVar, 92.5 A S.Antão: 2.8 MW, 0.9 MVar, 53.5 A</p>
Steps	<p>Run a telemetry script which simulates (by this order)</p> <ol style="list-style-type: none"> 1 - Begin of Batalha feeder breaker reclosing cycle 2 - Batalha breaker trip 3 - (close breaker / open breaker – recloser automatism) 4 - Active fault indicators: Ocr1, Ocr2 5 - End of Batalha feeder breaker reclosing cycle
Expected result	<p>The following switching actions have been automatically performed on the network:</p> <ol style="list-style-type: none"> 1 – Open Ocr1, Open Ocr2, Open Ocrb (fault isolation) 2 – Close Batalha feeder breaker (upstream restoration) 3 – Close 668 (downstream restoration) <p>The network configuration should be as follows:</p> <ul style="list-style-type: none"> - Ocr1-Ocr2 fed by S.Jorge TP1 through Batalha feeder - Ocr2 to 668 fed by S.Jorge TP2 through S.Mamede feeder - Ocr1-Ocrb-Ocr2 isolated
Additional comments	

Result card	
Name/code/test case identifier	Fault-Ocr1-Ocr2
Results	Test successful.

Test card	
Name/code/test case identifier	SwitchingOrder
Objectives	Verify that the details of the actions automatically performed at MVGMU can be consulted by an operator at a TLGMU workstation.

Devices involved	MVGMU, TLGMU
Pre-requirements	Run test scenario Fault-Ocr1-Ocr2
Steps	In the alarm list of TLGMU workstation verify that the Batalha breaker trip alarm has an “FDIR” icon. Press the mouse right button on this icon and choose the “View switching order”
Expected result	A window is displayed showing the following information: - Identification of the feeder breaker that tripped - Trip time - Switching order showing the actions automatically executed by MVGMU: o RESET fault indicators for Ocr1 and Ocr2 switches o Network reconfiguration actions: ▪ Open Ocr1, Open Ocr2, Open Ocr3 (fault isolation) ▪ Close Batalha feeder breaker (upstream restoration) ▪ Close 668 (downstream restoration) Each switching order line shows the date when the corresponding action was automatically executed by MVGMU.
Additional comments	

Result card	
Name/code/test case identifier	SwitchingOrder
Results	Test successful.

4.3 Integration of VOS, OPF and communication platform

4.3.1 Interface description

The Optimized Power Flow (OPF) module’s main objective is to determine the optimal MV grid topology, which minimises the grid active power losses and consequently it’s operating costs. The primary task is to find a set of system states, including switch state, within a region defined by the operating constraints such as voltage limits and branch flow limits. The secondary task is to optimise a cost function within this region. Typically, in OPF, the dispatch of active power is considered constant, which means that the power input from the transmission grid and the distributed power generation, are considered constant. The results obtained by this module must comply with operational constraints such as equipment operating limits, system security limits and radial operation of the grid.

Four modes of operation defined for the OPF module can be selected by the distribution system operator in the configurations parameters application, namely:

- Mode 1 – Reconfiguration mode that searches for the optimal grid configuration. In this mode, the OPF module will consider the available switches in order to determine the optimal grid topology.
- Mode 2 – Power flow optimization mode, which determines the optimal state variables without considering the reconfiguration of the distribution grid (i.e. switches are not considered controllable variables). In this case, the OPF will provide the optimal states for transformers and capacitor banks taps and the coordination with DER units in order to ensure that the MV grid voltages are within limits.
- Mode 3 – Runs both reconfiguration and power flow optimization. This mode combines the optimization objectives of Mode 1 and Mode 2.

- Mode 4 — Restoration mode, where the OPF will find alternative grid configurations which minimise the power not supplied after fault isolation.

Depending on the OPF mode of operation, the grid configuration will be optimised considering an objective function, which is a definition of how the solution state is to be evaluated and includes a mean of penalizing small changes of controls, in order to avoid unnecessary changes. Two different objective functions can be selected:

- Minimise the sum of active power losses (total or specifically for a given MV grid).
- Minimise the sum of active power not supplied. This objective function is selected in Mode 4.

Within the scope of this section, the integration tests are focused mainly on OPF and Validation of Optimised Solutions (VOS) functions concerning the minimisation of active power losses and user interaction. The minimisation of power not supplied is also covered in section 4.2.

The main objective of the Validation of Optimised Solutions (VOS) application is to determine a reconfiguration procedure according to the optimal reconfiguration scheme determined by the OPF module. OPF only determines the actions that must be made to improve the actual configuration and not necessarily the correct order of doing it. Therefore, the VOS module is responsible for determining an automated reconfiguration sequence, ensuring the operational safety of the distribution grid during the sequence steps. From the user perspective, OPF calls the VOS module transparently.

The solution found by the algorithm is presented in a tabular form in results application. It is possible to navigate from this tabular to others application like browser and diagrams with the tool “Go To Equipment”.

4.3.2 Underlying Communication

The system uses the communication platform (CP) on top of relevant communication protocols (see Figure 15).

The CP offers a publish / subscribe paradigm to link software components, hiding the individual connections between applications from the components, transforming the logical design of the system. The following diagram gives an idea of how this changes the design process, each component being designed as a consumer or provider of bus based services.

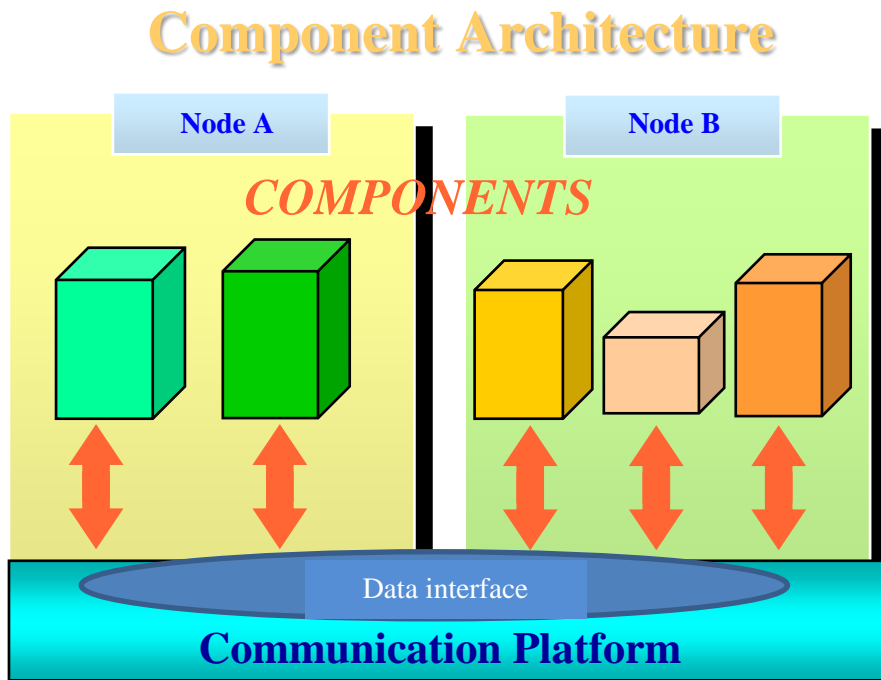


Figure 15: Component architecture

The information is exchanged between components in the form of events, these events being defined in the system.

The events are published by one component to inform any subscribing component of the changes originating in or processed by the publishing component. This allows the subscribing component to update itself. However, this does not provide a synchronised 'initial state' where the subscriber can start from. A DataRequest is, as the name implies, a request for data and thus a DataRequest naturally is made by a subscriber and processed by a publisher. A DataRequest can be viewed as a data service.

The subscribing component may make a DataRequest for two reasons:

- to collect the actual information for a limited set of objects without trying to maintain a model covering all possible objects, for example a tabular interface showing objects following a user defined filter
- to synchronise a complete model to enable the subscribing component to perform its function well synchronised with the rest of the system

4.3.3 Tests and results

Integration testing was supported by a simplified network configuration scheme created specifically for this objective.

Test card	
Name/code/test case identifier	Restoration
Objectives	Obtain a list of controls that minimise the power not supplied after fault isolation.
Devices involved	TLGMU
Pre-requirements	Operator logged in at TLGMU workstation

Steps	1 - Open the configuration's power application 2 - Run a "Restoration" considering the equipment fault "Line 1" of network test.
Expected result	- The table of results shows a sequence of controls that allow performing the minimisation of the power not supplied after fault isolation.
Additional comments	

Result card	
Name/code/test case identifier	Restoration
Results	Test successful.

Test card	
Name/code/test case identifier	Power flow optimization mode
Objectives	Obtain a list of optimal states for transformers and capacitor banks taps
Devices involved	TLGMU
Pre-requirements	Operator logged in at a TLGMU workstation
Steps	1 - Open the configuration's power application. 2 - Choose the options: <ul style="list-style-type: none"> o Minimise the sum of active power losses o Control transformers and capacitor bank taps
Expected result	- The table of results shows the list of controls that allow achieving the optimal states for transformer and capacitors bank taps.
Additional comments	

Result card	
Name/code/test case identifier	Power flow optimization mode
Results	Test successful.

Test card	
Name/code/test case identifier	Reconfiguration
Objectives	Obtain a list of controls that determine the optimal grid topology.
Devices involved	TLGMU
Pre-requirements	Operator logged in at a TLGMU workstation
Steps	1 - Open the configuration's power application. 2 - Choose the option "Reconfiguration" <ul style="list-style-type: none"> o minimise the sum of active power losses o control only switches.
Expected result	- The table of results shows the sequence of switching actions allowing to move safely to the new optimised grid topology
Additional comments	

Result card	
Name/code/test case identifier	Reconfiguration
Results	Test successful.

Test card	
Name/code/test case identifier	Navigation
Objectives	Navigate between windows
Devices involved	TLGMU
Pre-requirements	Operator logged in at a TLGMU workstation
Steps	1 - Open the results table. 2 – Select the tool “Navigate to equipment” 3 – Choose the equipment in the results table. 4 - Press the mouse right button on this equipment and choose the “Apply Equipment” 5 – Go to the diagram, press the mouse right button and choose “Apply Application.”
Expected result	- The chosen equipment is selected in the diagram.
Additional comments	

Result card	
Name/code/test case identifier	Navigation
Results	Test successful.

Test card	
Name/code/test case identifier	Highlight
Objectives	Find in the results tabular the equipment
Devices involved	TLGMU
Pre-requirements	Operator logged in at a TLGMU workstation
Steps	1 – In the diagram choose the equipment. 2 – Select the tool “Highlight” and choose the desired colour. 3 - Press the mouse right button on this equipment and choose the “Apply Highlight” option. 5 – Go to the table results and check that the line corresponding to the same equipment is highlighted in the same colour (first column background colour).
Expected result	- The line corresponding to the same equipment is highlighted in the same colour (first column background colour).
Additional comments	

Result card	
Name/code/test case identifier	Highlight
Results	Test successful.

4.4 Integration of LV Grid Resilience Modules

4.4.1 LV Fault Management - LV Fault Prevention - Dynamic Voltage Control

4.4.1.1 Interface description

A dynamic voltage control algorithm, for Low Voltage (LV) power distribution grids to which various micro and mini producers are connected – designated in the literature as Distributed Generators (DG) – was previously described in D5.3.

The objective of the algorithm is to maximise the DG production, but it may also enforce fairness among DG producers, taking into account the contract limitations with power grid operators or regulators and while keeping voltage levels within the standard operational limits in all coupling points of the power grid.

This algorithm will be implemented in an external PC, conceptually seen as an extension of the Efacec's implementation of the LVGMU (G Smart), as described in section 5.1.2 of D6.1, and shown with adaptations in Figure 16.

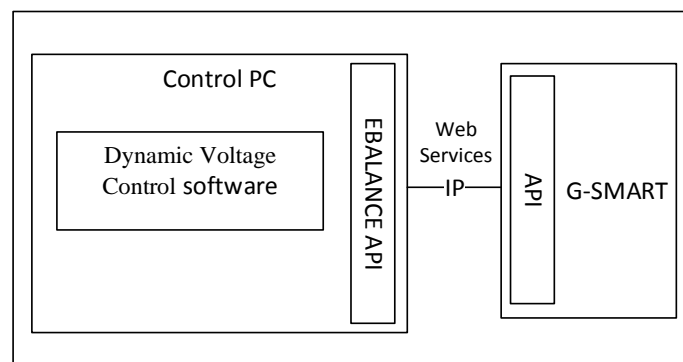


Figure 16: Efacec's "G Smart" extended with Dynamic Voltage Control software on a PC

The following WebServices are defined in the interface between the Control PC and the G Smart:

- Request data values from meters (S01):
This service contains as parameters one or more meter IDs and returns the Voltage, Current and Power Factor measured by the meters who's IDs were included in the request.
- Send set points to Inverter (B35):
This service contains as parameters one or more inverter IDs and the correspondent set points to change the power produced by each one of the identified Inverters.
- Read produced power from Inverter (S35):
This service contains as parameters one or more inverter IDs and obtains as response the power currently produced by each one of the identified Inverters.

An example of the utilization of WebServices in the integration of the Dynamic Voltage Control (DVC) module with the G Smart is shown in the sequence diagram of Figure 17.

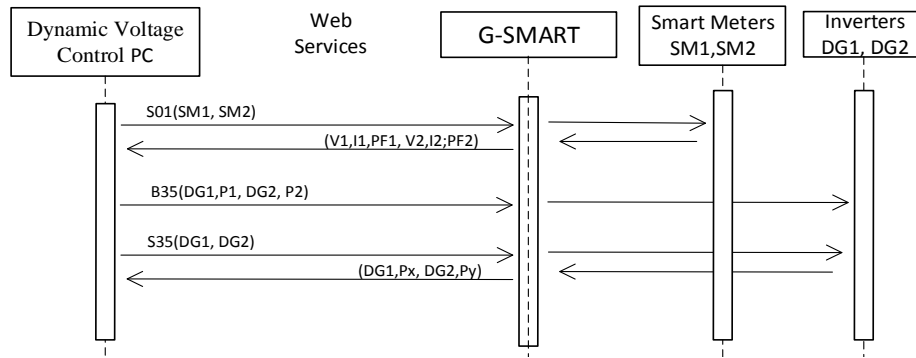


Figure 17: Sequence diagram of the communication between the Voltage Control PC and the G Smart

In the first message (S01) the Control PC asks the Smart Meters connected at each output of the Distributed Generators DG1 and DG2 the respective Voltage, Current and Power Factor values.

Based on the values read from the Smart Meters the Control PC calculates the power generated at each DG and decides the set point values which should be sent in order to evaluate how the electrical network will behave. This is done using the service B35, with set points P1 and P2 respectively for DG1 and DG2.

To check if the set point commands were correctly received by the DGs, the service S35 can be used, as shown in Figure 17.

This sequence of messages is repeated until the Control PC obtains all the information needed, as described in the Dynamic Voltage algorithm in D5.3.

4.4.1.2 Tests and results

The first set of tests was performed to assess the correct communication between the Control PC and the G Smart WebServices. All services are working well.

The next sequence of tests was performed to test the response of the inverters to the set points received. These tests showed a slight difference between the values sent and returned, requiring a calibration of the measurements.

4.4.2 Integration of LV Fraud, LV Quality and LV Fault Management modules

The LV Fraud, LV Quality and LV Fault Management modules were developed and integrated in the G Smart software, as shown in Figure 18.

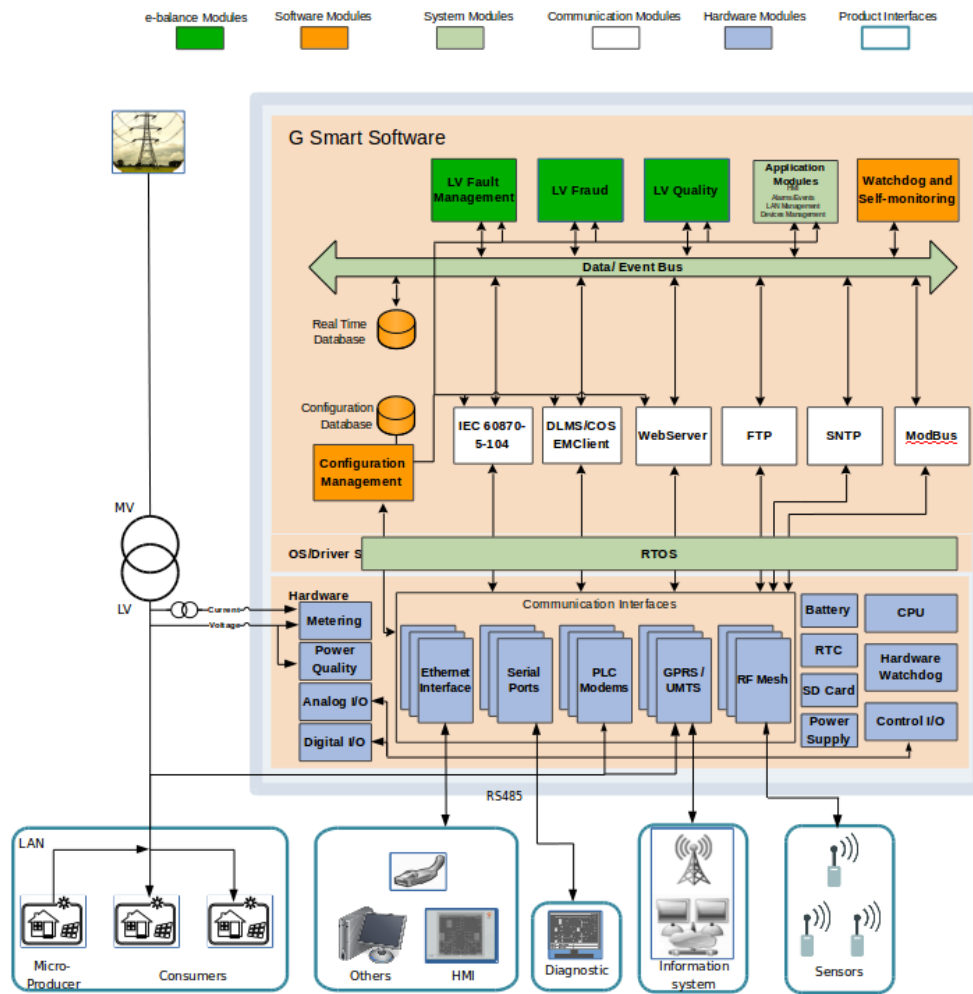


Figure 18: Integration of grid resilience modules in the G Smart

4.4.2.1 Interface description

The G Smart grid resilience modules have as inputs data from the LV Sensors and Smart Meters, which are spread through the LV Network. The used protocol for the communication between those LV devices (sensors or Smart Meters) and the G Smart is the DLMS/COSEM over GPRS, PLC or RF Mesh as the physical media.

Internally the G Smart uses a real time data and an event bus as means for the internal application software modules. Through this bus it is possible to receive or transmit data or events between all internal modules. The G Smart internal bus abstracts all protocol communication details, such as: DLMS/COSEM, NTP, WebServices or others that can be used. A WebServices interface over TCP/IP is also available for data read or write and on event processing.

The grid resilience modules need the following input data from the LV Devices (sensors and Smart Meters):

Inputs	Description
Current Inst. L1	Current instantaneous value in the phase L1
Voltage Inst. L1	Voltage instantaneous value in the phase L1
Active Power Inst. L1	Active power instantaneous value in the phase L1
Current Inst. L2	Current instantaneous value in the phase L2
Voltage Inst. L2	Voltage instantaneous value in the phase L2
Active Power Inst. L2	Active power instantaneous value in the phase L2
Current Inst. L3	Current instantaneous value in the phase L3
Voltage Inst. L3	Voltage instantaneous value in the phase L3
Active Power Inst. L3	Active power instantaneous value in the phase L3

Capture Time	Capture time of the last current streetlight average value
Current Streetlight	Last current streetlight average value
Voltage Streetlight	Last voltage streetlight average value
Clock	Clock
Load Profile	Load Profile
Billing Profile	Daily Billing
QoS data	Smart Meters quality of service data

The module outputs are written in the internal Real Time Database and are available for WebServices and local GUI, which may represent local or remote clients. The output data is:

Outputs	Description
Alarms	Alarms from the grid resilience modules
QoS data	QoS data and reports
LV Measures	LV Measurements of grid variables – e.g. voltage, current, etc.
Setpoint controls	Setpoint controls from voltage control algorithm

4.4.2.2 Tests and results

Test card	
Name/code/test case identifier	MeasurementRequest
Objectives	Test measurements requests
Devices involved	LVGMU
Pre-requirements	Configure one LV sensor and one Smart Meter; Configure grid resilience modules to poll sensors and Smart Meters measurements every minute.
Steps	Open the device debug console to monitor the communication trace; Change the acquired measured values in the Smart Meter and Sensors.
Expected result	In the communication trace, the measurements are correctly requested and processed; In the real time database, the measurements are updated; In the grid resilience modules (through debug messages), the measurements are received correctly; The GUI must show the correct measurements update.
Additional comments	

Result card	
Name/code/test case identifier	MeasurementRequest
Results	Test successful.

Test card	
Name/code/test case identifier	LVFaultManagement
Objectives	Test the LV Fault Management module
Devices involved	LVGMU
Pre-requirements	Configure one LV sensor.
Steps	Open the device debug console to monitor the communication trace; Change the acquired measures values in the sensor to induce an alarm in the LV Fault Management module; Open the GUI and see if the alarm was generated.
Expected result	The GUI must indicate that the new alarm is generated; In the real time database, the alarms are generated.
Additional comments	

Result card	
Name/code/test case identifier	LVFaultManagement
Results	Test successful.

Test card	
Name/code/test case identifier	LVQuality
Objectives	Test the LV Quality module
Devices involved	LVGMU
Pre-requirements	Configure one Smart Meter
Steps	Open the device debug console to monitor the communication trace; Change the acquired measures values in the Smart Meter to induce a set of QoS events; Open the GUI and see if the QoS alarms were generated.
Expected result	The GUI must indicate that the new alarm is generated; In the real time database, the alarm is generated.
Additional comments	

Result card	
Name/code/test case identifier	LVQuality
Results	Test successful.

Test card	
Name/code/test case identifier	LVPrevention
Objectives	Test the LV Fault Management (Fault Prevention) module.
Devices involved	LVGMU
Pre-requirements	Configure one microgeneration Smart Meter.
Steps	Open the device debug console to monitor the communication trace; Change the measured values in the Smart Meter to induce a high voltage alarm in the LV Fault Management module;
Expected result	The LV Fault Management module detects and identifies the high voltage; The communication trace shows that a power setpoint is sent from the LV Fault Management module to the Smart Meter.
Additional comments	

Result card	
Name/code/test case identifier	LVPrevention
Results	Test successful.

Test card	
Name/code/test case identifier	LVFraud
Objectives	Test the LV Fraud module outputs.
Devices involved	LVGMU
Pre-requirements	Configure 3 Smart Meters;
Steps	Open the device debug console to monitor the communication trace; Change the energy consumption for one of the Smart Meters.
Expected result	The GUI must indicate that the fraud alarm is generated.
Additional comments	

Result card	
Name/code/test case identifier	LVFraud
Results	Test successful.

5 Conclusions

In this deliverable we described the results of the work done when we integrated the modules of the energy management platform. This especially means that the defined and executed tests are provided in form of test and result cards. In addition the software modules for energy balancing and security and privacy have been integrated with the communication platform middleware developed in WP4, an integration step that was originally planned to be done as part of WP6. The energy balancing modules will be used in the Bronsbergen demonstrator within Work Package 6. The energy resilience modules have already been integrated in the commercial products of the consortium partner EFACEC. This will simplify their integration in the real world demonstrator at Batalha, i.e. also this will reduce effort in WP6.

References

- [1] J. J. Peralta, et al, "Deliverable D5.2 - Detailed specification, implementation and evaluation of energy balancing algorithms," Public deliverable of e-balance project, FP7-Smartcities-2013, Project number 609132, 2015.
- [2] A. Bernardo, et al, "Deliverable D5.3 - Energy resilience and self-healing," Public deliverable of e-balance project, FP7-Smartcities-2013, Project number 609132, 2016.
- [3] K. Piotrowski, et al, "Deliverable D5.4 - Detailed specification, implementation and evaluation of security and privacy means," Public deliverable of e-balance project, FP7-Smartcities-2013, Project number 609132, 2015.
- [4] D. Garrido, et al, "Deliverable D4.3 - Detailed middleware specification and implementation," Public deliverable of e-balance project, FP7-Smartcities-2013, Project number 609132, 2015.
- [5] H. Toersche, "Effective and Efficient Coordination of Flexibility in Smart Grids", unpublished PhD thesis, 2016.
- [6] K. Piotrowski, et al, "Deliverable D3.2 - Detailed System Architecture Specification," Public deliverable of e-balance project, FP7-Smartcities-2013, Project number 609132, 2015.