



e-balance

Deliverable D4.4

Implementation of an integrated communication platform

Editor:	Daniel Garrido (UMA)
Dissemination level: (Confidentiality)	PU
Suggested readers:	Consortium/Experts
Version:	1.0
Total number of pages:	44
Keywords:	Smart grids, communication platform, integration , evaluation

Abstract

This document describes the integration and evaluation process of the communication platform developed in WP4. The integration describes the approach and steps taken into account to connect the different modules of the communication platform, including a legacy device called GSmart. The integrated communication platform is evaluated according to an expected behaviour defined in a set of test cards. The sets of test cards are divided into two different categories. They are the functional testing (unit testing and integration testing) and the non-functional testing (performance and security testing). The result of each test with respect to the expected behaviour is described in a result card.

The tests have been implemented and evaluated using automated tools. The result obtained in the evaluation process is described in this deliverable and shows that the communication platform meets the expected behaviour.

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, nor that use of the information is free from risk, and accepts 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] WP4 Communication platform

[Document title] Implementation of an integrated communication platform

[Editor: Name, company] Daniel Garrido, UMA

[Work-package leader: Name, company] Daniel Garrido, UMA

Copyright notice

© 2015 Participants in project e-balance

Executive Summary

This deliverable describes the integration process of the communication platform and the different modules and algorithms developed in WP4. It also presents the evaluation carried out to test the correctness of the implemented prototype.

The integration and evaluation approach is presented in Section 2. In this section, the different modules and functionality integrated in the e-balance communication platform are presented:

- The integration/evaluation of the security modules described in D4.2
- The integration/evaluation of two programming languages wrappers that allow the e-balance communication platform to be used from Java and Javascript
- The integration/evaluation of the external device GSmart

At the end of this section the evaluation methodology is presented. Both functional and non-functional testing has been carried out by means of a set of tests. Each test is represented by a test card which defines the expected behaviour of the system from a specific set of input.

All the tests are defined and carried out in Section 3. The evaluation has been divided into 4 main sets of tests. In the first set described in Section 3.1, individual tests have been carried out taking into account a single instance of the e-balance communication platform. In Section 3.2 stress tests are carried out in the communication platform to evaluate the performance of the prototype in scenarios with a high number of requests. Security tests are described in Section 3.3. Finally communication between different management units is carried out in Section 3.4.

List of authors

Company	Author
UMA	Jaime Chen Eduardo Cañete Daniel Garrido
IHP	Krzysztof Piotrowski Ievgen Kabin
INOV	António Grilo

Table of Contents

Executive Summary.....	3
List of authors.....	4
Table of Contents	5
List of Tables.....	6
List of Figures.....	7
Abbreviations	8
1 Introduction	9
1.1 Deliverable Position in the Project.....	9
2 Integration methodology	11
2.1 Tools and equipment setup	13
2.2 Integration details.....	14
2.2.1 Security module integration.....	14
2.2.2 Data interface wrappers	14
2.2.3 GSmart.....	16
2.3 Test definition	16
3 Test definition and evaluation	19
3.1 Single tests	19
3.1.1 Create Variable	19
3.1.2 Delete a variable	20
3.1.3 Basic Write	21
3.1.4 Basic Query	22
3.1.5 Subscribe to a normal Event	22
3.1.6 Unsubscribe from a normal Event	23
3.1.7 Subscribe to Periodic event.....	24
3.1.8 Unsubscribe from a Periodic event.....	25
3.1.9 Subscribe to NewVariableValueEvent	26
3.1.10 Unsubscribe from NewVariableValueEvent.....	27
3.1.11 Basic Query Latest Tuple	27
3.1.12 Variable GSMART read operation.....	28
3.2 Stress tests.....	29
3.2.1 Simple write stress test	29
3.2.2 Simple query stress test.....	30
3.2.3 Concurrent write/query test	31
3.2.4 Concurrent write test.....	32
3.3 Security tests	34
3.3.1 Access control test 1	34
3.3.2 Access control test 2	35
3.3.3 Management unit registration in the group management – obtaining the certificate from the Certification Authority	37
3.3.4 Management unit registration in the group management – registration approach without proper credentials	39
3.4 Test with multiple CMUs.....	40
3.4.1 Broadcast Query	40
3.4.2 Remote procedure emulation	40
4 Summary and conclusions.....	43
References	44

List of Tables

Table 1: Test sets	13
Table 2: Test card template	17
Table 3: Result card template	17

List of Figures

Figure 1: The position of deliverable D4.4 within the e-balance project	9
Figure 2: The e-balance network architecture	11
Figure 3: The setup of the test environment	13
Figure 4: Communication platform data access control	14
Figure 5: Java wrapper implementation details	15
Figure 6: Javascript wrapper implementation details	15
Figure 7: Integration architecture of the GSmart.....	16

Abbreviations

CMU	Customer Management Unit
DMU	Device Management Unit
HAN	Home Area Network
LV	Low Voltage
LVGMU	Low Voltage Grid Management Unit
MU	Management Unit
MV	Medium Voltage
MVGMU	Medium Voltage Grid Management Unit
PS	Primary Substation
SS	Secondary Substation
TLS	Top Level System

1 Introduction

The goal of e-balance is to provide the infrastructure and management system necessary to support the future smart grid. This new system is expected to replace the traditional electricity grid and will provide new services that will allow a better control of energy consumption and production and as a result a better usage of the network.

The communication platform that connects the different devices in the smart grid is designed and developed in WP4. Task T4.1 surveys and proposes the networking layer that supports the e-balance energy balancing system. In task T4.2 different security and privacy mechanisms have been studied in order to ensure that the communication platform is secure. The data exchange middleware, the software in charge of handling information exchange, was designed and implemented in task T4.3.

This document describes the integration process and the evaluation of the protocols and mechanisms studied in previous tasks of this work package. This evaluation is composed of two different types of tests. On the one hand, the different levels of the communication architecture (e.g. HV grid, MV grid, etc.) are evaluated separately. This evaluation can be carried out in parallel to evaluate the correct implementation of each communication component and to detect possible issues. On the other hand, the system is integrated in-lab to a global communication platform and is evaluated with scenarios where management units at different levels in the architecture hierarchy are involved (e. g. CMU, LVGMU, etc.).

In Section 2 the integration process is described. Each level of communication (architecture) is presented and the different tests that have been carried out in each of them are described. The system has been evaluated by means of expected results according to specific inputs. This expected behaviour has been modelled by means of test cards. Section 3 describes all the test cards that have been used to evaluate the system. Section 4 describes the actual tests and results obtained based on the test cards defined. Finally, in Section 5, conclusions are presented.

1.1 Deliverable Position in the Project

Figure 1 shows the position of this deliverable within the e-balance project. This deliverable is part of work package WP4 – Communication Platform.

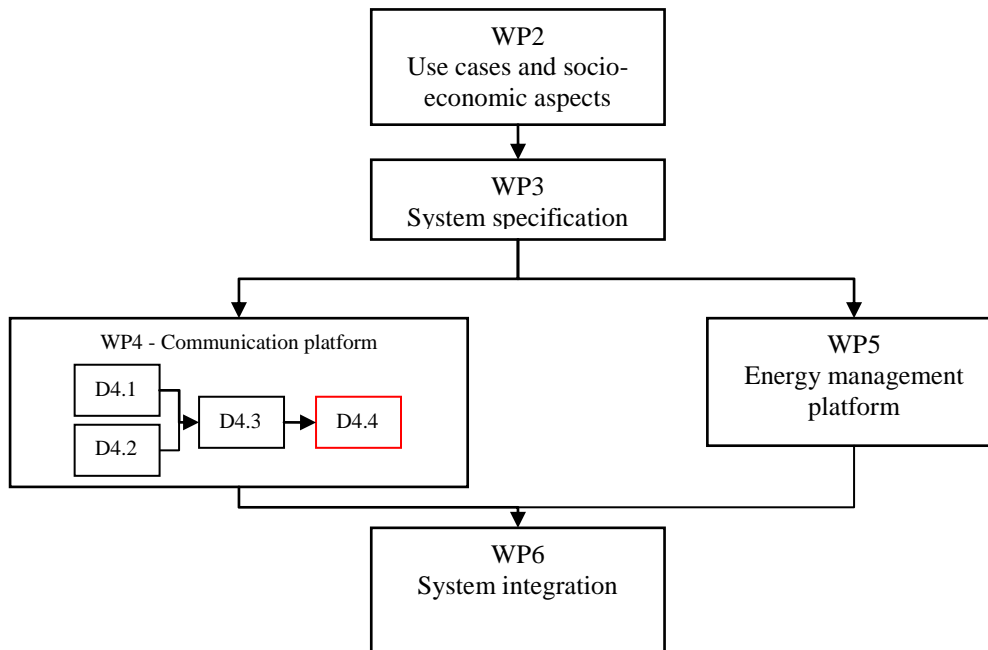


Figure 1: The position of deliverable D4.4 within the e-balance project

Deliverables D4.1 [1] and D4.2 [2] have studied the networking layer and the security and privacy mechanisms respectively. From the output of these two deliverables a data exchange middleware has been designed and implemented which was described in deliverable D4.3 [3]. This document describes the evaluation of the communication platform developed in task T4.3 and described in D4.3. This evaluation is carried out by means of in-lab tests taking into account the requirements defined in deliverable D2.4 [4]. Once the communication platform has been verified and tested, a more thorough in-site evaluation will be carried out in WP6 together with the energy management platform developed in WP5.

This document is a report for the integration task. Thus, state-of-the-art and beyond state-of-the-art status are not considered here. Instead, the focus is on reporting the different tests carried out to assure a correct integration of the communication platform components. Nevertheless, state-of-the-art status and the contribution beyond state-of-the-art can be queried in other WP4 deliverables.

2 Integration methodology

This section summarizes the high level plan for the integration of the communication platform.

Figure 2, extracted from the deliverable D3.1 [5] shows the generic communication architecture of e-balance, further described in deliverable D3.2 [6]. The e-balance architecture is composed of different hierarchy levels. These levels communicate with each other by means of devices called management units (MUs) represented in blue with the initials MU in Figure 2. The different MUs will communicate with each other using the communication platform developed in WP4. This communication platform has been integrated with the different protocols developed in this work package and is evaluated in this document. The communication platform used is identical at each level. The applications running on top of the communication platform define the behaviour of the MUs. Because of that, in this document the middleware has been evaluated in a generic way without taking into account specific behaviour of each MU.

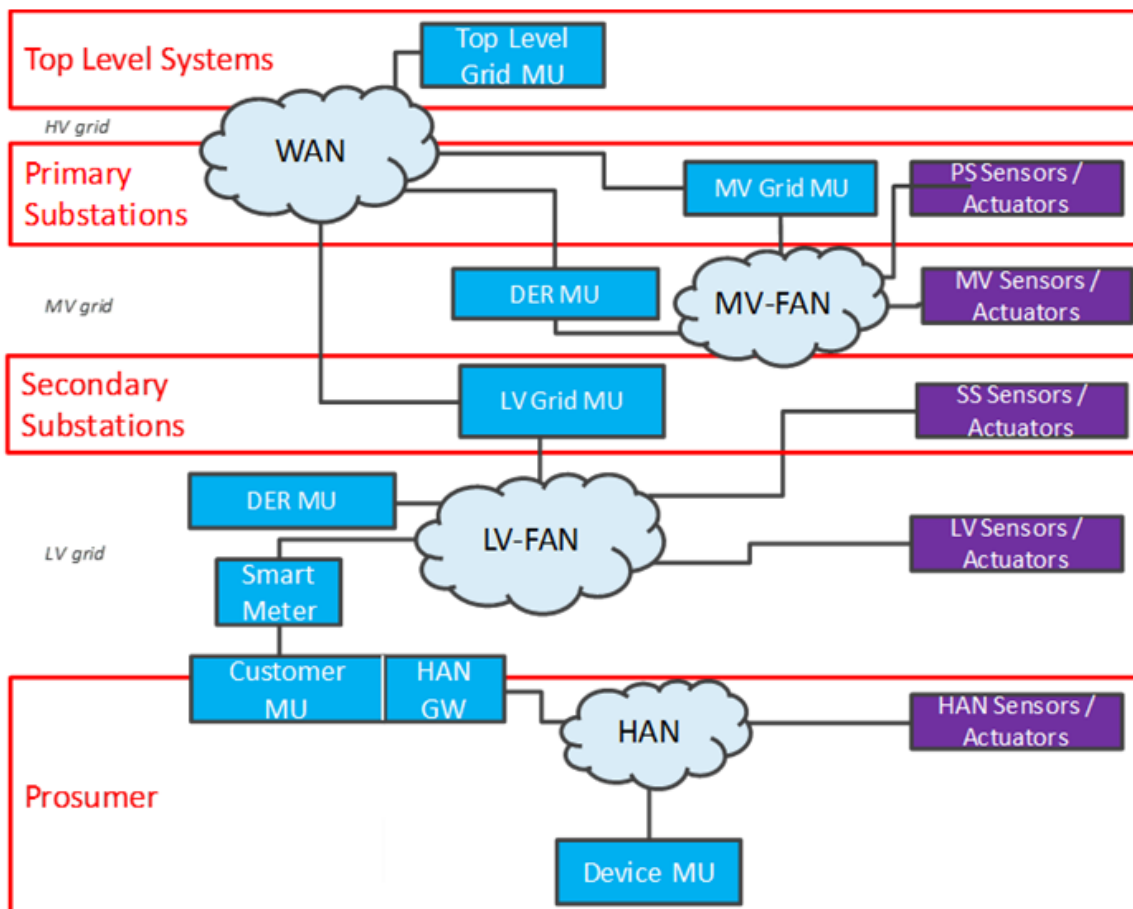


Figure 2: The e-balance network architecture

The software validation process has to confirm that it performs the user needs and intended uses, and that its requirements can be consistently fulfilled. A validation process has static and dynamic elements. Static features include the analysis of the documentation, implementation issues, manuals, etc. The dynamic features include the experimentation of functional and non-functional properties.

Integration is an essential part in the process of software validation. In this deliverable we focus on the integration and evaluation process of the e-balance communication platform.

At D2.4, the e-balance requirements were defined attending to the use cases. The design and implementation of the communication platform, takes these requirements and use cases into account in order to carry out the integration evaluation.

Specifically, we distinguish the following integration validation areas as depicted in Table 1:

- Functional requirements: this area will validate functional properties of the communication platform. These properties have been selected taking into account how they can help in the development of applications using for the smart grid.
- Non-Functional requirements: this area will study non-functional aspects of the middleware architecture and how these aspects affect important features such as security, performance, etc.

The validation criteria of the integration communication platform include the following aspects:

- How the e-balance API supports most of the interactions needed by the e-balance applications.
- Feasibility to express the interaction among applications (identification, broadcasting, group communication...)
- How the communication platform can be used in different levels of the e-balance architecture (LV, MV, CMU, ...)
- Communication platform flexibility to allow reusability of the middleware in different devices, platforms, operating systems, etc.
- Scalability support considered by-design
- Interoperability with external devices (e.g. GSmart)
- Possibility of using different programming languages
- Middleware efficiency

Taking these aspects into account, the software integration of the different components of the system has been carried out in-lab using the tools and setup explained in Section 2.1. The following modules have been tested and integrated in the e-balance middleware:

1. e-balance middleware module
2. Security module
3. e-balance wrappers
 - a. Javascript wrapper
 - b. Java wrapper
4. GSmart

The methodology used to test the different components of the system divides the set of tests in two different categories.

- Functional testing: unit testing and integration testing.
- Non-functional testing: Includes performance and security testing.

Functional testing includes all the different tests carried out to verify that the system meets the expected behaviour in terms of features offered. These tests include unit testing which tests individual operations and integration testing which verifies the interaction between different components of the system.

On the other hand, non-functional testing focuses on a different set of requirements met by the system but not directly related to its behaviour but rather to its operation. These sets of tests include all the tests related to performance and security. The majority of the tests have been carried out using automated tools. For C#, Visual studio test explorer included in Visual Studio has been used. JUnit framework has been used for Java. The middleware GUI has been used as a help tool in order to verify the results of some of the tests.

The different tests have been described using a set of test cards and the results of the tests have been described in result cards. The whole set of test cards have been divided into 4 main sections that is described in Table 1.

Table 1: Test sets

Type of test	Test set	Section	Description
Functional test	Unit testing	3.1	Individual data interface operations
Functional test	Integration testing	3.4	Interaction between different management units
Non-functional test	Performance testing	3.2	Stress tests
Non-functional test	Security testing	3.3	Security tests: data access control, middleware access control

2.1 Tools and equipment setup

The communication platform has been implemented using C# and the ServiceStack framework [7] that runs over Mono [8] and has been deployed in Beaglebone Black devices [9]. The Beaglebone Black devices have been connected to the local area network by means of Ethernet cables.

The communication platform provides a set of REST web services that can be accessed from any device in the same local area network. In order to simplify the development of applications in different programming languages with the e-balance data interface, two different wrappers have been developed: a Javascript wrapper and a Java wrapper. These two wrappers allow programs in Javascript and Java to use the e-balance middleware respectively. In addition, each Beaglebone hosts a website that simplifies the use of some of these web services and can be used to query different parameters.

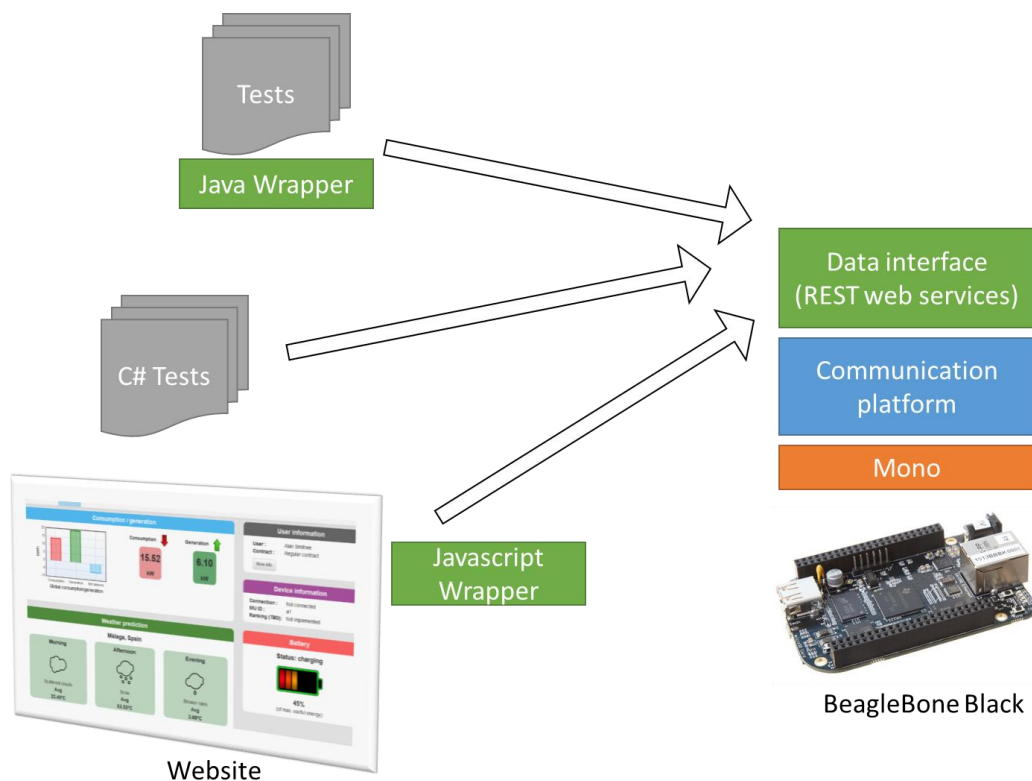


Figure 3: The setup of the test environment

The tests consist of different interactions between the test program and the web services of the BeagleBone as presented in Figure 3.

2.2 Integration details

2.2.1 Security module integration

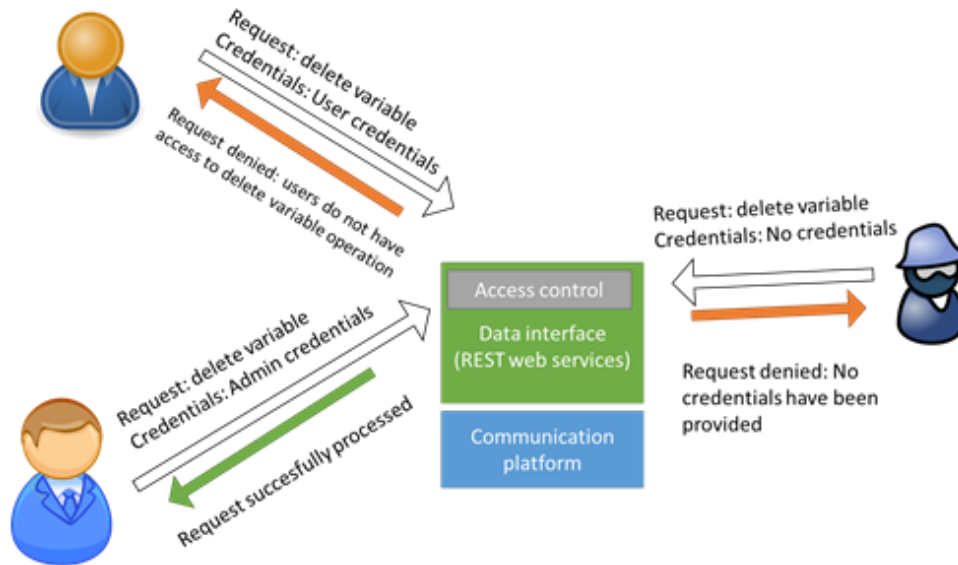


Figure 4: Communication platform data access control

The first set of security measures provided by the communication platform is the access control (Figure 4). Only the users of the communication platform with the appropriate credentials are able to execute the different web services provided by the platform. Prior to using the communication platform, the user must authenticate in the system. With each subsequent request, information about the user is sent so that the system can check if the user has the corresponding credentials to access the information. This functionality has been integrated in the communication platform by means of request filters. Before processing requests, filter requests are executed. In the access control request filter, permission is granted or denied based on the provided credentials.

Other security and privacy related functions that have been integrated include also the security and privacy module that is integrated as part of the Java wrapper offering its functions to the services in the energy management platform. Further, in order to allow the secure group management module to work properly the certification authority software was provided and running on a (remote) machine, allowing registration of stakeholders and management units. For those two groups proper certificates are generated and are further stored on the respective management unit and are used by the middleware to authenticate (prove the identity of) the management unit (RSA hardware certificate of the management unit) or by the security and privacy module in the energy management platform to authenticate the requests by the services of that specific stakeholder (ECC stakeholder/service certificates). This registration process is the basis for all other security related actions and the secure data exchange.

2.2.2 Data interface wrappers

Integration of applications developed in other programming languages or websites that run in conventional web browsers is achieved by means of wrappers. Wrappers offer access to the data interface API of the e-balance communication platform programmatically instead of having to use REST web services.

2.2.2.1 Java wrapper

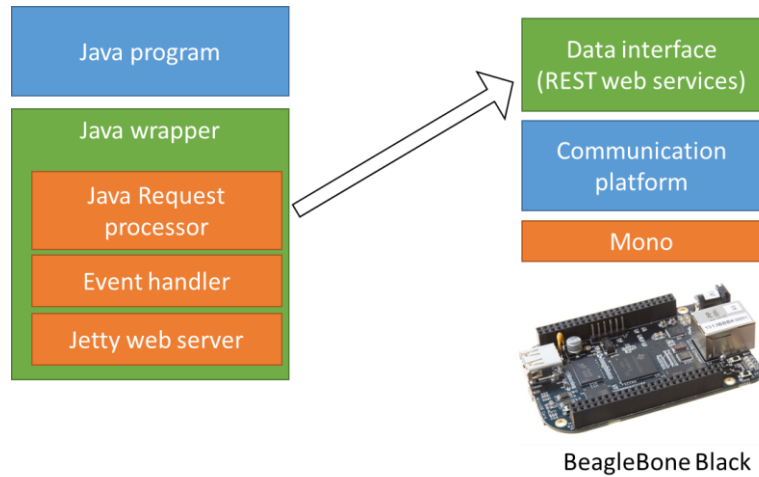


Figure 5: Java wrapper implementation details

Java programs can access the middleware by means of the Java wrapper. This wrapper hides the underlying REST web service architecture and allows Java code to programmatically use the e-balance data interface API. Figure 5 shows some implementation details of the Java wrapper. All requests are processed by the Java request processor.

On the one hand on demand operations such as query and write are directly translated into calls to web services hosted in the remote communication platform by the Java request processor. On the other hand, events sent from the communication platform that are received in Java need special processing. The e-balance communication platform is entirely based on a REST web service data interface and therefore events are only sent to other web services. In order for a Java code to receive an event, a translation is needed from a webservice to Java callback. The java wrapper internally hosts a lightweight web server implemented using Jetty [10] that receives events and translates them to the appropriate Java callback. This allows users to transparently use Java code to access all the functionality in the middleware.

2.2.2.2 Javascript wrapper

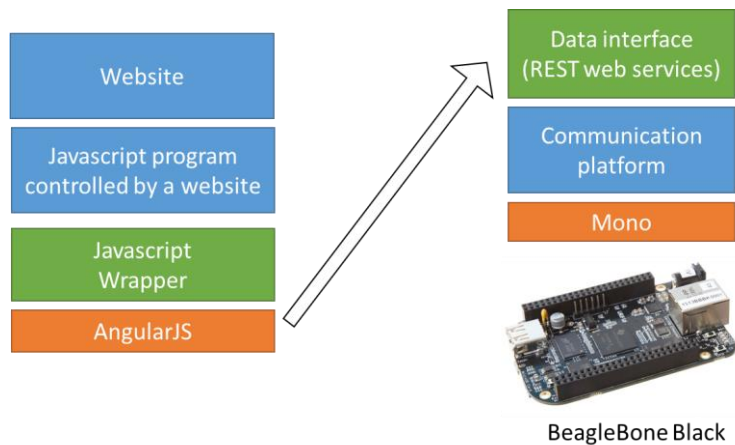


Figure 6: Javascript wrapper implementation details

As mentioned in Section 2.1 each Beaglebone device hosts a website that shows information about the middleware state and also allows different parameters to be modified. In order to easily allow websites and mobile devices to access the e-balance middleware a Javascript wrapper has been developed and integrated in the e-balance system. Figure 6 shows some details about the Javascript wrapper implementation. The Javascript wrapper has been implemented over the AngularJS [11] application framework. Each service available in the Javascript wrapper is implemented as an asynchronous AJAX call to the middleware using the \$resource object provided by AngularJS.

The website hosted in each BeagleBone uses this Javascript wrapper to monitor and control the local middleware. Use of events is not supported at this moment in the Javascript wrapper.

2.2.3 GSmart

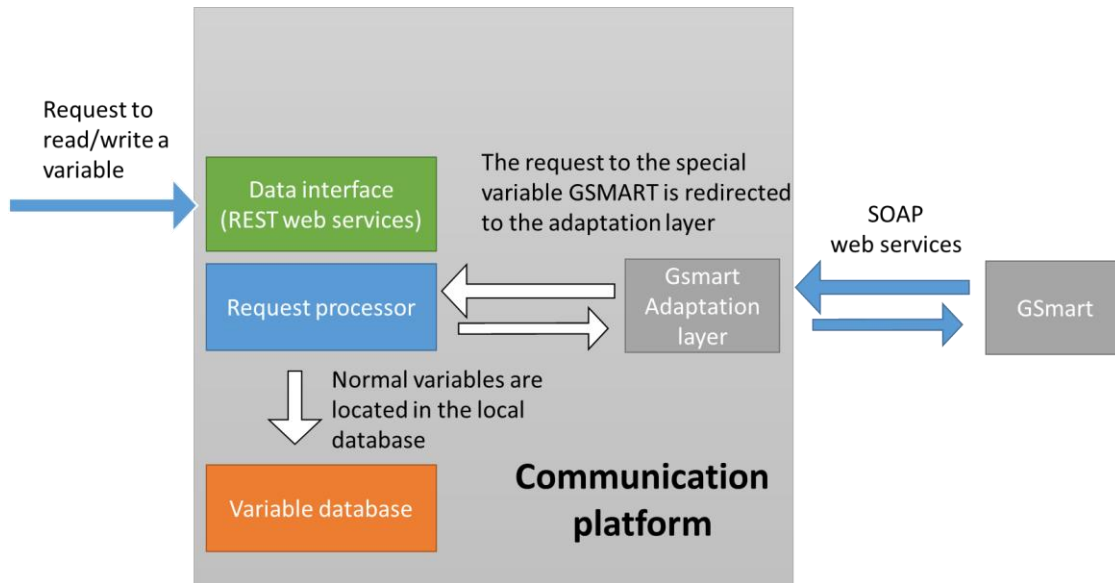


Figure 7: Integration architecture of the GSmart

The e-balance communication platform provides a single interface (data interface API) that all devices in the system have to use to exchange information and obtain data. However, it is common in existing infrastructures to deal with legacy code and equipment. These devices are not able to use or provide data by means of the data interface. For these kinds of devices an adaptation module is used to adapt the provided interface to the e-balance data interface. In the Bronsbergen demonstrator described in [12] the use of an existing piece of equipment called GSmart is expected to be used. A GSmart is a server device that controls sensors in the grid and provides its data by means of a set of SOAP web services. SOAP and REST web services cannot directly communicate and there the use of an adaptation layer is needed. Figure 7 shows the adaptation module implemented to allow communication between the GSmart and the communication platform. The GSmart is accessed through the e-balance data interface API. A special variable called GSMART has been created. The adaptation module receives requests that operate with this special variable and instead of locally processing the request using communication platform local database the request is translated to a SOAP request and redirected to the GSmart. When the response is received the SOAP response is processed and its content is written in the special variable GSMART.

As a result the use of the GSmart external device is transparent to the developers which can obtain its data by means of the common data interface API provided by the communication platform.

2.3 Test definition

The requirements of the communication platform have been tested by comparing obtained results to expected results when specific inputs are fed into the system. This behaviour is controlled by a set of tests that are specified in what is called a “test card”. A test card describes a test for a specific part of the system and presents the details of it by describing the pre-conditions of the test, the input that the system is given and the expected results. If the expected results and obtained results are the same, then the test is passed successfully. Results of the tests are collected in what is called a “result card”. A result card describes the results and the outcome of a particular test card. The evaluation process consists of defining a set of test cards and filling the corresponding result cards based on the results obtained from the tests.

Table 2: Test card template

Test card	
Name/code/test case identifier	<p>TEST.<TEST_TYPE><TESTNUMBER> E.g.: TEST.S001</p> <p>TEST_TYPE: indicates the type of test. Possible values are: I: Individual tests (only one MU is involved) M: Tests where multiple MUs are involved S: Stress test X: Security test</p> <p>TESTNUMBER: unique id for each test (use at least 3 digits)</p>
Objectives	Goal of this test
Devices involved	Devices involved in the use case. Figures or diagram are preferred
Pre-requirements	Assumptions or pre-requirements needed to evaluate this test
Steps	Steps to be carried out to evaluate this test
Expected result	
Additional comments	

Table 3: Result card template

Result card	
Name/code/test case identifier	<p>TEST.<TEST_TYPE><TESTNUMBER> E.g.: TEST.S001</p> <p>TEST_TYPE: indicates the type of test. Possible values are: I: Individual tests (only one MU is involved) M: Tests where multiple MUs are involved S: Stress test X: Security test</p> <p>TESTNUMBER: unique id for each test (use at least 3 digits)</p>
Results	Detailed description of the outcome of this test, highlighting interesting results or issues detected
Test outcome	<p>OK / FAIL</p> <p>In case of FAIL create a new version of the test card (use the “Test repetition number” field accordingly) and repeat the test once the issue has been fixed. Indicate in this test card the test case identifier of the test card where this test is repeated)</p>
Additional comments	

Table 2 shows the test card template that defines the expected behaviour of each test. The obtained results corresponding to each test card are described in a result card using the template shown in Table 3. In each of these tables the fields are described in detail.

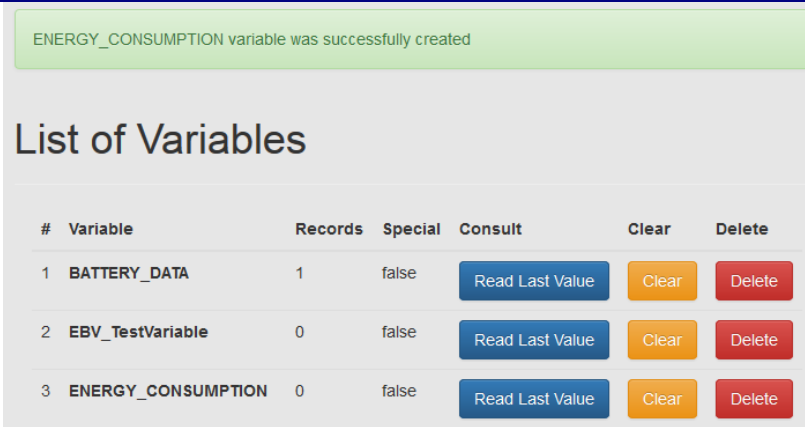
3 Test definition and evaluation

This section defines the tests and describes the results of the evaluation. The integration evaluation has been divided into 4 main sets of tests. In the first set described in Section 3.1, individual tests have been carried out taking into account a single instance of the e-balance communication platform. These tests are used to check the correctness of the different management units and/or sensors controlled by it without taking into account interaction between different management units. This information is useful to detect possible issues/bugs in the system in specific components and to study the performance of each MU in an isolated manner before carrying out tests with multiple MUs. In Section 3.2 stress tests are carried out in the communication platform to evaluate the performance of the prototype in scenarios with a high number of requests. Security tests are described in Section 3.3. Finally communication between different management units is carried out in Section 3.4. These tests study the communication and interaction between management units/devices in different positions of the architecture hierarchy.

3.1 Single tests

3.1.1 Create Variable

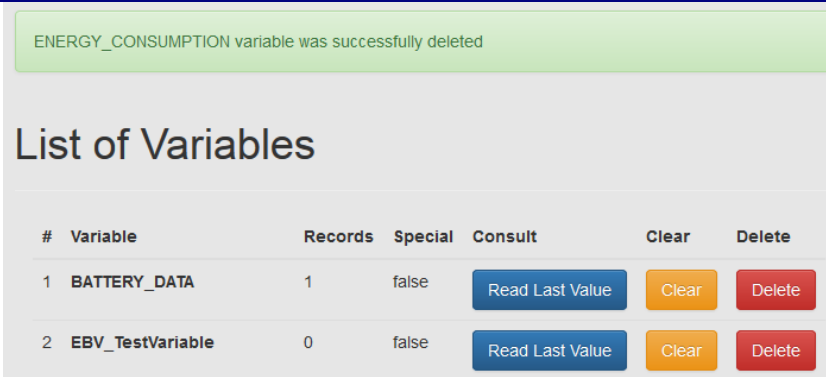
Test card	
Name/code/test case identifier	TEST.I001
Objectives	Check if the middleware is able to create a new variable called "ENERGY_CONSUMPTION" with the following properties: ID, TIMESTAMP and VALUE.
Devices involved	Customer Management Unit (CMU)
Pre-requirements	The user executing the test has access to the middleware due to the appropriate credentials A variable ENERGY_CONSUMPTION does not exist.
Steps	1. Call the web service offered by the middleware to create a variable called "ENERGY_CONSUMPTION"
Expected result	A message confirming that the variable was created.
Additional comments	

Result card																													
Name/code/test case identifier	TEST.I001																												
Results	 <p>ENERGY_CONSUMPTION variable was successfully created</p> <p>List of Variables</p> <table border="1"> <thead> <tr> <th>#</th> <th>Variable</th> <th>Records</th> <th>Special</th> <th>Consult</th> <th>Clear</th> <th>Delete</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>BATTERY_DATA</td> <td>1</td> <td>false</td> <td>Read Last Value</td> <td>Clear</td> <td>Delete</td> </tr> <tr> <td>2</td> <td>EBV_TestVariable</td> <td>0</td> <td>false</td> <td>Read Last Value</td> <td>Clear</td> <td>Delete</td> </tr> <tr> <td>3</td> <td>ENERGY_CONSUMPTION</td> <td>0</td> <td>false</td> <td>Read Last Value</td> <td>Clear</td> <td>Delete</td> </tr> </tbody> </table>	#	Variable	Records	Special	Consult	Clear	Delete	1	BATTERY_DATA	1	false	Read Last Value	Clear	Delete	2	EBV_TestVariable	0	false	Read Last Value	Clear	Delete	3	ENERGY_CONSUMPTION	0	false	Read Last Value	Clear	Delete
#	Variable	Records	Special	Consult	Clear	Delete																							
1	BATTERY_DATA	1	false	Read Last Value	Clear	Delete																							
2	EBV_TestVariable	0	false	Read Last Value	Clear	Delete																							
3	ENERGY_CONSUMPTION	0	false	Read Last Value	Clear	Delete																							

	This screenshot shows the output received in the GUI when a request to create a variable is sent to the middleware.
Test outcome	OK
Additional comments	

3.1.2 Delete a variable

Test card	
Name/code/test case identifier	TEST.I002
Objectives	Check if the middleware is able to delete a variable previously created. This test will try to delete a variable called "ENERGY_CONSUMPTION".
Devices involved	Customer Management Unit (CMU)
Pre-requirements	The user executing the test has access to the middleware due to the appropriate credentials A variable called ENERGY_CONSUMPTION exists.
Steps	1. Call the web service offered by the middleware to delete a variable called "ENERGY_CONSUMPTION"
Expected result	A message confirming that the variable was deleted.
Additional comments	

Result card																						
Name/code/test case identifier	TEST.I002																					
Results	 <p>The screenshot shows a green notification bar at the top stating "ENERGY_CONSUMPTION variable was successfully deleted". Below it is a section titled "List of Variables" containing a table with the following data:</p> <table border="1"> <thead> <tr> <th>#</th> <th>Variable</th> <th>Records</th> <th>Special</th> <th>Consult</th> <th>Clear</th> <th>Delete</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>BATTERY_DATA</td> <td>1</td> <td>false</td> <td>Read Last Value</td> <td>Clear</td> <td>Delete</td> </tr> <tr> <td>2</td> <td>EBV_TestVariable</td> <td>0</td> <td>false</td> <td>Read Last Value</td> <td>Clear</td> <td>Delete</td> </tr> </tbody> </table>	#	Variable	Records	Special	Consult	Clear	Delete	1	BATTERY_DATA	1	false	Read Last Value	Clear	Delete	2	EBV_TestVariable	0	false	Read Last Value	Clear	Delete
#	Variable	Records	Special	Consult	Clear	Delete																
1	BATTERY_DATA	1	false	Read Last Value	Clear	Delete																
2	EBV_TestVariable	0	false	Read Last Value	Clear	Delete																
Test outcome	OK																					
Additional comments																						

3.1.3 Basic Write

Test card	
Name/code/test case identifier	TEST.I003
Objectives	Check if the middleware allows to write data in a variable previously created.
Devices involved	Customer Management Unit (CMU)
Pre-requirements	The user executing the test has access to the middleware due to the appropriate credentials The middleware contains a variable called ENERGY_CONSUMPTION which has the following properties: ID, TIMESAMTP and VALUE.
Steps	1. Call the write web service to add the following tuple: [X, CURRENT_TIME, 29]
Expected result	The variable ENERGY_CONSUMPTION must contain the following new tuple: [X, CURRENT_TIME, 29]
Additional comments	

Result card																												
Name/code/test case identifier	TEST.I003																											
Results	<p>Screenshot obtained from the GUI, before the test is executed</p> <table border="1"> <thead> <tr> <th>ID</th> <th>Timestamp</th> <th>VALUE</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1446566867</td> <td>75</td> </tr> <tr> <td>2</td> <td>1446566867</td> <td>31</td> </tr> <tr> <td>3</td> <td>1446566867</td> <td>92</td> </tr> </tbody> </table> <p>Screenshot obtained from the GUI of the middleware:</p> <table border="1"> <thead> <tr> <th>ID</th> <th>Timestamp</th> <th>VALUE</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1446566867</td> <td>75</td> </tr> <tr> <td>2</td> <td>1446566867</td> <td>31</td> </tr> <tr> <td>3</td> <td>1446566867</td> <td>92</td> </tr> <tr> <td>4</td> <td>1446567063</td> <td>29</td> </tr> </tbody> </table>	ID	Timestamp	VALUE	1	1446566867	75	2	1446566867	31	3	1446566867	92	ID	Timestamp	VALUE	1	1446566867	75	2	1446566867	31	3	1446566867	92	4	1446567063	29
ID	Timestamp	VALUE																										
1	1446566867	75																										
2	1446566867	31																										
3	1446566867	92																										
ID	Timestamp	VALUE																										
1	1446566867	75																										
2	1446566867	31																										
3	1446566867	92																										
4	1446567063	29																										
Test outcome	OK																											
Additional comments																												

3.1.4 Basic Query

Test card	
Name/code/test case identifier	TEST.I004
Objectives	Check reading an e-balance variable stored in the middleware
Devices involved	Customer Management Unit (CMU)
Pre-requirements	<p>The user executing the test has access to the middleware due to the appropriate credentials</p> <p>The middleware contains a variable called ENERGY_CONSUMPTION which has the following properties: ID, TIMESTAMP and VALUE. Furthermore, the variable contains the following three tuples:</p> <p>[Id, Timestamp, Value]</p> <p>1, 1446563855, 34</p> <p>2, 1446563860, 36</p> <p>3, 1446563864, 38</p>
Steps	1. Call the query web service to read information from the ENERGY_CONSUMPTION variable.
Expected result	<p>An object with the following tuples</p> <p>[1, 1446563855, 34]</p> <p>[2, 1446563855, 36]</p> <p>[3, 1446563855, 38]</p>
Additional comments	

Result card	
Name/code/test case identifier	TEST.I004
Results	<p>A data object with the following tuples:</p> <p>[1, 1446563855, 34]</p> <p>[2, 1446563855, 36]</p> <p>[3, 1446563855, 38]</p>
Test outcome	OK
Additional comments	

3.1.5 Subscribe to a normal Event

Test card	
Name/code/test case identifier	TEST.I005
Objectives	Check if an e-balance application can subscribe to an event related to a variable stored in the middleware.
Devices involved	PC, Customer Management Unit (CMU)
Pre-requirements	<p>The user executing the test has access to the middleware due to the appropriate credentials</p> <p>The middleware contains a variable called ENERGY_CONSUMPTION</p>

	<p>which has the following properties: ID, TIMESTAMP and VALUE. Furthermore, the variable contains the following three tuples:</p> <p>[Id, Timestamp, Value]</p> <p>1, 1446566867, 75</p> <p>2, 1446566869, 31</p> <p>3, 1446566871, 92</p> <p>4, 1446566873, 29</p>
Steps	<ol style="list-style-type: none"> 1. Create the variable we want to subscribe to. E.g.: We want to consult all the properties of the ENERGY_CONSUMPTION variable which “value” property is higher than 40. 2. Define an event for the previously created variable where the period with which we want to receive information of the variable is indicated. E.g.: Every 10 seconds. 3. Create a handler to manage the information received periodically. <pre>EBVariable eventVariable = new EBVariable("ENERGY_CONSUMPTION", "*", "value > 40"); EBEvent event = new EBEvent("my_demo_event", 10000, eventVariable); MyDemoEventHandler1 eventHandler = new MyDemoEventHandler1(); Event.EventResponse eventResponse = EbalanceAPI.SubscribeToEvent(remoteAddress, event, eventHandler);</pre> <p>Note: The subscriber will only receive events if the condition specified in the variable is satisfied.</p>
Expected result	<p>Every 10 seconds receive the following tuples:</p> <p>[1, 1446566867, 75]</p> <p>[3, 1446566871, 92]</p>
Additional comments	

Result card	
Name/code/test case identifier	TEST.I005
Results	<p>A data object with the following tuples:</p> <p>[1, 1446566867, 75]</p> <p>[3, 1446566871, 92]</p>
Test outcome	OK
Additional comments	The tuples were received every 10 seconds.

3.1.6 Unsubscribe from a normal Event

Test card	
Name/code/test case identifier	TEST.I006
Objectives	The user executing the test has access to the middleware due to the appropriate credentials

	Check if an e-balance application can unsubscribe from a normal event related to a variable stored in middleware.
Devices involved	PC, Customer Management Unit (CMU)
Pre-requirements	The middleware contains an event subscribed to a variable called ENERGY_CONSUMPTION. The event has the following properties: <ul style="list-style-type: none"> • Variable Name: ENERGY_CONSUMPTION • Checking Frequency: 10 seconds • Properties: * • Condition: value > 40
Steps	<pre>EBVariable eventVariable = new EBVariable("ENERGY_CONSUMPTION", "*", "value > 40"); EBEvent event = new EBEvent("my_demo_event", 10000, eventVariable); Event.EventResponse eventResponse = EbalanceAPI.UnsubscribeFromEvent(remoteAddress, event);</pre>
Expected result	Stop receiving events.
Additional comments	

Result card	
Name/code/test case identifier	TEST.I006
Results	e-balance application stops receiving events.
Test outcome	OK
Additional comments	

3.1.7 Subscribe to Periodic event

Test card	
Name/code/test case identifier	TEST.I007
Objectives	The user executing the test has access to the middleware due to the appropriate credentials Check if an e-balance application can subscribe to a periodic event related to a variable stored in the middleware.
Devices involved	PC, Customer Management Unit (CMU)
Pre-requirements	The middleware contains a variable called ENERGY_CONSUMPTION which has the following properties: ID, TIMESTAMP and VALUE. Furthermore, the variable contains the following three tuples: [Id, Timestamp, Value] 1, 1446566867, 75 2, 1446566869, 31 3, 1446566871, 92 4, 1446566873, 29
Steps	<ol style="list-style-type: none"> 1. Create the variable we want to subscribe to. E.g.: We want to consult all the properties of the ENERGY_CONSUMPTION variable which “value” property is lower than 40. 2. Define a periodic for the previously created variable where the

	<p>period with which we want to receive information of the variable is indicated. E.g.: Every 20 seconds.</p> <p>3. Create a handler to manage the information received periodically.</p> <pre>EBVariable periodicVariable = new EBVariable("ENERGY_CONSUMPTION", "*", "value < 40"); EBPeriodic periodic = new EBPeriodic("Per_TestVariable", 20000, periodicVariable); MyDemoPeriodicHandler1 periodicHandler = new MyDemoPeriodicHandler1(); Periodic.PeriodicResponse periodicResponse = EbalanceAPI.SubscribeToPeriodic(remoteAddress, periodic, periodicHandler);</pre> <p>Note: The subscriber will always receive periodics. Even, if the condition specified in the variable is not satisfied.</p>
Expected result	<p>Every 20 seconds receive the following tuples:</p> <p>[2, 1446566869, 31]</p> <p>[4, 1446566873, 29]</p>
Additional comments	

Result card	
Name/code/test case identifier	TEST.S007
Results	<p>A data object with the following tuples:</p> <p>[2, 1446566869, 31]</p> <p>[4, 1446566873, 29]</p>
Test outcome	OK
Additional comments	The tuples were received every 20 seconds.

3.1.8 Unsubscribe from a Periodic event

Test card	
Name/code/test case identifier	TEST.I008
Objectives	Check if an e-balance application can unsubscribe from a periodic related to a variable stored in the middleware.
Devices involved	PC, Customer Management Unit (CMU)
Pre-requirements	<p>The user executing the test has access to the middleware due to the appropriate credentials</p> <p>The middleware contains a periodic subscribed to a variable called ENERGY_CONSUMPTION. The event has the following properties:</p> <ul style="list-style-type: none"> • Variable Name: ENERGY_CONSUMPTION • Checking Frequency: 20 seconds • Properties: * • Condition: value < 40

Steps	<pre>EBVariable periodicVariable = new EBVariable("ENERGY_CONSUMPTION", "*", "value < 40"); EBPeriodic periodic = new EBPeriodic("Per_TestVariable", 20000, periodicVariable); Periodic.PeriodicResponse periodicResponse = EbalanceAPI.UnsubscribeFromPeriodic(remoteAddress, periodic);</pre>
Expected result	Stop receiving periodic updates.
Additional comments	

Result card	
Name/code/test case identifier	TEST.I008
Results	e-balance application stopped receiving periodic updates
Test outcome	OK
Additional comments	

3.1.9 Subscribe to NewVariableValueEvent

Test card	
Name/code/test case identifier	TEST.I009
Objectives	Check if an e-balance application can subscribe to a NewVariableValueEvent related to a variable stored in the middleware.
Devices involved	PC, Customer Management Unit (CMU)
Pre-requirements	The user executing the test has access to the middleware due to the appropriate credentials The middleware contains a variable called ENERGY_CONSUMPTION.
Steps	<p>1.Subscription to the NewVariableValueEvent:</p> <pre>MyDemoEventHandler1 eventHandler = new MyDemoEventHandler1(); Event.EventResponse eventResponse = EbalanceAPI.SubscribeToNewVariableValueEvent(remoteAddress, "ENERGY_CONSUMPTION", eventHandler);</pre> <p>2.Write data into the ENERGY_CONSUMPTION variable:</p> <pre>Write.WriteResponse writeResponse = EbalanceAPI.Write(remoteAddress, variableName, "value", "323");</pre>
Expected result	To receive – by means of an event – data stored in the ENERGY_CONSUMPTION variable, just in the moment an external actor stores a new value in this variable. The received data will have the following form: [Id, Timestamp, Value] [5, Y, 323] Y will be the timestamp used in the moment of storing the information.
Additional comments	

Result card	
Name/code/test case identifier	TEST.I009
Results	[Id, Timestamp, Value] [5, 1446652432, 323]
Test outcome	OK
Additional comments	

3.1.10 Unsubscribe from NewVariableValueEvent

Test card	
Name/code/test case identifier	TEST.I010
Objectives	Check if an e-balance application can unsubscribe from a NewVariableValueEvent.
Devices involved	PC, Customer Management Unit (CMU)
Pre-requirements	The user executing the test has access to the middleware due to the appropriate credentials The middleware contains a NewVariableValueEvent on a ENERGY_CONSUMPTION variable.
Steps	<pre>Event.EventResponse eventResponse = EbalanceAPI. UnsubscribeFromNewVariableValueEvent(remoteAddress, "ENERGY_CONSUMPTION");</pre>
Expected result	Stop receiving NewVariableValue events.
Additional comments	

Result card	
Name/code/test case identifier	TEST.I010
Results	e-balance application stops receiving NewVariableValue events although new values are written in the ENERGY_CONSUMPTION variable.
Test outcome	OK
Additional comments	

3.1.11 Basic Query Latest Tuple

Test card	
Name/code/test case identifier	TEST.I011
Objectives	Check that an e-balance variable stored in the middleware can be queried.
Devices involved	Customer Management Unit (CMU)

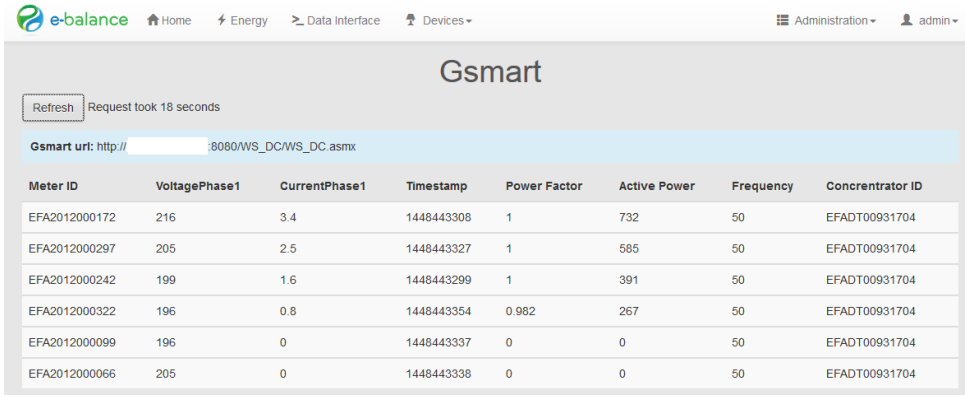
Pre-requirements	The user executing the test has access to the middleware due to the appropriate credentials
Steps	<ol style="list-style-type: none"> 1. Create an empty variable called “TestVariable” 2. Execute QueryLatestTuple and check that no result is returned 3. Write a value VALUE1 to the variable “TestVariable” 4. Execute QueryLatestTuple and check that VALUE1 is received 5. Write a value VALUE2 to the variable “TestVariable” 6. Execute QueryLatestTuple and check that VALUE2 is received 7. Delete variable “TestVariable”
Expected result	<p>All tests must be executed successfully</p> <p>Checks carried out in steps 2, 4 and 6 must be successful.</p>
Additional comments	

Result card	
Name/code/test case identifier	TEST.I011
Results	<div style="border: 1px solid #ccc; padding: 10px; background-color: #f9f9f9;"> <p>Test Name: TestBasicQueryLatestTuple</p> <p>Test Outcome: ✔ Passed</p> <hr/> <p>Standard Output</p> <p>Debug Trace: Variable created ok Query returned 0 results Value 1 received ok Value 2 received ok Variable deleted ok</p> </div> <p>This screenshot shows the result of an automated test</p>
Test outcome	OK
Additional comments	

3.1.12 Variable GSMART read operation

Test card	
Name/code/test case identifier	TEST.I012
Objectives	Check that the e-balance special variable GSMART can be queried. The information in this variable is actually information provided by an external device called GSmart.
Devices involved	<p>Low Voltage Management Unit (LVGMU)</p> <p>GSmart</p>
Pre-requirements	The user executing the test has access to the middleware due to the

	appropriate credentials
Steps	1. Query the special variable GSMART by means of the LVGMU web application. It contains a helper webpage that automatically makes the request.
Expected result	The operation must return a Success operation code. Meaningful information from the GSmart must be obtained.
Additional comments	The obtained latency in the request will be measured

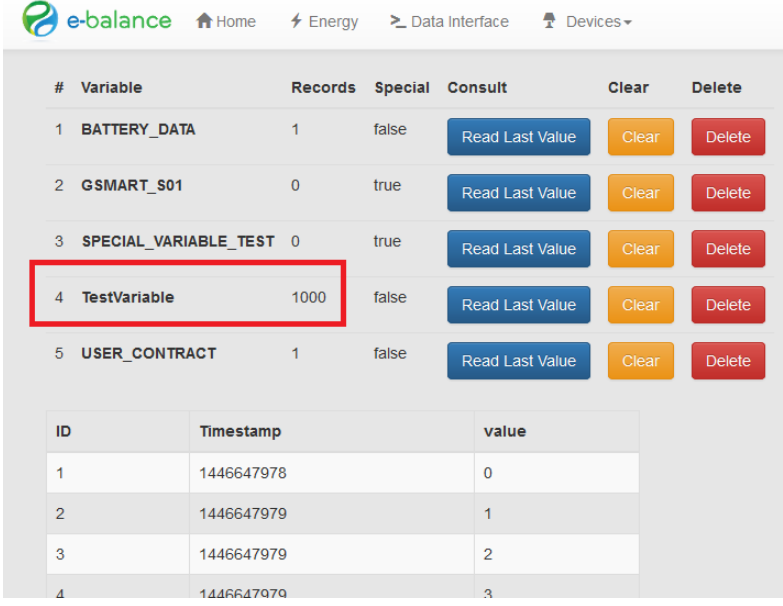
Result card																																																									
Name/code/test case identifier	TEST.I012																																																								
Results	 <p>The screenshot shows the 'Gsmart' web application interface. At the top, there is a navigation bar with 'e-balance' logo and links for Home, Energy, Data Interface, and Devices. A 'Refresh' button and 'Request took 18 seconds' message are visible. Below, the 'Gsmart url' is shown as 'http://:8080/WS_DC/WS_DC.asmx'. A table displays the following data:</p> <table border="1"> <thead> <tr> <th>Meter ID</th> <th>VoltagePhase1</th> <th>CurrentPhase1</th> <th>Timestamp</th> <th>Power Factor</th> <th>Active Power</th> <th>Frequency</th> <th>Concentrator ID</th> </tr> </thead> <tbody> <tr> <td>EFA2012000172</td> <td>216</td> <td>3.4</td> <td>1448443308</td> <td>1</td> <td>732</td> <td>50</td> <td>EFADT00931704</td> </tr> <tr> <td>EFA2012000297</td> <td>205</td> <td>2.5</td> <td>1448443327</td> <td>1</td> <td>585</td> <td>50</td> <td>EFADT00931704</td> </tr> <tr> <td>EFA2012000242</td> <td>199</td> <td>1.6</td> <td>1448443299</td> <td>1</td> <td>391</td> <td>50</td> <td>EFADT00931704</td> </tr> <tr> <td>EFA2012000322</td> <td>196</td> <td>0.8</td> <td>1448443354</td> <td>0.982</td> <td>267</td> <td>50</td> <td>EFADT00931704</td> </tr> <tr> <td>EFA2012000099</td> <td>196</td> <td>0</td> <td>1448443337</td> <td>0</td> <td>0</td> <td>50</td> <td>EFADT00931704</td> </tr> <tr> <td>EFA2012000066</td> <td>205</td> <td>0</td> <td>1448443338</td> <td>0</td> <td>0</td> <td>50</td> <td>EFADT00931704</td> </tr> </tbody> </table>	Meter ID	VoltagePhase1	CurrentPhase1	Timestamp	Power Factor	Active Power	Frequency	Concentrator ID	EFA2012000172	216	3.4	1448443308	1	732	50	EFADT00931704	EFA2012000297	205	2.5	1448443327	1	585	50	EFADT00931704	EFA2012000242	199	1.6	1448443299	1	391	50	EFADT00931704	EFA2012000322	196	0.8	1448443354	0.982	267	50	EFADT00931704	EFA2012000099	196	0	1448443337	0	0	50	EFADT00931704	EFA2012000066	205	0	1448443338	0	0	50	EFADT00931704
Meter ID	VoltagePhase1	CurrentPhase1	Timestamp	Power Factor	Active Power	Frequency	Concentrator ID																																																		
EFA2012000172	216	3.4	1448443308	1	732	50	EFADT00931704																																																		
EFA2012000297	205	2.5	1448443327	1	585	50	EFADT00931704																																																		
EFA2012000242	199	1.6	1448443299	1	391	50	EFADT00931704																																																		
EFA2012000322	196	0.8	1448443354	0.982	267	50	EFADT00931704																																																		
EFA2012000099	196	0	1448443337	0	0	50	EFADT00931704																																																		
EFA2012000066	205	0	1448443338	0	0	50	EFADT00931704																																																		
Test outcome	OK																																																								
Additional comments	A high latency (18 seconds) has been experienced. The cause of it comes from the performance of the GSmart, not from the e-balance communication platform.																																																								

3.2 Stress tests

3.2.1 Simple write stress test

Test card	
Name/code/test case identifier	TEST.S001
Objectives	Execute 1000 write queries one after the other.
Devices involved	Customer Management Unit (CMU)
Pre-requirements	The user executing the test has access to the middleware due to the appropriate credentials
Steps	<ol style="list-style-type: none"> 1. Create an empty variable called TestVariable with one attribute named "Value" which is of type Integer 2. Execute 1000 (not concurrent) write operations over this variable as fast as possible. The value that will be written in the variable will increment after each write operation.
Expected result	A variable that contains the following values (property "value"): 0

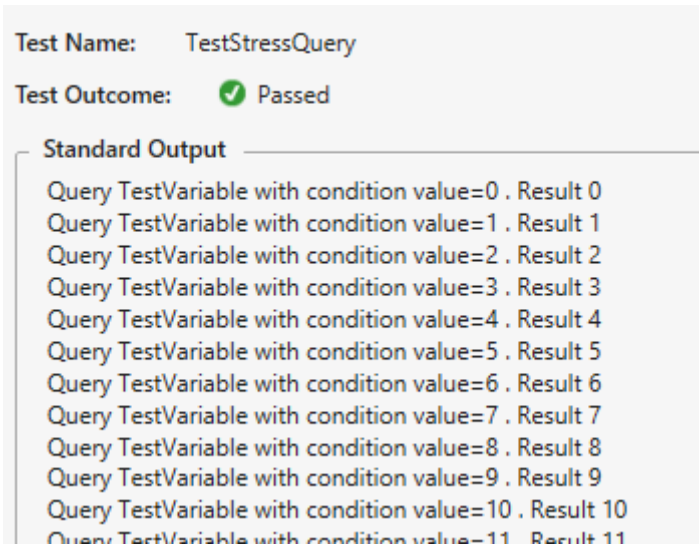
	1 2 3 ... 999
Additional comments	

Result card	
Name/code/test case identifier	TEST.S001
Results	 <p>A screenshot from the GUI shows that the 1000 values have been stored as expected</p>
Test outcome	OK
Additional comments	

3.2.2 Simple query stress test

Test card	
Name/code/test case identifier	TEST.S002
Objectives	Execute 1000 query operations one after the other.
Devices involved	Customer Management Unit (CMU)
Pre-requirements	<p>There exists a variable called TestVariable with a property named “value” of type integer with 1000 distinct values from 0 to 999 (TEST.S001 creates this pre-requirement)</p> <p>The user executing the test has access to the middleware due to the appropriate credentials</p>
Steps	<ol style="list-style-type: none"> i = 0; Query TestVariable with the following condition (value == i)

	<p>3. i++</p> <p>4. if(i < 1000) go to step 2</p>
Expected result	<p>Each query must return only one value. Values returned by the 1000 queries must be:</p> <p>0</p> <p>1</p> <p>...</p> <p>999</p>
Additional comments	

Result card	
Name/code/test case identifier	TEST.S002
Results	 <p>This screenshot shows the result of an automated test</p>
Test outcome	OK
Additional comments	

3.2.3 Concurrent write/query test

Test card	
Name/code/test case identifier	TEST.S003
Objectives	Execute two threads, one querying and the other writing a variable in the middleware. 1000 operations will be executed by each thread.
Devices involved	Customer Management Unit (CMU)
Pre-requirements	The user executing the test has access to the middleware due to the appropriate credentials
Steps	A thread writes values in a variable called “TestVariable”. This variable has one property named “value” of type int. Unique values from 0 to 999

	<p>will be written. A trace message is shown every time a value is written.</p> <p>A thread queries all the values in the “TestVariable” written so far. A trace message is shown with the values that have been queried.</p>
Expected result	<p>Write and read operations are concurrently executing.</p> <p>Traces show that write operations go from 0 to 999</p> <p>Traces show that read operations return the N results with values going from 0 to N where N is the value shown in the last write trace. For example, if the last write operation wrote a 10 the query operation must return 10 values ranging from 0 to 9.</p>
Additional comments	

Result card	
Name/code/test case identifier	TEST.S003
Results	<p>Test Name: ConcurrentStressTest1</p> <p>Test Outcome: ✔ Passed</p> <p>Standard Output</p> <pre> Debug Trace: Start W Start R R W0 R[0] W1 R[0][1] W2 R[0][1][2] R[0][1][2] W3 R[0][1][2][3] W4 R[0][1][2][3][4] W5 R[0][1][2][3][4][5] </pre> <p>This screenshot shows the result of an automated test. Lines with an R show the result of a query operation. Lines showing a W show the different write operations together with the value written in each operation.</p>
Test outcome	OK
Additional comments	

3.2.4 Concurrent write test

Test card	
Name/code/test case identifier	TEST.S004
Objectives	Execute two threads that perform 1000 write operations in the same variable.

Devices involved	Customer Management Unit (CMU)
Pre-requirements	The user executing the test has access to the middleware due to the appropriate credentials
Steps	Two threads write values in a variable called "TestVariable". This variable has one property named "value" of type int. The first thread writes unique even numbers from 0 to 1998. The second one writes unique odd numbers from 1 to 1999. A trace message is shown every time a value is written.
Expected result	2000 values are written in the variable "TestVariable". Values go from 0 to 1999.
Additional comments	Each thread executes one write each 200 ms

Result card	
Name/code/test case identifier	TEST.S004
Results	<div style="border: 1px solid #ccc; padding: 10px; background-color: #f9f9f9;"> <p>Test Name: ConcurrentStressTest2</p> <p>Test Outcome: ✔ Passed</p> <p>Standard Output</p> <pre> Debug Trace: Start 2W Start 1W 2W0 1W0 2W1 1W1 2W2 1W2 2W3 1W3 2W4 1W4 2W5 1W5 2W6 1W6 2W7 1W7 2W8 1W8 </pre> </div> <p>This screenshots shows the result of an automated test. Threads are called 1W and 2W. Each line indicates the id of the thread and the value written. E.g. 1W0 means that thread 1W writes a 0 value.</p>
Test outcome	OK
Additional comments	

3.3 Security tests

3.3.1 Access control test 1

Test card	
Name/code/test case identifier	TEST.X001
Objectives	Test the data interface API with different credentials
Devices involved	Customer Management Unit (CMU)
Pre-requirements	The tool “cURL” is installed in the system The credentials used for the user and admin tests already exist in the system
Steps	<ol style="list-style-type: none"> 1. Generate requests using cURL to query/write/query/periodic with no credentials 2. Generate requests to query/write/query/periodic with user credentials 3. Generate requests to query/write/query/periodic with admin credentials
Expected result	Normal users and admins are given access to the web services. Users without credentials are disallowed to use the web services
Additional comments	

Result card	
Name/code/test case identifier	TEST.X001
Results	<pre> curl -L http://garbo.adabyron.uma.es:8220/write?format=json -v --silent --stderr - grep HTTP > GET http://garbo.adabyron.uma.es:8220/write?format=json HTTP/1.1 * HTTP 1.0, assume close after body < HTTP/1.0 401 Unauthorized < Server: Mono-HTTPAPI/1.0 * HTTP/1.0 connection set to keep alive! curl -L http://garbo.adabyron.uma.es:8220/query?format=json -v --silent --stderr - grep HTTP > GET http://garbo.adabyron.uma.es:8220/query?format=json HTTP/1.1 * HTTP 1.0, assume close after body < HTTP/1.0 401 Unauthorized < Server: Mono-HTTPAPI/1.0 * HTTP/1.0 connection set to keep alive! curl -L http://garbo.adabyron.uma.es:8220/event?format=json -v --silent --stderr - grep HTTP > GET http://garbo.adabyron.uma.es:8220/event?format=json HTTP/1.1 * HTTP 1.0, assume close after body < HTTP/1.0 401 Unauthorized < Server: Mono-HTTPAPI/1.0 * HTTP/1.0 connection set to keep alive! curl -L http://garbo.adabyron.uma.es:8220/periodic?format=json -v --silent --stderr - grep HTTP > GET http://garbo.adabyron.uma.es:8220/periodic?format=json HTTP/1.1 * HTTP 1.0, assume close after body < HTTP/1.0 401 Unauthorized < Server: Mono-HTTPAPI/1.0 * HTTP/1.0 connection set to keep alive! </pre> <p>No operations are allowed without credentials</p>

	<pre> curl -L http://garbo.adabvtron.uma.es:8220/write?format=json -v --silent --stderr - -u <user>:<pass> grep HTTP > GET http://garbo.adabvtron.uma.es:8220/write?format=json HTTP/1.1 * HTTP 1.0, assume close after body < HTTP/1.0 200 OK < Server: Mono-HTTPAPI/1.0 curl -L http://garbo.adabvtron.uma.es:8220/query?format=json -v --silent --stderr - -u <user>:<pass> grep HTTP > GET http://garbo.adabvtron.uma.es:8220/query?format=json HTTP/1.1 * HTTP 1.0, assume close after body < HTTP/1.0 200 OK < Server: Mono-HTTPAPI/1.0 curl -L http://garbo.adabvtron.uma.es:8220/event?format=json -v --silent --stderr - -u <user>:<pass> grep HTTP > GET http://garbo.adabvtron.uma.es:8220/event?format=json HTTP/1.1 * HTTP 1.0, assume close after body < HTTP/1.0 200 OK < Server: Mono-HTTPAPI/1.0 curl -L http://garbo.adabvtron.uma.es:8220/periodic?format=json -v --silent --stderr - -u <user>:<pass> grep HTTP > GET http://garbo.adabvtron.uma.es:8220/periodic?format=json HTTP/1.1 * HTTP 1.0, assume close after body < HTTP/1.0 200 OK < Server: Mono-HTTPAPI/1.0 curl -L http://garbo.adabvtron.uma.es:8220/write?format=json -v --silent --stderr - -u <admin>:<pass> grep HTTP > GET http://garbo.adabvtron.uma.es:8220/write?format=json HTTP/1.1 * HTTP 1.0, assume close after body < HTTP/1.0 200 OK < Server: Mono-HTTPAPI/1.0 curl -L http://garbo.adabvtron.uma.es:8220/query?format=json -v --silent --stderr - -u <admin>:<pass> grep HTTP > GET http://garbo.adabvtron.uma.es:8220/query?format=json HTTP/1.1 * HTTP 1.0, assume close after body < HTTP/1.0 200 OK < Server: Mono-HTTPAPI/1.0 curl -L http://garbo.adabvtron.uma.es:8220/event?format=json -v --silent --stderr - -u <admin>:<pass> grep HTTP > GET http://garbo.adabvtron.uma.es:8220/event?format=json HTTP/1.1 * HTTP 1.0, assume close after body < HTTP/1.0 200 OK < Server: Mono-HTTPAPI/1.0 curl -L http://garbo.adabvtron.uma.es:8220/periodic?format=json -v --silent --stderr - -u <admin>:<pass> grep HTTP > GET http://garbo.adabvtron.uma.es:8220/periodic?format=json HTTP/1.1 * HTTP 1.0, assume close after body < HTTP/1.0 200 OK < Server: Mono-HTTPAPI/1.0 </pre> <p>Operations are allowed when using normal user or admin credentials</p>
Test outcome	OK
Additional comments	

3.3.2 Access control test 2

Test card	
Name/code/test case identifier	TEST.X002
Objectives	Test administration web services with different credentials
Devices involved	Customer Management Unit (CMU)
Pre-requirements	The tool “cURL” is installed in the system The credentials used for the user and admin tests already exist in the system
Steps	<ol style="list-style-type: none"> 1. Generate requests using cURL to createVariable/clearVariable/deleteVariable with no credentials 2. Generate requests to createVariable/clearVariable/deleteVariable with user credentials 3. Generate requests to createVariable/clearVariable/deleteVariable with admin credentials
Expected result	Only admins are given access to these web services. Users without credentials or normal users are disallowed to use these web services
Additional comments	

Result card		
Name/code/test identifier	case	TEST.X002
Results		<pre> curl -L http://garbo.adabyron.uma.es:822/clear_variable?format=json -v --silent --stderr - grep HTTP > GET http://garbo.adabyron.uma.es:8220/clear_variable?format=json HTTP/1.1 * HTTP/1.0, assume close after body < HTTP/1.0 401 Unauthorized < Server: Mono-HTTPAPI/1.0 * HTTP/1.0 connection set to keep alive! curl -L http://garbo.adabyron.uma.es:822/delete_variable?format=json -v --silent --stderr - grep HTTP > GET http://garbo.adabyron.uma.es:8220/delete_variable?format=json HTTP/1.1 * HTTP/1.0, assume close after body < HTTP/1.0 401 Unauthorized < Server: Mono-HTTPAPI/1.0 * HTTP/1.0 connection set to keep alive! curl -L http://garbo.adabyron.uma.es:822/create_variable?format=json -v --silent --stderr - grep HTTP > GET http://garbo.adabyron.uma.es:8220/create_variable?format=json HTTP/1.1 * HTTP/1.0, assume close after body < HTTP/1.0 401 Unauthorized < Server: Mono-HTTPAPI/1.0 * HTTP/1.0 connection set to keep alive! curl -L http://garbo.adabyron.uma.es:8220/create_variable?format=json -v --silent --stderr - -u <user>:<pass> grep HTTP > GET http://garbo.adabyron.uma.es:8220/clear_variable?format=json HTTP/1.1 * HTTP/1.0, assume close after body < HTTP/1.0 403 Forbidden < Server: Mono-HTTPAPI/1.0 * HTTP/1.0 connection set to keep alive! curl -L http://garbo.adabyron.uma.es:822/delete_variable?format=json -v --silent --stderr - -u <user>:<pass> grep HTTP > GET http://garbo.adabyron.uma.es:8220/delete_variable?format=json HTTP/1.1 * HTTP/1.0, assume close after body < HTTP/1.0 403 Forbidden < Server: Mono-HTTPAPI/1.0 * HTTP/1.0 connection set to keep alive! curl -L http://garbo.adabyron.uma.es:822/create_variable?format=json -v --silent --stderr - -u <user>:<pass> grep HTTP > GET http://garbo.adabyron.uma.es:8220/create_variable?format=json HTTP/1.1 * HTTP/1.0, assume close after body < HTTP/1.0 403 Forbidden < Server: Mono-HTTPAPI/1.0 * HTTP/1.0 connection set to keep alive! No operations are allowed without credentials or with normal user credentials curl -L http://garbo.adabyron.uma.es:822/clear_variable?format=json -v --silent --stderr - -u <admin>:<pass> grep HTTP > GET http://garbo.adabyron.uma.es:8220/clear_variable?format=json HTTP/1.1 * HTTP/1.0, assume close after body < HTTP/1.0 200 OK < Server: Mono-HTTPAPI/1.0 curl -L http://garbo.adabyron.uma.es:822/delete_variable?format=json -v --silent --stderr - -u <admin>:<pass> grep HTTP > GET http://garbo.adabyron.uma.es:8220/delete_variable?format=json HTTP/1.1 * HTTP/1.0, assume close after body < HTTP/1.0 200 OK < Server: Mono-HTTPAPI/1.0 curl -L http://garbo.adabyron.uma.es:822/create_variable?format=json -v --silent --stderr - -u <admin>:<pass> grep HTTP > GET http://garbo.adabyron.uma.es:8220/create_variable?format=json HTTP/1.1 * HTTP/1.0, assume close after body < HTTP/1.0 200 OK < Server: Mono-HTTPAPI/1.0 Operations are allowed when admin credentials are used </pre>
Test outcome		OK
Additional comments		

3.3.3 Management unit registration in the group management – obtaining the certificate from the Certification Authority

Test card	
Name/code/test case identifier	TEST.X003
Objectives	Check if the registration process is working correctly. In this test the focus is on the registration step one – obtaining the certificate from the certification authority.
Devices involved	Customer Management Unit (CMU) – device that wants to register at the CA. Certification Authority (CA) server.
Pre-requirements	At the CA proper means (credentials) for authorising the generation of the certificate are configured: user/password or token.
Steps	<ol style="list-style-type: none"> 1. The CMU connects to the CA and provides the CMU information and credentials. 2. The CA verifies the credentials and generates (and stores) the certificate, in case the results of the tests were positive. 3. The CMU receives an acknowledgement message and stores the outcome of the operation (CA certificate, CMU certificate).
Expected result	Using the authorized credentials the CMU receives the certificates
Additional comments	

Result card	
Name/code/test case identifier	TEST.X003

Results

```

CA registration started
Choose certificate type: HW (1) or USER (2)
1
You have chosen HW certificate generation. Do You want to continue? (yes/no)
yes
Certificate generation started
PublicKey for 127.0.0.1: Sun RSA public key, 2048 bits
modulus:
184563811780787617468737817215968675656308867486916359695595460623349745949681
590555424611126505306021434430497652283487814079368996736216676967716438202222
153278249295452998953170964279687885216671155768402808537345257505989350453116
798895055447620113440924723214783399529684325562731711561291524533924406649735
484602374085446707315387604536966434637475771805887793140546766516549245274770
757146424673302148882145974419386711893333353850336730418307166079733298508128
333356766446038708376828956779572698774557268754966300517974354650588881147695
75708411441305010906420080273213305038591891150767723397140236675378911
public exponent: 65537
Connecting to CA server...
Connection to CA server was successful at Wed Feb 24 10:16:21 CET 2016
Sending registration data from 127.0.0.1 to CA...
Enter your token:
127001
Registration of HW Certificate for 127.0.0.1 was successful at Wed Feb 24
10:16:21 CET 2016
Credential for 127.0.0.1 saved.
Your HW signed Certificate: [
[
Version: V3
Subject: CN=127.0.0.1, OU=SY, O=IHP, L=FFO, ST=Brandenburg, C=DE
Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

Key: Sun RSA public key, 2048 bits
modulus:
184563811780787617468737817215968675656308867486916359695595460623349745949681
590555424611126505306021434430497652283487814079368996736216676967716438202222
153278249295452998953170964279687885216671155768402808537345257505989350453116
798895055447620113440924723214783399529684325562731711561291524533924406649735
484602374085446707315387604536966434637475771805887793140546766516549245274770
757146424673302148882145974419386711893333353850336730418307166079733298508128
333356766446038708376828956779572698774557268754966300517974354650588881147695
75708411441305010906420080273213305038591891150767723397140236675378911
public exponent: 65537
Validity: [From: Wed Feb 24 10:16:34 CET 2016,
To: Sat Mar 12 10:57:03 CET 2016]
Issuer: CN=IHP_CA, OU=CY, O=IHP, L=FFO, ST=Brandenburg, C=DE
SerialNumber: [ 2adcc839 b289c4ff]

Certificate Extensions: 3
[1]: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
CA:false
PathLen:0
]

[2]: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
DigitalSignature
]

[3]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: DB 74 D4 1A E6 61 C0 AE 60 9C F5 B8 4E 71 CD 85 .t...a...`...Nq..
0010: 32 DE 22 A3 2."
]
]
]
Algorithm: [SHA1withRSA]
Signature:
0000: 0E 92 6B FB 84 75 C9 18 F0 9C C0 93 54 09 E8 64 ..k.u.....T..d
0010: D5 21 22 1E 42 DB B5 A9 0C F1 5B E5 2D C4 37 70 !".B.....[-.7p
0020: 12 59 C7 7E 95 9E B6 D3 90 EF 69 6B 65 E5 6A B2 .Y.....ike.j.
0030: 0A A4 AF 84 4A 5A 4E 85 4E 05 78 33 7A 59 FF AC ....JZN.N.x3zY..
0040: 61 A1 1D 6E 19 0A 33 02 AE 8E 83 75 82 C3 53 00 a..n..3.....u..S.
0050: 51 91 BC D8 96 72 D6 DA ED A8 E7 AC 63 97 11 DE Q.....r.....c....
0060: 7D F8 29 50 AE 60 BD EA C6 6E 70 35 42 BF 07 31 ..)P.....np5B..1
0070: FC 8E 6E 8F BD 7C A9 AB CE 64 01 27 C1 54 B4 56 ..n.....d.'..T.V
0080: CD FD 2A 05 DA F3 6F F2 39 EA FE DB 16 67 CB 81 ...*...o.9....g...
0090: BD 5C A0 CD EF A6 E8 F5 C4 E9 48 28 13 BD 1A 98 .\.....H(....
00A0: 55 62 8C 01 E7 7C 7E 0F 35 A6 83 5A 00 68 36 80 Ub.....S...Z.h6.
00B0: E4 A5 8B C4 14 04 30 33 00 CA 4C 37 50 87 AB 18 .....03...L7P...
00C0: 0E 6C 3A C2 5D 58 E7 F7 D7 C5 6D 96 D1 3D 8D B4 .l:]X.....m..=.
00D0: 10 74 CB 6A 90 9A 03 CD AA BB 8B 24 D2 35 5A 50 .t.j.....$.SZP
00E0: 78 2F 8F 10 87 54 20 A4 7C 86 6D A7 52 47 78 DE x/...T ...m.RGx.
00F0: 02 65 91 12 91 2D 83 1C 3D 5A 80 48 A4 63 22 34 .e.....=Z.H.c"4
]

CA registration finished. Goodbye.

```


Test outcome	OK
Additional comments	

3.3.4 Management unit registration in the group management – registration approach without proper credentials

Test card	
Name/code/test case identifier	TEST.X004
Objectives	Check if the registration process is working correctly. In this test the focus is on the protection against registration approach without proper credentials.
Devices involved	Customer Management Unit (CMU) – device that wants to register at the CA Certification Authority (CA) server.
Pre-requirements	At the CA proper means (credentials) for authorising the generation of the certificate are configured: user/password or token.
Steps	<ol style="list-style-type: none"> 1. The CMU connects to the CA and provides the CMU information and credentials. 2. The CA verifies the credentials and generates (and stores) the certificate, in case the results of the tests were positive. 3. The CMU receives an acknowledgement message and stores the outcome of the operation (CA certificate, CMU certificate).
Expected result	Using unauthorized credentials causes the operation to fail
Additional comments	

Result card	
Name/code/test case identifier	TEST.X004
Results	<pre> CA registration started Choose certificate type: HW (1) or USER (2) 1 You have chosen HW certificate generation. Do You want to continue? (yes/no) yes Certificate generation started PublicKey for 127.0.0.1: Sun RSA public key, 2048 bits modulus: 193376926257822195908739774514702304250175417237075272821097235954869461658941 126942931685653160997625865631362851434483411549767129346245040370503975241074 539630072076354065968251681577849473402969826232698336759559185891606953151412 108542583721544355766440664954129949045152650613215066720634078664221916337534 758098665741423079904435521363547319099671669404897403884639486113220637639386 516488773851735421670895869115215669632015681381442932988994205819057860548166 395400218357371764914466919225578101632645267112309658621960947425117980543722 02714238516920322040233489651995075996928761738080024680732202490410129 public exponent: 65537 Connecting to CA server... Connection to CA server was successful at Wed Feb 24 10:20:51 CET 2016 Sending registration data from 127.0.0.1 to CA... Enter your token: gjl676444 Registration of USER Certificate for 127.0.0.1 failed. Token doesn't match. CA registration finished. Goodbye. </pre>
Test outcome	OK
Additional comments	

3.4 Test with multiple CMUs

3.4.1 Broadcast Query

Test card	
Name/code/test case identifier	TEST.M001
Objectives	Execute a broadcast query that will retrieve information from those child CMUs that depend on an LVGMU. In this test, the LVGMU has three children and will retrieve 50 data tuples from each one.
Devices involved	LVGMU, Customer Management Unit (CMU)
Pre-requirements	The user executing the test has access to the middleware due to the appropriate credentials. There are three CMUs (children) registered in the LVGMU. Each CMU contains a variable called ENERGY_CONSUMPTION which has 50 data tuples.
Steps	<p>1. Call the “QueryFromChildren” API function to get information of the ENERGY_CONSUMPTION variable from all the CMUs (children) registered in a LV (caller).</p> <pre> public static void QueryFromChildren(){ Map<EUrl, QueryResponse> res = EbalanceAPI.QueryFromChildren("ENERGY_CONSUMPTION", "*", ""); System.out.println("Number of children that have answered correctly: " + res.size()); for (Map.Entry<EUrl, QueryResponse> entry: res.entrySet()){ if (entry.getValue().getOperationResult().isSuccess()){ System.out.println("Child " + entry.getKey().ip + " has returned " + entry.getValue().getData().size() + " tuples"); } } } </pre>
Expected result	All three children will return 50 data tuples
Additional comments	

Result card	
Name/code/test case identifier	TEST.M001
Results	<pre> Number of children that have answered correctly: 3 Child 192.168.43.75 has returned 50 tuples Child 192.168.43.74 has returned 50 tuples Child 192.168.43.51 has returned 50 tuples </pre>
Test outcome	OK
Additional comments	

3.4.2 Remote procedure emulation

Test card	
Name/code/test case identifier	TEST.M002
Objectives	Emulate a remote procedure call by means of the e-balance middleware which follows a data-centric approach.

	<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px; width: 45%;"> <p style="text-align: center; margin: 0;">CMU</p> <div style="border: 1px solid black; padding: 5px; margin: 5px;"> <p style="margin: 0;">Variable: TEST2</p> </div> <p style="margin: 5px 0;">2. Remote procedure App</p> <div style="border: 1px solid black; padding: 5px; margin: 5px;"> <p style="margin: 0;">Increment value of TEST2 and write TEST1</p> </div> </div> <div style="border: 1px solid black; padding: 5px; width: 45%;"> <p style="text-align: center; margin: 0;">LVGMU</p> <div style="border: 1px solid black; padding: 5px; margin: 5px;"> <p style="margin: 0;">Write values in TEST2</p> </div> <p style="margin: 5px 0;">4. Get remote procedure result</p> <div style="border: 1px solid black; padding: 5px; margin: 5px;"> <p style="margin: 0;">Get and print results</p> </div> <p style="margin: 5px 0;">Variable: TEST1</p> </div> </div>
--	--

Result card	
Name/code/test case identifier	TEST.M002

<p>Results</p>	<p>Output from the LVGMU program</p> <pre> WRITING 994 VALUE RECEIVED: 995 WRITE ok TEST2 994 WRITING 995 VALUE RECEIVED: 996 WRITE ok TEST2 995 WRITING 996 VALUE RECEIVED: 997 WRITE ok TEST2 996 WRITING 997 VALUE RECEIVED: 998 WRITE ok TEST2 997 WRITING 998 VALUE RECEIVED: 999 WRITE ok TEST2 998 WRITING 999 VALUE RECEIVED: 1000 WRITE ok TEST2 999 </pre> <p>Output from the CMU program</p> <pre> value received 997 Value incremented 997 WRITING BACK 997 WRITE BACK OK TEST1 997 Value received 997 Value incremented 998 WRITING BACK 998 WRITE BACK OK TEST1 998 Value received 998 Value incremented 999 WRITING BACK 999 WRITE BACK OK TEST1 999 Value received 999 Value incremented 1000 WRITING BACK 1000 WRITE BACK OK TEST1 1000 </pre> <p>All output has been analysed and the results obtained are similar to the expected ones.</p>
<p>Test outcome</p>	<p>OK</p>
<p>Additional comments</p>	

4 Summary and conclusions

This document presents the integration and evaluation tasks for the communication platform designed and implemented in WP4. It describes the integration process of the different modules developed in WP4, namely the communication platform, security mechanisms, programming languages wrappers to allow communication with other programming languages and some external devices such as the GSmart, provided by EFA which is a device used to control the grid sensors and measurements. A set of tests have been defined and evaluated, according to a set of tests cards which define the expected behaviour of the system when a specific input is provided. These tests have been used to check the correctness of the prototype implementation of the communication platform. They are also useful to validate new changes and future updates in the system.

References

- [1] A. Grilo, et al., “Deliverable D4.1 – Detailed network stack specification and implementation”, Public deliverable of e-balance project, FP7-Smartcities-2013, Project number 609132, 2015.
- [2] K. Piotrowski, et al., “Deliverable D4.2 – Detailed security and privacy specification and implementation”, Public deliverable of e-balance project, FP7-Smartcities-2013, Project number 609132, 2015.
- [3] 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.
- [4] J.J Peralta, et al., “Deliverable D2.4 – Stakeholder requirements”, Public deliverable of e-balance project, FP7-Smartcities-2013, Project number 609132, 2014.
- [5] M. Gerards, M. Jongerden, et al., “Deliverable D3.1 - High Level System Architecture Specification”, Public deliverable of e-balance project, FP7-Smartcities-2013, Project number 609132, 2014.
- [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.
- [7] ServiceStack framework. <https://servicestack.net/>
- [8] Mono. Cross platform, open source .NET framework. <http://www.mono-project.com>
- [9] Beaglebone Black. <http://www.beagleboard.org>
- [10] Jetty java web server. <http://eclipse.org/jetty>
- [11] AngularJS application framework. <http://angularjs.org>
- [12] M. Geers et al., “Deliverable D6.1 – Specification of the demonstrators”, Public deliverable of e-balance project, FP7-Smartcities-2013, Project number 609132, 2014.