D5.3 - Aggregator Infrastructure Integration and Verification

Document ID:	SEMIAH-WP5-D5.3-v1.1
Document version:	1.11
Document status:	Final
Dissemination level:	PU
Deliverable number:	D5.3
Deliverable title:	Aggregator infrastructure integration and verification
WP number:	WP5
Lead beneficiary:	DEVOTEAM
Main author(s):	Harald Loland, Jarle Einar Qvigstad (DEVOTEAM), and Stefan Siegl (FRAUNHOFER)
Nature of deliverable:	R
Delivery date from Annex 1:	M24
Actual delivery date:	
Funding scheme / call:	STREP-FP7-ICT-2013-11
Project / GA number:	619560
Project full title:	Scalable Energy Management Infrastructure for Aggregation of Households
Project start date:	01/03/2014
Project duration:	36 months

Funded by the European Union

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 619560.

Executive Summary

Aggregator infrastructure integration and verification of the SEMIAH project provides the server infrastructure for both the system integration stage and for the production stage for the back-end system of SEMIAH. These servers are integrated and then defined as a back-end software release. The back-end software release, based on SaaS release labels and Bitbucket commit labels, is tested prior to labels are set, but just slightly after the back-end software release is defined, due to test setup problems and lack of time. A list of test cases relevant for a back-end software release is found in Annex part.

Abbreviations

API	Application Programming Interface
AWS	Amazon Web Services
CM	Configuration Management
D	Deliverable
DoW	Description of Work
DR	Demand Response
DSO	Distribution System Operator
EMG	Energy Manager Gateway
GUI	Graphical User Interface
GVPP	Generic Virtual Power Plant
HEMG	Home Energy Manager Gateway
HTTP	Hyper Text Transfer Protocol
HTTPS	Secure Hyper Text Transfer Protocol
HW	HardWare
IEC	International Electrotechnical Commission
IT	Information Technology
IWES	Institute for Wind Energy and energy System technology
OGEMA	Open Gateway Energy MAnagement
PC	Personal Computer
REST	Representation State Transfer
SALSA	Scalable Aggregation of Load Schedulable Appliances
SEMIAH	Scalable Energy Management Infrastructure for Aggregation of Households
SOAP	Service-oriented Architecture Protocol
SQUID	Superconducting QUantum Interference Devices
SW	Software
VPP	Virtual Power Plant
WP	Work Package
WT	Work Task
XML	eXtendeble Markup Language

Contents

List of Figures

List of Tables

1 Introduction

The purpose of this document is to provide the SEMIAH back-end server infrastructures and ensuring the integration of these servers into SEMIAH back-end releases using a process with a related verification structure.

1.1 Motivation

The main motivations of this deliverable are to provide the back-end servers as integrated and bundled as a SEMIAH back-end release that works as a whole. This also reduces the integration risk when integrating a limited and functionally related set of servers, instead of many more or all servers planned for SEMIAH at once. This deliverable also paves the way for handling the release process and triggers the system integration and system integration phases and give precedence and experience to future integration of the remaining parts of SEMIAH system. As a first back-end component, the release of the SEMIAH Intelligence encompasses the integration of two parts named the Scalable Aggregation of Load Schedulable Appliances (SALSA) algorithm and Distribution System Operator (DSO) Grid Constraints. SALSA algorithm is an extension of the algorithm proposed in Deliverable D5.1. This is used inside the SEMIAH Intelligence. As the second back-end component, the release and integration of Flexibility Forecast (FF) and Generic Virtual Power Plant (GVPP) is ensured. The SEMIAH back-end can be operated by, for instance, an aggregator role, who intends to exploit the flexibility of the electricity consumption of a large (or for functional test a small) number of households.

1.2 Objectives

The main objective of the SEMIAH project is to deliver a Demand Response solution for control of electrical loads at a competitive price. Work Package 5 focuses on developing the aggregator backbone infrastructure that is based on a server capacity that registers and manages the flexible electricity consumptions offered by the customers. The Back-end infrastructure provides an interface towards the front-end and is the engine of the system operations. Users register electrical loads which are subjected to intelligent load control (SEMIAH Intelligence). Therefore, this document presents the back-end infrastructure showing its releases and how they are integrated into the infrastructure. It integrates the functional modules developed in Tasks 5.1 to 5.4 and packages these into software releases. The software is maintained in a source code repository under version control as defined in the configuration management strategy from WP3.

1.3 Achievements

During the second year, regarding the first component, a SEMIAH intelligence system for scheduling load requests, including SALSA algorithm integrated with DSO grid constraints, has been released. SALSA offers an event-based automated demand response utilizing a buffering system to locate diverse load requests of appliances in different buffers which accelerates the SALSA's responding function. Furthermore, DSO grid constraints web application for defining the DSO's boundaries has been developed and tested. More in detail, these constraints have been integrated with SEMIAH Intelligence to fetch grid constraints. Regarding the second component, the Flexibility Forecast model has been released which can forecast the electricity consumption of SEMIAH consumers and calculate the amount of flexibility that a household can provide to the SEMIAH system.

1.4 Outline

This deliverable is organised as follows. Chapter 2 gives an overview of the components used in the back-end. Chapter 3 provides the back-end server infrastructure. Chapter 4 describes the back-end integration of software components and the process of this. Chapter 5 defines back-end test configurations. Chapter 6 defines System Integration and System Test Tasks.

2 Back-End System Description

This chapter is an overall summary of how the back-end infrastructure, features and interfaces to be integrated and verified. Refer to the Interface, Feature and Technical design description documents and the D5.4 Back-end System Release [4] for more information. This is based on the implementation of Tasks WT5.1 to WT5.4.

- An IT aggregator infrastructure for smart grid
- Algorithms for intelligent load control and scheduling
- Algorithms for probabilistic load forecasts from households
- DSO services / data interfaces.
- Intelligence Algorithm Insertion

2.1 Components

The Virtual Power Plant (VPP) back-end system consists of three main parts:

1. VPP-Back-end
2. VPP-Front-end
3. Database

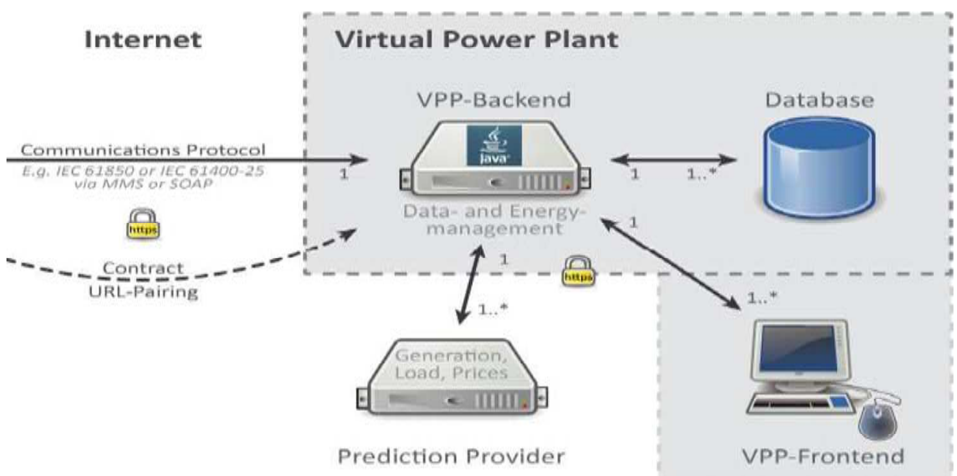


Figure 1: The main parts of the Virtual Power plant

These components are designed as Figure 2 shows.

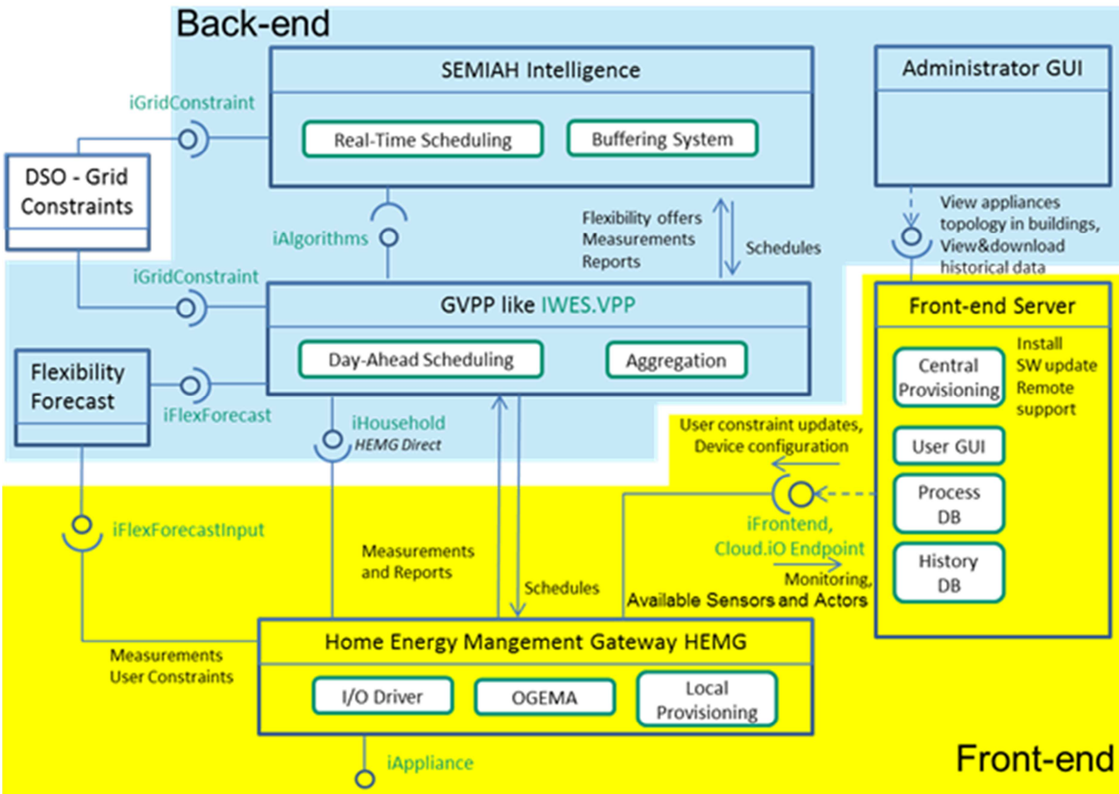


Figure 2: SEMIAH technical architecture presenting its components

2.1.1 VPP-Back-End

The VPP-Back-end is hosted as SaaS on a server from Fraunhofer and is under test hosted as SaaS on the server applik-d19.iwes.fraunhofer.de from Fraunhofer, which is under joint controlled release-label-and-version-control-upgrade:

- Role: Aggregation and Scheduling
- Dependencies
- Interfaces

The REST service is available here:

https://applik-d19.iwes.fraunhofer.de:8443/RESTfulService/

2.1.2 Database

The VPP-Back-end Database is a MongoDB database integrated into the back-end. This database is not directly accessible via the SaaS interface.

2.1.3 VPP-Front-End Client Tool

The IWES VPP-Front-end Client tool is hosted on a local windows PC, and can be downloaded and installed from:

https://applik-d19.iwes.fraunhofer.de:8443/vppcore/home.xhtml

Note that this is IWES VPP Client tool is used to read (or push test data) for a household directly

Role: Readout load time series data of and levels (like 'Nominal Power' and 'Supplied Installed Electrical Power' shown in Annex D - Environment Integration Configuration) from/for households Defining the household Username and household Password that is normally used by OGEMA to access VPP can be seen in Figure 33.

2.1.4 DSO Service/Data Interface

This interface, which is known as DSO grid constraints, contributes to Task 5.4. It is a service for configuring grid constraints for a collection of households belonging to a SEMIAH aggregator. The interface is used by DSOs to define boundaries for the operation of the scheduling of power consumption for the Collection. This interface complies with the operating regimes as defined by Universal Smart Energy Framework (USEF). Currently, the service supports the *soft* and *hard* electricity consumption thresholds. The design makes use of the Google Maps APIs. The native platforms: Web, Android and iOS supports the Google Maps API and uses HTTP-based web services. During the second year, this interface was launched. It has been implemented with Python and HTML. It is mainly used by DSOs to define boundaries for the operation of the scheduling of power consumption for the Collection. At this stage, DSOs interact with this interface manually. They update the threshold values and then, the interface provides the corresponding XML file. For the third year, it has been decided to update this interface with Restful API service programming. Using this, DSOs or any other third-party services, for instance SALSA or GVPP, can interact with it automatically. This will definitely induce performance, scalability, and reliability. Also, it is aimed to adapt the current grid's topology on the concept of Collections used in this interface. This can fundamentally be done using Google Maps. DSO Grid Constraints web service is hosted on a Linux Ubuntu Apache2 Web server at:

http://radagast4.netlab.eng.au.dk/semiah/

It can be fetched and installed from

https://bitbucket.org/semiah/dsoif

- Role: Setting Grid constraints and providing an interface for readout of Grid constraints
- Dependencies
- Interfaces

2.1.5 Flexibility Forecast

The Flexibility Forecast is hosted as SaaS on a server from Fraunhofer and is under test hosted as SaaS on a server e.g. applik-d19.iwes.fraunhofer.de (or later on another separate server) from Fraunhofer that is under joint controlled release-label-and-version-control-upgrade:

- Role: Matures reporting from the household into probabilistic load forecasts from households
- Dependencies
- Interfaces

2.1.6 SEMIAH Intelligence

SEMIAH Intelligence mainly contributes to Task 5.3. With respect to D4.3, it includes a Scalable Aggregation of Load Schedulable Appliances (SALSA) algorithm and a buffering system. During the second year, SALSA has been updated for several times. The major change was the transfer from a procedural programming to the object-oriented programming. Furthermore, preliminary studies of how to adapt SALSA to current grid's topology have been made. It has been decided to propose a hierarchical multi-layer version of SALSA in the third year. This version will activate the scalability feature of SEMIAH using an event-driven demand response approach. Another major update will be integrating various demand response communication protocols with SALSA to enable a complete software platform. The final major update will include developing a multi-layer Knapsack concept and integrating it with SALSA to manage the top-down data flow from the DSO level to household level. This requires using the virtualization concept in the simulations. Version 1.0 of SALSA is hosted on the Bitbucket and can be fetched and installed from

https://bitbucket.org/semiah/salsa

- Role: Updates the schedules of the VPP.
- Dependencies
- Interfaces

3 Back-End Server Infrastructure

The following presents what back-end Server Infrastructure encompasses includes, where the development/integration and system test/pilots are separated.

3.1 Server Topologies

Figure 3 shows current back-end Server Infrastructure Feature Development combined with System Integration. Figure 4 and Figure 5 show the topology where some servers in the future possibilities are moved to a common cloud server AWS environment. OGEMA with drivers is in the process of being moved to the SQUID HEMG.

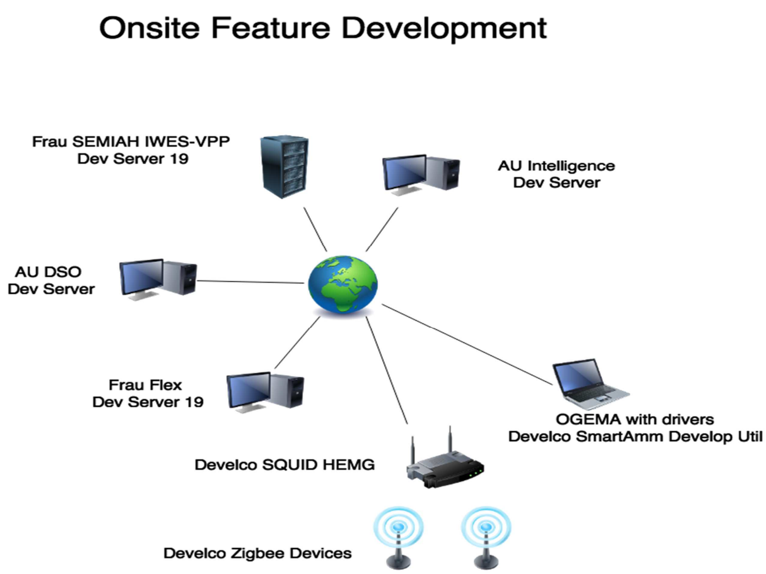


Figure 3: The SEMAIH back-end focused topology with all on-site severs

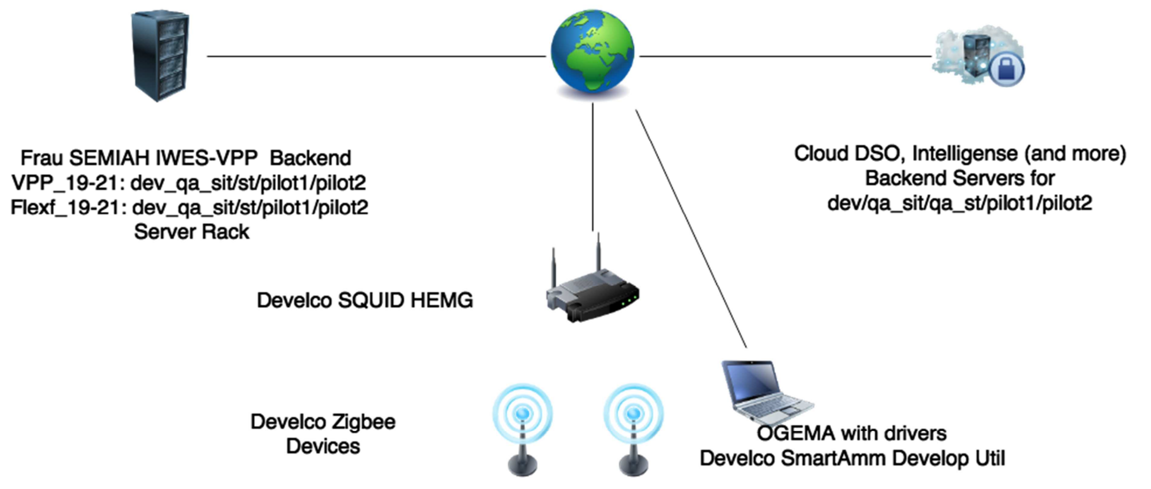


Figure 4: SEMIAH back-end focused topology with some SaaS and some cloud_or_local site servers

3.2 Back-End Server Infrastructure Per Integration Phase

The SW releases flow from server releases via system integration and into system test that at one stage reach pilots where each of these steps is considered a phase. This could be understood as the “system integration stage” and the “production stage” stages.

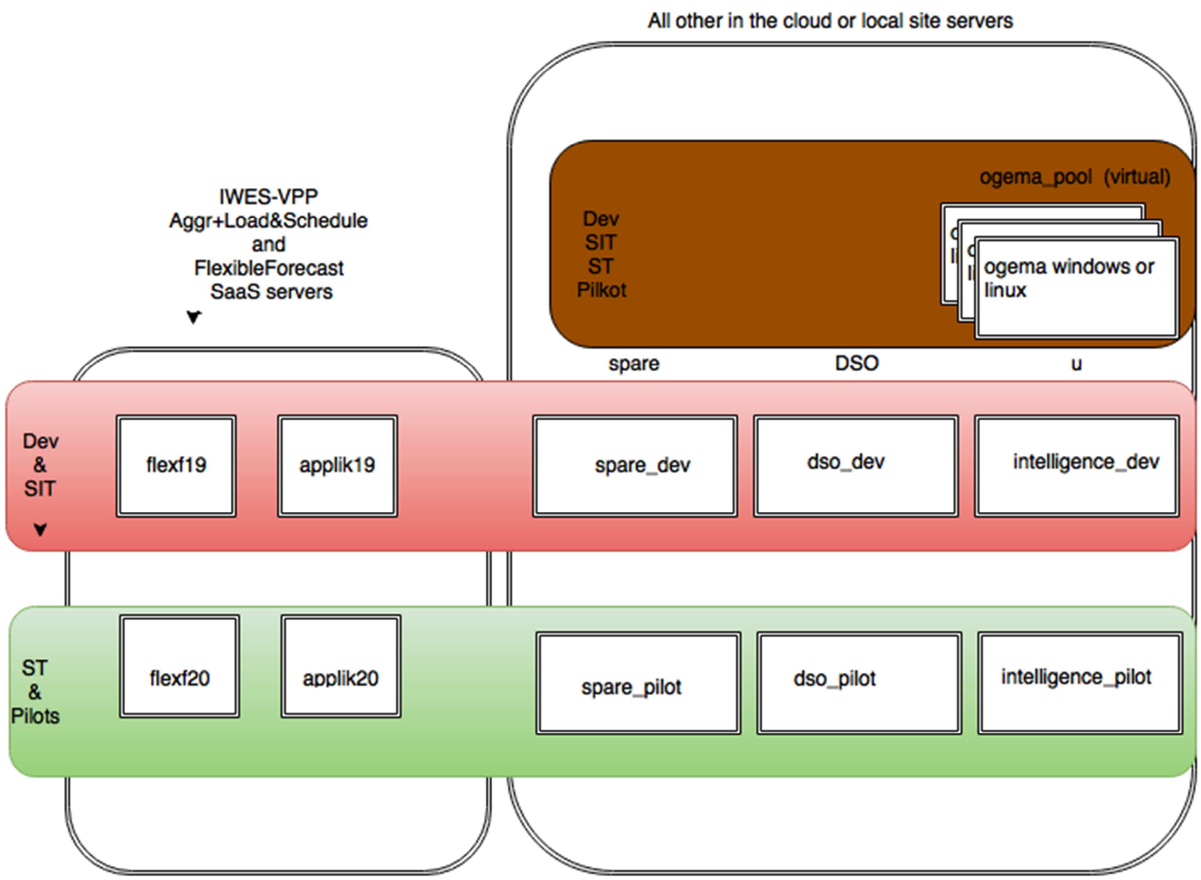


Figure 5: The SEMIAH back-end server infrastructure per Integration Stage

3.3 Back-End Server Infrastructure Channel Booking from Amazon

The cloud server infrastructure for the cloud servers from Amazon Web Services (AWS) can be booked by sending a request to:

semiah_book_test_channel@googlegroups.com

using template test_channel_booking.txt as defined in

https://bitbucket.org/semiah/system_integration

The cost per site or per subproject/main project needs to be addressed.

4 SW Integration of Back-End Servers

The process of how the Back-end Servers are integrated is described here and this is assumed to be the process for the remaining of integration of servers in SEMIAH. This section will describe the current status and plans for the integration and verification of the infrastructure. Within the project there exist two scheduling algorithms. One is implemented and running within the GVPP and the other one is running external. The first one is called stochastic optimization and is implemented by Fraunhofer. The second one is called SALSA and is implemented by the AU. Both version of the scheduling can run independently.

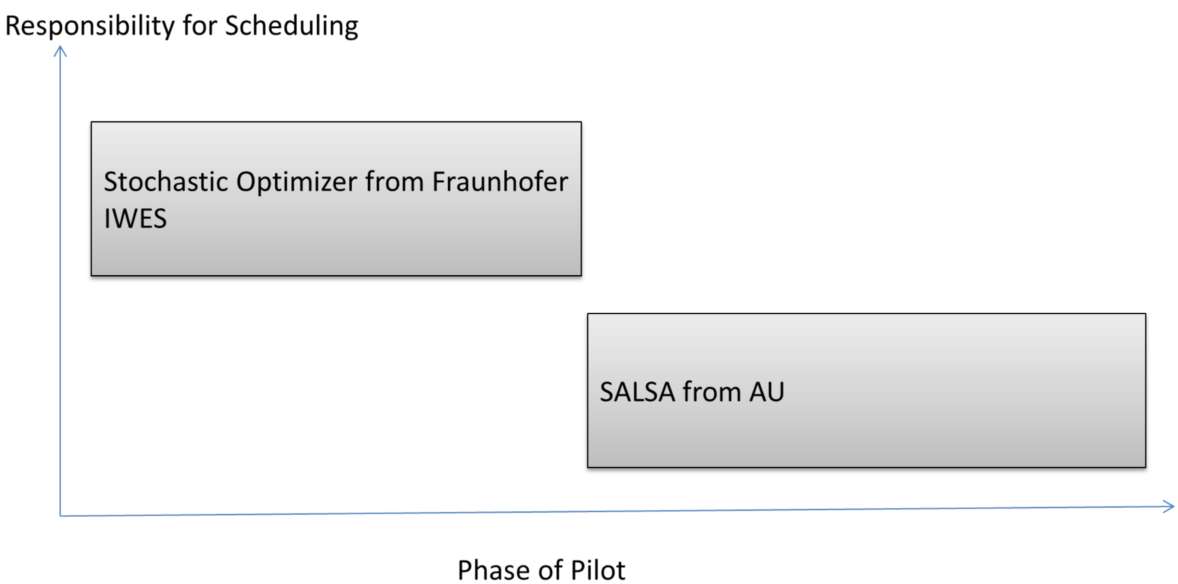


Figure 6: application plan for the two scheduler

Figure 6 shows the application plan for the pilots. At the beginning of the pilot the stochastic optimization will be enabled and provide the schedules for the households. The SALSA will be disabled at that time. At the end of the field test, there should be a trial period for the SALSA. At this time, the stochastic optimization from Fraunhofer will be disabled and the SALSA is responsible for providing the schedules. SALSA can be seen as a possible replacement for the internal running scheduling algorithm from Fraunhofer.

4.1 Fraunhofer: Back-End Servers with OGEMA and Flexibility Forecast

Fraunhofer has established a test household within our building. This test household contains one room which is heated by an electric heating system.

Figure 7 shows the structure of the test household with real hardware. OGEMA is running on the DEVELCO gateway and is connected with a smart plug and a temperature sensor from DEVELCO. These devices are controlled and measured over ZigBee. The necessary measurements are sent to the IWES.vpp and the flexibility forecast. The forecast calculates a probabilistic flexibility forecast, which is used by the stochastic optimization inside of the IWES.vpp. The results are schedules of each household, which are sent to the associated OGEM A instance.

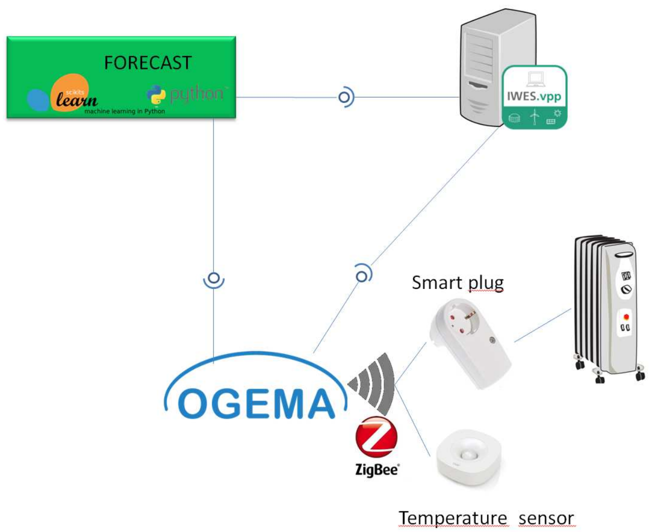


Figure 7: SEMIAH test lab at FRAUNHOFER

With this construction we want to show:

- the integration of the GVPP with OGEMA,
- the integration of OGEMA with the Flexibility Forecast
- and the integration of the Flexibility Forecast with the GVPP

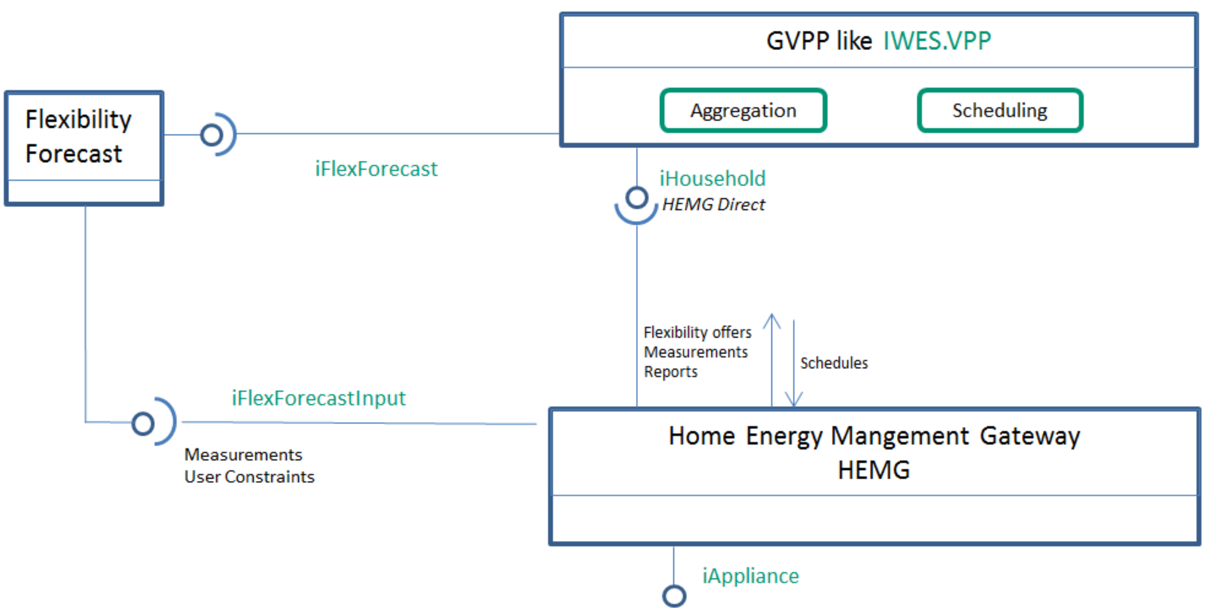


Figure 8: Integration test environment: Extract of SEMIAH system architecture

Figure 8 shows the central architecture of the SEMIAH test lab construction at Fraunhofer. It is a subset of the overall architecture. This kind of structure was chosen to ensure, that we are close enough on the first phase of the pilot.

That the GVPP is integrated with OGEMA is shown in Figure 8. OGEMA send the measured active power from the electric heating system to the GVPP.

The GVPP is also integrated with the flexibility forecast. This is shown in Figure 9. The frontend only visualizes three quantiles from the flexibility forecast. For the stochastic optimization all available quantiles will be used.

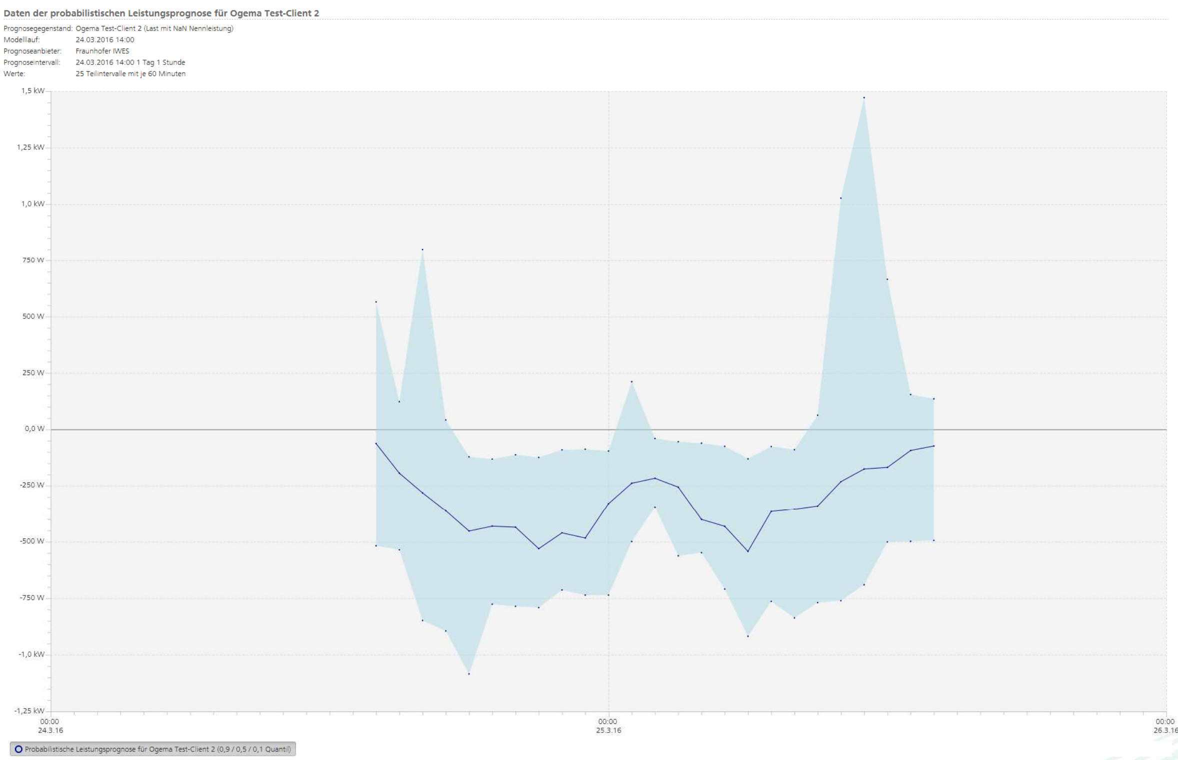


Figure 9: Flexibility Forecast in IWES.VPP

4.2 Fraunhofer: Stochastic Optimization integrated into the Back-End Servers

The stochastic optimization described in deliverable D5.1 is integrated into the IWES.vpp. After the optimization is finished the results can be survey with the frontend of the IWES.vpp. The result is a schedule band for each household and is shown in Figure 10.

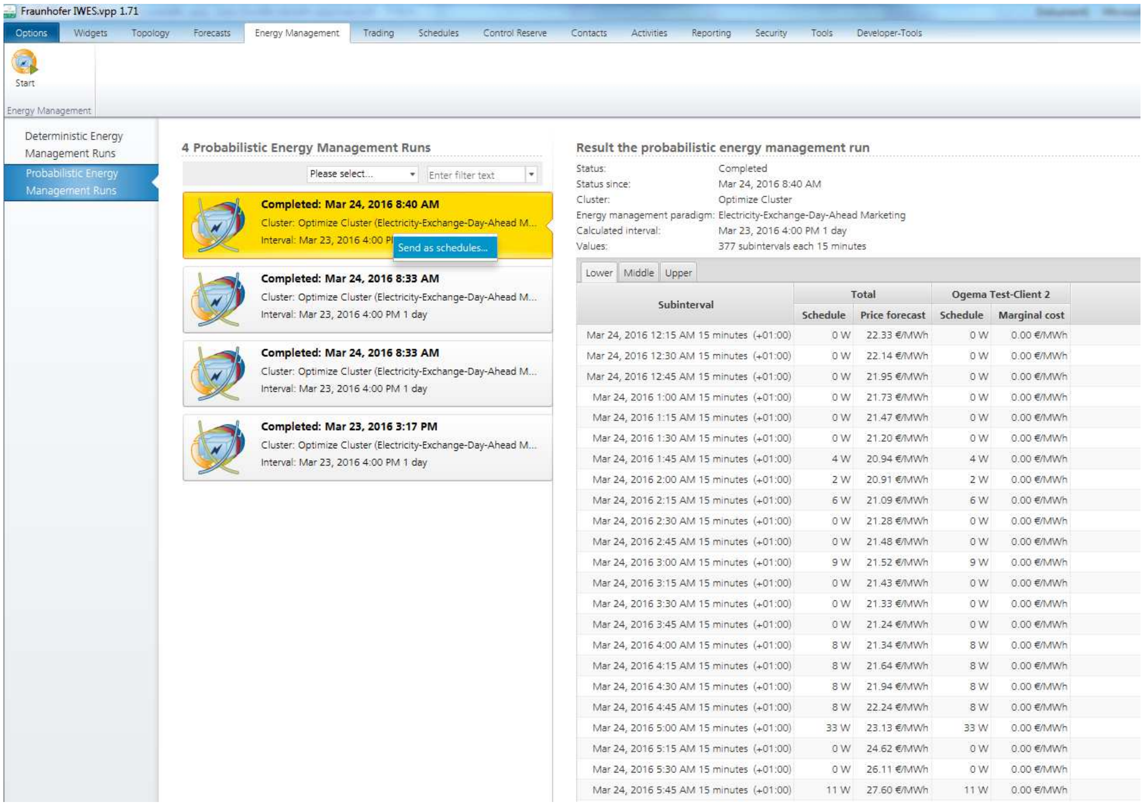


Figure 10: The result of the stochastic optimization

The IWES.vpp provides the possibility to send the schedule band to the OGEMA in the household. This dialog is shown in Figure 11.

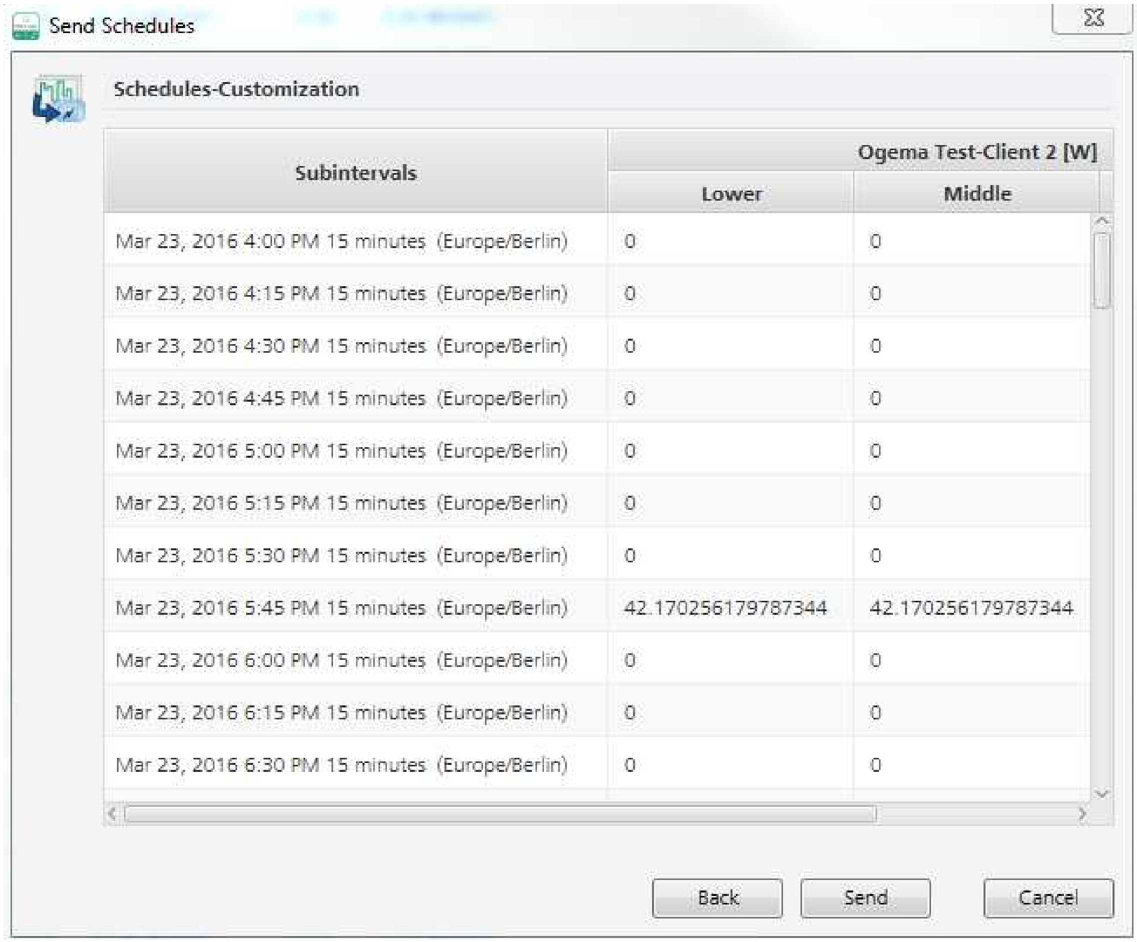


Figure 11: The schedules can be customized before they are provided for OGEMA

Not every stochastic optimization must be provided to OGEMA. To show only the schedules that are sending to OGEMA the frontend of the IWES.VPP can be used for that (Figure 12).

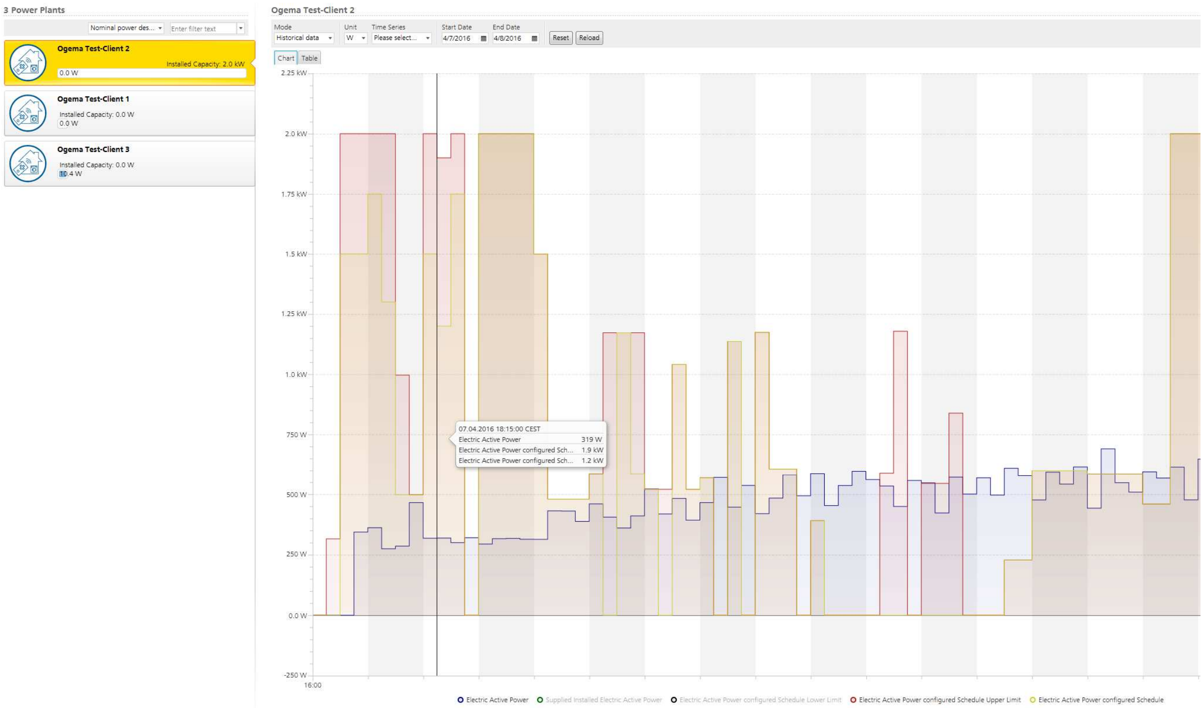


Figure 12: OGEMA move within the schedule band (blue – active power, yellow – probable schedule, red – upper schedule)

4.3 Fraunhofer: Flexibility Forecast integrated with OGEMA

The flexibility has no separate frontend for visualization. To show that the flexibility forecast is integrated with OGEMA the Figure 13 shows an extract of the database of the flexibility forecast. Here we can see two value types, that were transferred from OGEMA connect with the electric heating system. One is the temperature (CONST_T_HEATER_ROOM) and the other one is the active power (CONST_PW_HEATER) of the smart plug, which is connected with the heater.

(4910) {_id : 576a8d440cc28c4507522433}	{ 10 fields }
_id	576a8d440cc28c4507522433
refid	I9272dcc8-6d20-42d2-84cc-6b80e1300e4c
timeStart	2016-06-22T13:01:05.000Z
timeEnd	2016-06-22T13:06:05.000Z
value	298.206
quality	GOOD
unit	Kelvin
storageid	Labor
category	CONST_T_HEATER_ROOM
owner	ogema2
(4911) {_id : 576a8d460cc28c4507522434}	{ 10 fields }
_id	576a8d460cc28c4507522434
refid	I9272dcc8-6d20-42d2-84cc-6b80e1300e4c
timeStart	2016-06-22T13:00:56.000Z
timeEnd	2016-06-22T13:05:56.000Z
value	0.0
quality	GOOD
unit	W
storageid	Labor
category	CONST_PW_HEATER
owner	ogema2

Figure 13: Extract of the database of the flexibility forecast

4.4 AU-Fraunhofer: Integrating SALSA with the Backend-Server

The REST interface proposed for sending the schedule band is finalized. The front-end of the IWES.vpp uses the same interface for sending schedules to OGEMA. Figure 14 shows an overview on how to use the interface with an external tool.

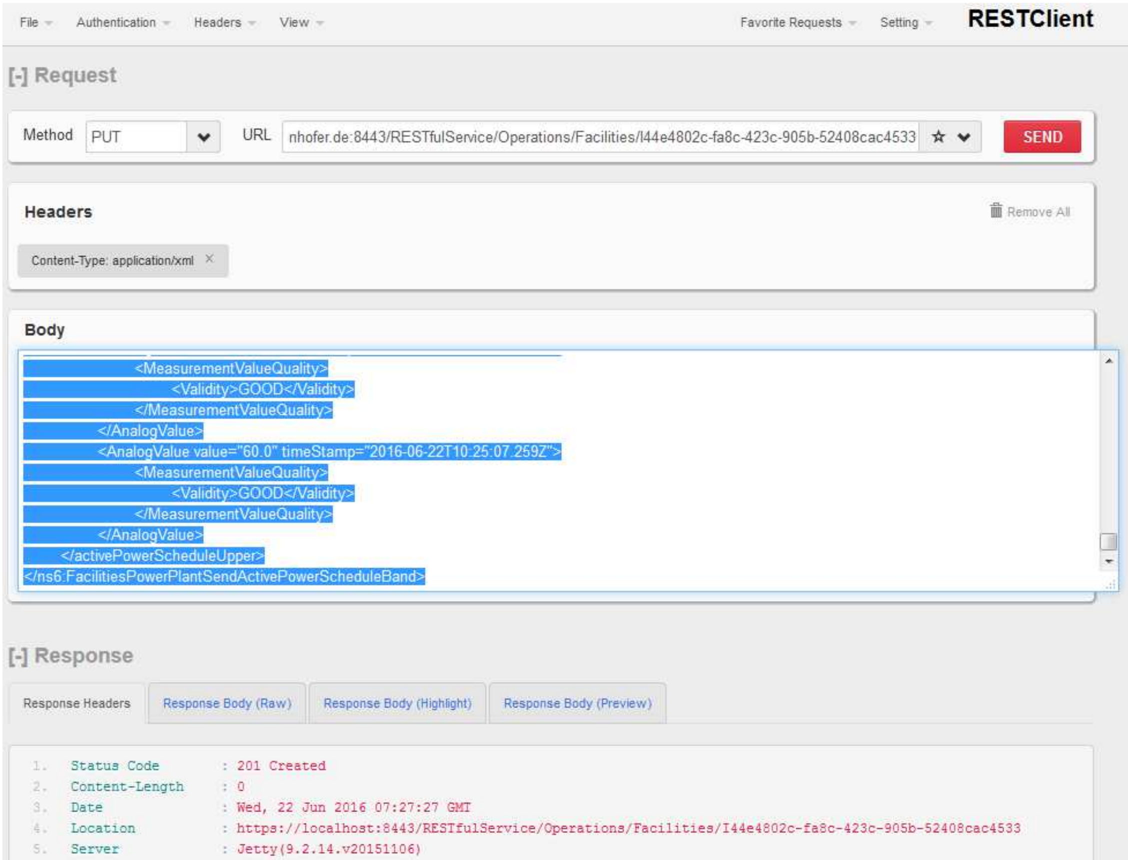


Figure 14: Calling iAlgorithm Interface with Firefox RESTclient plugin: Sending schedules to OGEMA

Figure 15 shows a sequence diagram of how SALSA communicates with Restful API service through iAlgorithm. First, GVPP runs some operations and make households ready to be scheduled. Then, SALSA connects to GVPP through *iAlgorithm* using a Restful API service. It gets each household's information and then, schedules it. Decisions are forwarded back by putting a new XML file in the corresponding household's profile. Finally, GVPP send the schedule band to OGEMA to actuate the appliances. These procedures are done repetitively for all households over time. Since SALSA has been coded and modelled in Matlab, it converts received XML schedule files into Struct, as Figure 16 shows. Then, SALSA updates AnalogValues using Matlab's syntax. Finally, it puts back the modified information, as a new XML file, into the Restful service of the corresponding household.

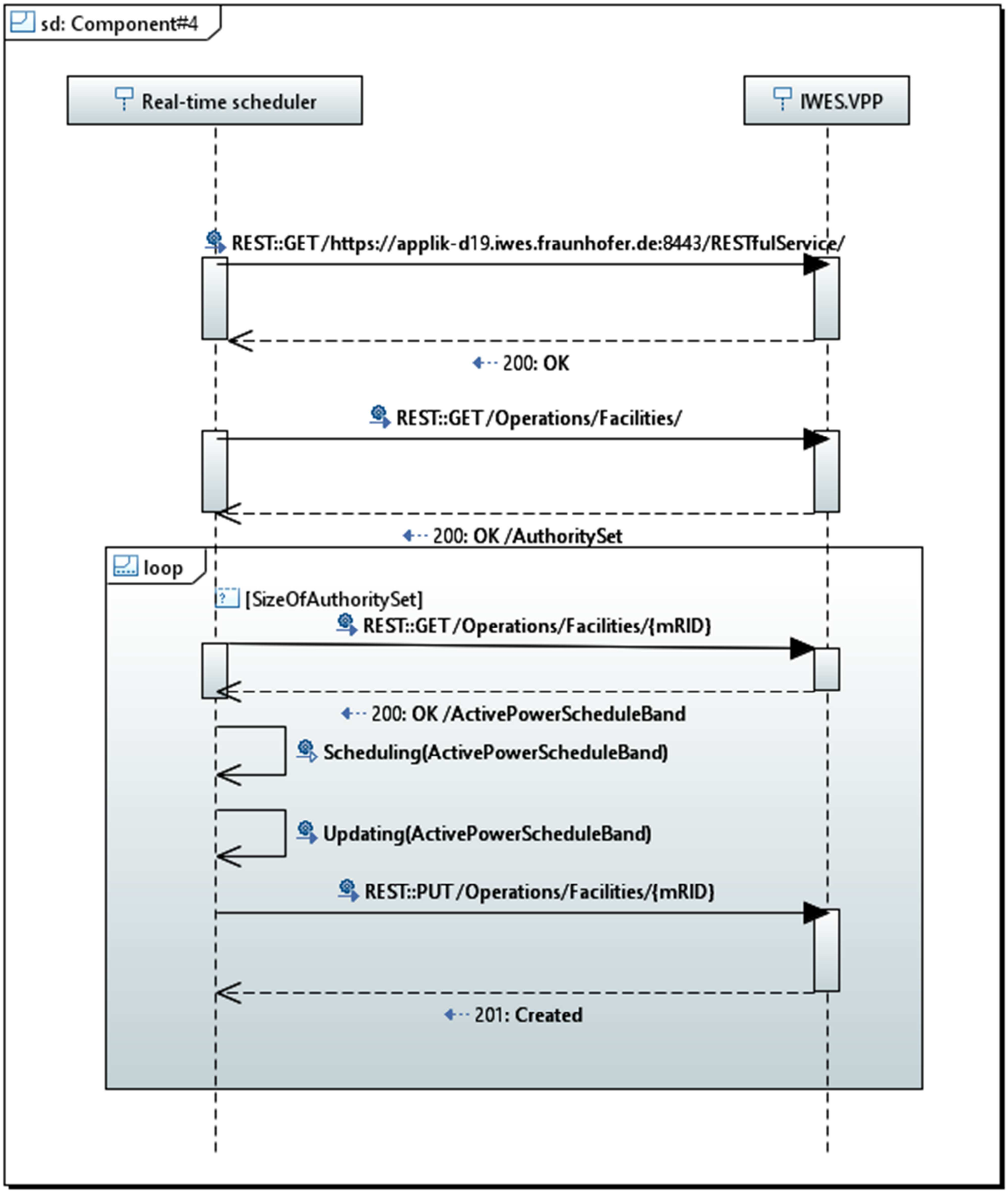


Figure 15: Sequence diagram of interactions between GVPP and SALSA

Workspace | Variables - Output1

Output1 | Output1.activePowerSchedule | Output1.activePowerSchedule.AnalogValue

Output1.activePowerSchedule.AnalogValue

Fields	MeasurementValueQuality	timeStamp	value
1	1x1 struct	'2016-04-07T08:00:00Z'	1.9843
2	1x1 struct	'2016-04-07T08:15:00Z'	0.5371
3	1x1 struct	'2016-04-07T08:30:00Z'	0.0182
4	1x1 struct	'2016-04-07T08:45:00Z'	0
5	1x1 struct	'2016-04-07T09:00:00Z'	0
6	1x1 struct	'2016-04-07T09:15:00Z'	0
7	1x1 struct	'2016-04-07T09:30:00Z'	0
8	1x1 struct	'2016-04-07T09:45:00Z'	0
9	1x1 struct	'2016-04-07T10:00:00Z'	0
10	1x1 struct	'2016-04-07T10:15:00Z'	0
11	1x1 struct	'2016-04-07T10:30:00Z'	0
12	1x1 struct	'2016-04-07T10:45:00Z'	0
13	1x1 struct	'2016-04-07T11:00:00Z'	0
14	1x1 struct	'2016-04-07T11:15:00Z'	0
15	1x1 struct	'2016-04-07T11:30:00Z'	0
16	1x1 struct	'2016-04-07T11:45:00Z'	0
17	1x1 struct	'2016-04-07T12:00:00Z'	0
18	1x1 struct	'2016-04-07T12:15:00Z'	0
19	1x1 struct	'2016-04-07T12:30:00Z'	0
20	1x1 struct	'2016-04-07T12:45:00Z'	0
21	1x1 struct	'2016-04-07T13:00:00Z'	0
22	1x1 struct	'2016-04-07T13:15:00Z'	0
23	1x1 struct	'2016-04-07T13:30:00Z'	0
24	1x1 struct	'2016-04-07T13:45:00Z'	0
25	1x1 struct	'2016-04-07T14:00:00Z'	0
26	1x1 struct	'2016-04-07T14:15:00Z'	0
27	1x1 struct	'2016-04-07T14:30:00Z'	1.5000
28	1x1 struct	'2016-04-07T14:45:00Z'	1.5000
29	1x1 struct	'2016-04-07T15:00:00Z'	1.7500
30	1x1 struct	'2016-04-07T15:15:00Z'	1.3000
31	1x1 struct	'2016-04-07T15:30:00Z'	0.4981
32	1x1 struct	'2016-04-07T15:45:00Z'	0.4981
33	1x1 struct	'2016-04-07T16:00:00Z'	0.4499
34	1x1 struct	'2016-04-07T16:15:00Z'	0.4499
35	1x1 struct	'2016-04-07T16:30:00Z'	0.4499
36	1x1 struct	'2016-04-07T16:45:00Z'	0.5088
37	1x1 struct	'2016-04-07T17:00:00Z'	2
38	1x1 struct	'2016-04-07T17:15:00Z'	2
39	1x1 struct	'2016-04-07T17:30:00Z'	2.0000

Figure 16: AnalogValues (schedules) fetched from GVPP in Matlab's workspace

4.5 AU-Fraunhofer: Integrating the DSO interface with the Backend-Server

Fraunhofer is planning the design of the component from the backend server to provide a possibility to send the Grid Constraints to the IWES.VPP.

The DSO Grid Constraints runs as a web application running as SaaS on an independent server at AU premises. On the one side it serves the DSO user interface e.g., GUI, which can be accessed from a DSO control room or similar and on the other side it serves as an information source for the SEMIAH backend components implementing the iGridConstratins interface.

The running version supports only GET method calls. This allows SEMIAH Intelligence i.e., the SALSA scheduler to fetch the grid constraints for a SEMIAH *collection*. The information is subsequently delivered as an XML document in accordance with the SEMIAH data models described in deliverable D4.2.

In the future, the DSO Grid Constraints will be updated to support also POST/PUT methods. This function is needed to interact with the GVPP (IWES.vpp) component. This integration furthermore requires a coherent mapping between the clustering concept in IWES.vpp and the more generic concept of a collection from the SEMIAH architecture, i.e., deliverable D3.2.

4.6 SEMIAH Back-End Servers with Release and Fault Handling

Most of the Back-end servers is a Feature Delivery by FRAUNHOFER delivered as SaaS. These have been integrated with the OGEMA front-end server as per time:

- IT aggregator infrastructure for smart grid
- Algorithms for intelligent load control and scheduling
- Algorithms for probabilistic load forecasts from households

As the servers are essentially already integrated and verified, SIT (System Integration Test) ensures and manages these deliveries and the remaining back-end servers into packages of servers and do some re-verification of some main flows taking into account the level of already done tests from the Fraunhofer deliveries. D3.1 defines CM (Configuration Management) and validation plan for SEMIAH (that also includes the back-end servers), as Figure 17 shows. Detailing for SW branching, release, maintenance strategy is needed due to changes in the project, since some SW is released as SaaS.

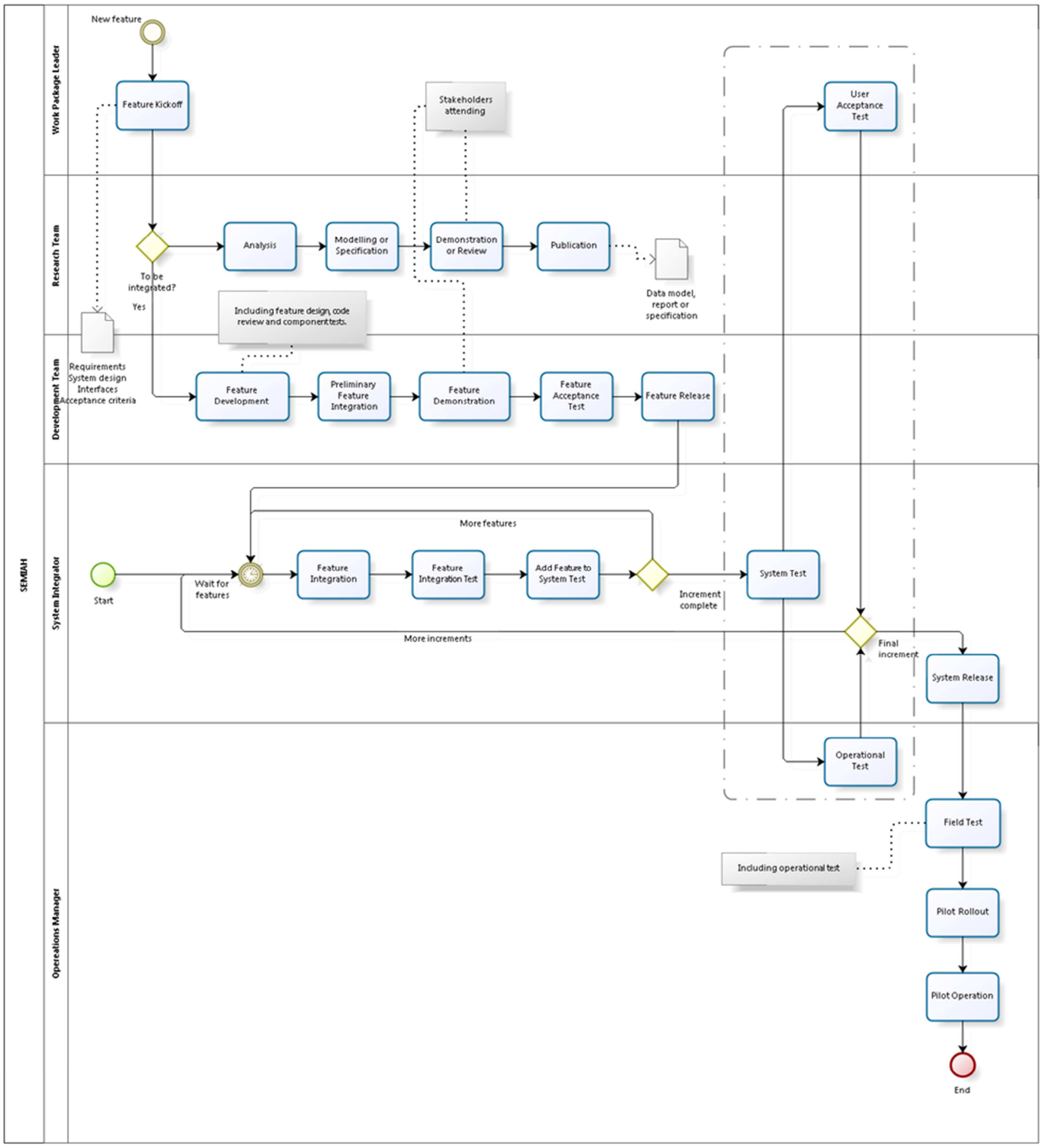


Figure 17: The SEMIAH integration strategy taken from D.3.1

4.6.1 Integration Understanding of SW Branching Strategy

Essentially Git Flow is used as a base for the release handling strategy:

https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

http://nvie.com/posts/a-successful-git-branching-model/

Figure 18 presents SW branching strategy per Integration Stage based on the following items[1]:

- DEV (Development, Software and Feature) is yellow and violet
- SIT is initial green items

[1] http://nvie.com/posts/a-successful-git-branching-model/

- ST is following green items
- Pilot braces are taken out from master or release branches, if needed

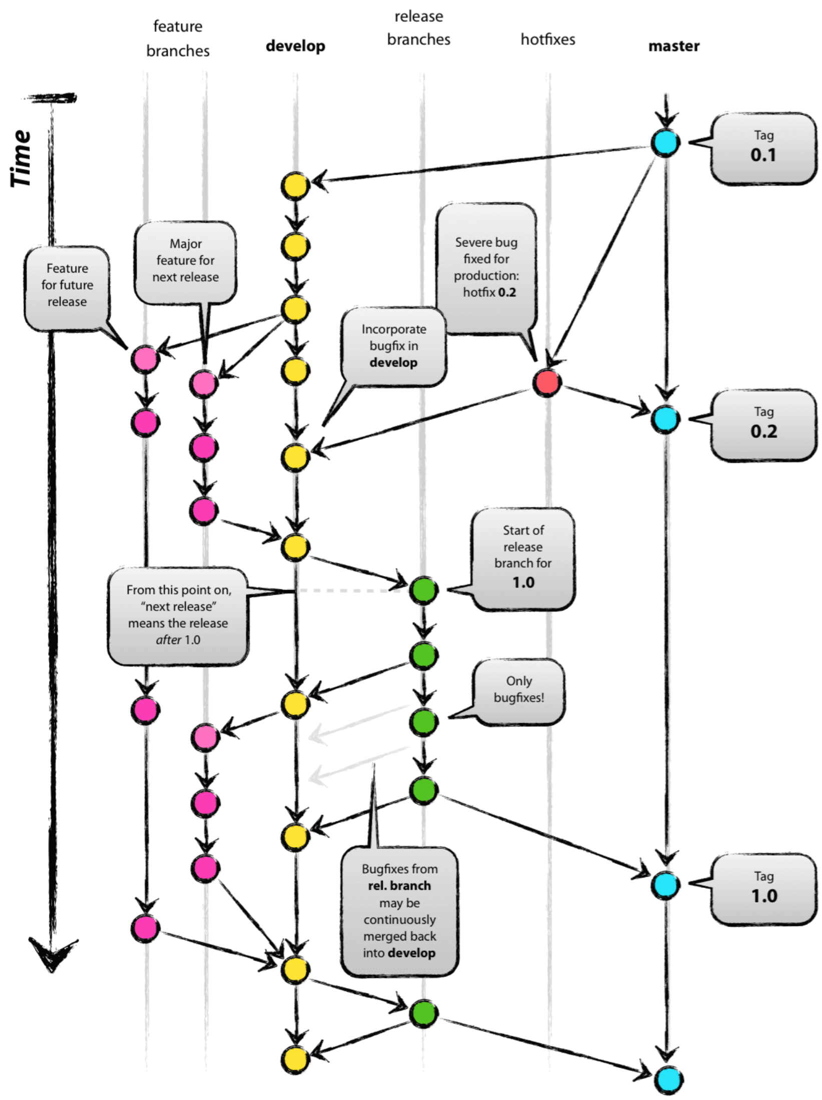


Figure 18: SW branching strategy per Integration Stage

4.6.2 SW Branching Adherence and Ownership

If the branching strategy is not adhered, then the respective site needs to make that fact invisible to cooperating partners and users. The feature team assists on all SW branches of the SW product in question. Branch ownership includes:

- The feature team owns and develops feature, hotfix and master branches.
- SIT team own release branches until release mail is sent to ST.
- ST team own release branches after release mail is received from to SIT.
- Pilot Production team per pilot-project owns the pilot branches (taken from master or from the latest-preferable release-branch named e.g. 'pilot_norway_1' or 'pilot_swiss_1').

All branching and merging are done by feature team members on any and all branches, unless the merge is nontrivial or if delivery is not understood by branch owner.

4.6.3 SW Release Strategy

The needed tools or ways of working to handle a SW release is described in the following subchapters.

4.6.3.1 Release Mail and Web Update

All SW or SaaS to be integrated or taken by another party shall be informed of release e-mails to one of these groups depending on request/delivery minimum 48 hours before release time starts:

- semiah_sit_notification@googlegroups.com
- semiah_st_notification@googlegroups.com
- semiah_pilot_notification@googlegroups.com
- semiah_fd_notification@googlegroups.com

and as cc to:

- semiah_release_notification@googlegroups.com

The "Release upgrade request "email template with email subject is found at http://semiah-wiki.wikispaces.com/.

Note SW release web pages found http://semiah-wiki.wikispaces.com/ are also updated

- The Feature team shall send release mail and update release web page when the feature or fix is ready to be integrated.
- The SIT team shall send release mail and update release web page when the system is ready to be tested.
- The ST team shall send release mail and update release web page when the system is to be integrated.
- The Pilot team shall send release mail and update release page when the Pilot effort is to be terminated.

4.6.3.2 Release Upgrade Time

Any release is preceded and initiated with one (or more) "Release upgrade request". The pre-noticed release is normally starting execution (reboots) at release upgrade time Mondays 10:05 AM CET for DEV and SIT. The pre-noticed release normally starts execution at a release upgrade time Wednesdays 11:05 AM CET for ST and Pilots. Release time for both environments (FD/SIT and ST/Pilots) are always notified with "Release upgrade pending" according to release time notification sent out 24 and 1 hours in advance for release upgrade time.

"Release upgrade completed" for both environments (FD/SIT and ST/Pilots) are sent when the upgrade is completed or rolled-back. The email templates for "Release upgrade pending" and "Release upgrade completed" with email subject is found http://semiah-wiki.wikispaces.com/Feature+deliverables.

4.6.3.3 Release Upgrade Rollback

All partner participants will investigate and track release upgrade and inform if rollback is needed. The rollback will be executed as soon as possible after failing release to upgrade attempt. All servers should have the possibility to be rolled-back 5 versions for pre-updated versions and existing old versions should preferably be kept background available without longer SW unpacking if feasible.

4.6.3.4 Emergency Release Upgrade Handling

On the Pilot (and other) releases, a regime using immediate and rapid and anytime 'Release upgrade request/pending/completed' messages are foreseen. This is however for now left for further studies/elaborations.

4.6.3.5 SW Maintenance Strategy

All faults or SW issues are handled using Bitbucket. Issue handler is located at: https://bitbucket.org/semiah/. This applies for both SaaS and Bitbucket as well as all SW deliveries (like Pilots, etc.) in SEMIAH. The issue tracker in Bitbucket needs to be personally activated first and each partner in SEMIAH must have at least 1 Issue handler responsible per partner.

https://confluence.atlassian.com/bitbucket/enable-an-issue-tracker-223216498.html

https://confluence.atlassian.com/bitbucket/use-the-issue-tracker-221449750.html

4.6.3.6 SW Integration Deployment Strategy

Initially the Bitbucket release is git cloned and installed.

4.7 CM Maintenance Strategy

Higher level requirements and project issues can also be reported with Bitbucket as Issues and should be considered the official handling of any issue in SEMIAH including Pilots and Documentation.

4.8 Test strategy

The test-strategy servers for SEMIAH Back-end at IWES is seen in **Error! Reference source not found.**

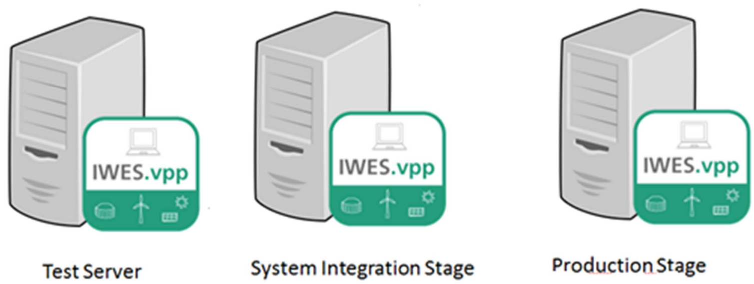


Figure 19: Test-Strategy SEMIAH Back-end server

As the two paradigms for the integration of the SEMIAH back-end system we choose “test driven development” and “continuous delivery”. The first paradigm ensures that every functionality is covered by one or more test. The second paradigm means, that every commit results in a complete build of the software. This build includes on the one site all the component tests and on the other site all of the integration tests, where we test user-story-based the functionality of the whole system. If the test were successful, then the system was continuously delivered to a test server. This server is used to test the front-end of the IWES.vpp with the new version of the back-end. The components of the front-end itself are also tested every commit. After all tests were completed, we can deploy the new version to the “system integration stage” and/or the “production stage”. The management task (checkput, build, test, deploy) is done with the continuous-integration-server based on the open source framework Jenkins. Figure 19shows the involved systems.

4.9 Source Code Repository

SW is handled according to D3.1 delivery including the CM strategy.

4.9.1 Bitbucket

All SW (unless agreed with main project to be e.g. SaaS) is stored using Bitbucket in: https://bitbucket.org/semiah/

SW available elsewhere is virtually (indirectly) made available from.

https://bitbucket.org/semiah/

SaaS SW is stored and maintained as agreed with the main project.

5 Back-End Test Configuration

Test configurations for back-end testing tried at SIT are:

- OGEMA with external SQUID. The focus is an external GUI demonstrator and a REST GUI ControlState update
- OGEMA without SQUID with SQUID-traffic-simulator with VPP(/FlexF/DSO/Intelligence)
- FE VPP GUI tool alone towards VPP
- Installation and Verification of GUI of DSO Grid constraints when stubbed
- OGEMA with external SQUID with-VPP(/FlexF/DSO/Intelligence)
- OGEMA on SQUID with VPP(/FlexF/DSO/Intelligence)

5.1 Minimum Room Setup with Real HW SmartPlug Controlled Heater and Temperature Sensor, External SQUID and OGEMA

This test setup is using the OGEMA environment to send data to the back-end of the system. This is a minimalistic real SEMIAH HW setup consisting of

- One Develco Zigbee TemperatureSensor
- One Develco Zigbee SmartPlug
- One Develco SQUID-HEMG
- One HEMG -OGEMA on Windows (alternative Linux)
- One HEMG-SmartAmm Driver
- One HEMG Zigbee Driver
- One HEMG -OGEMA FXML (example GUI) and REST GUI Config GUI for HEMG configuration

5.1.1 Software Module Architecture & Configuration

Back-end module architecture is shown in Figure 20.

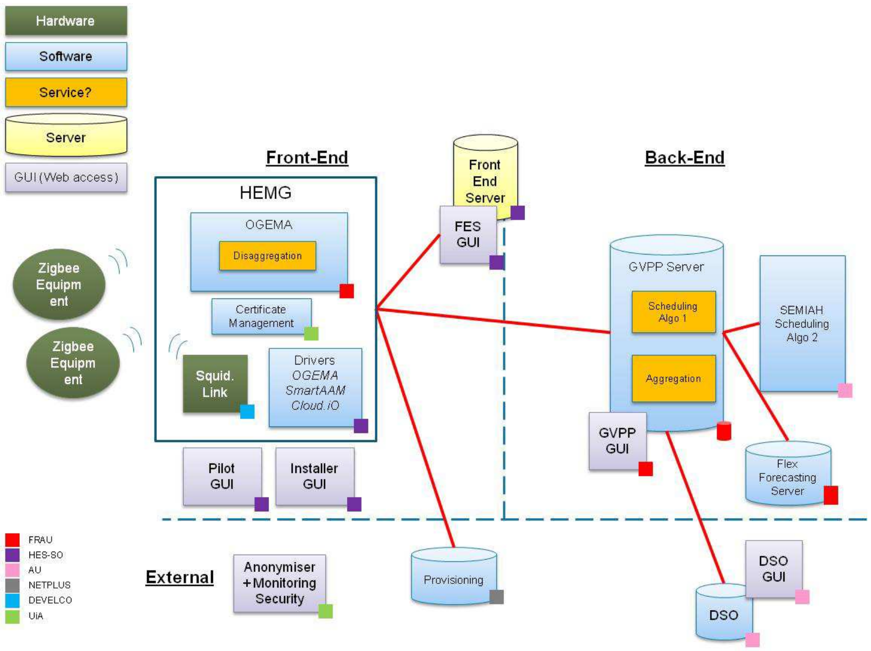


Figure 20: architecture with Back-end focus

5.1.2 Infrastructure – Hosting Environment

Develco SW is hosted on the delivered Develco Starter kit HW. The SQUID, Tempsensor and the SmartPlug are default part of a Develco starter kit. Develco SmartAmm Developer Utility Tool is hosted on a Windows 7 PC and is contained in the StarterKit zipped delivery. The Temperature Sensor and SmartPlug are registered and configured to report according to guideline videos in the Develco StarterKit zip.

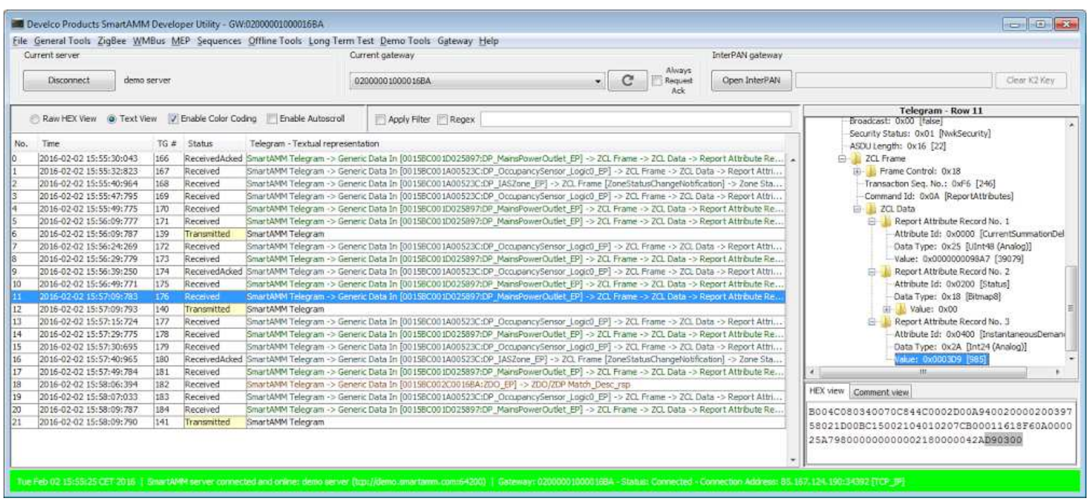


Figure 21: Develco Products SmartAMM Developer Utility GUI tool used to add and configure operation of e.g. Zigbee devices

HEMG -OGEMA with drivers are running on a local PC with initially Windows 7 and Java1.8 with SW fetched from

https://bitbucket.org/semiah/semiah-ogema

https://bitbucket.org/semiah/zigbee-library-java

https://bitbucket.org/semiah/smartamm-library-java

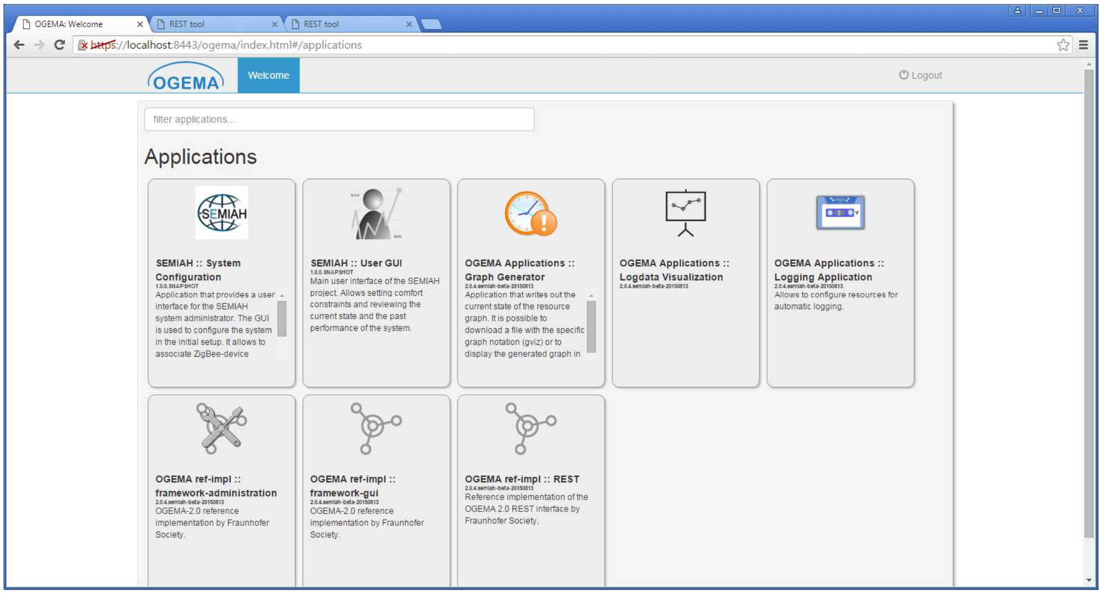


Figure 22: OGEMA initial GUI page

HEMG -OGEMA FXML (example GUI) and REST GUI Config GUI for HEMG configuration. FXML GUI Example is a compiled program that runs GUI and exports XML setting to OGEMA defining the configuration (given as an input to real GUI development).

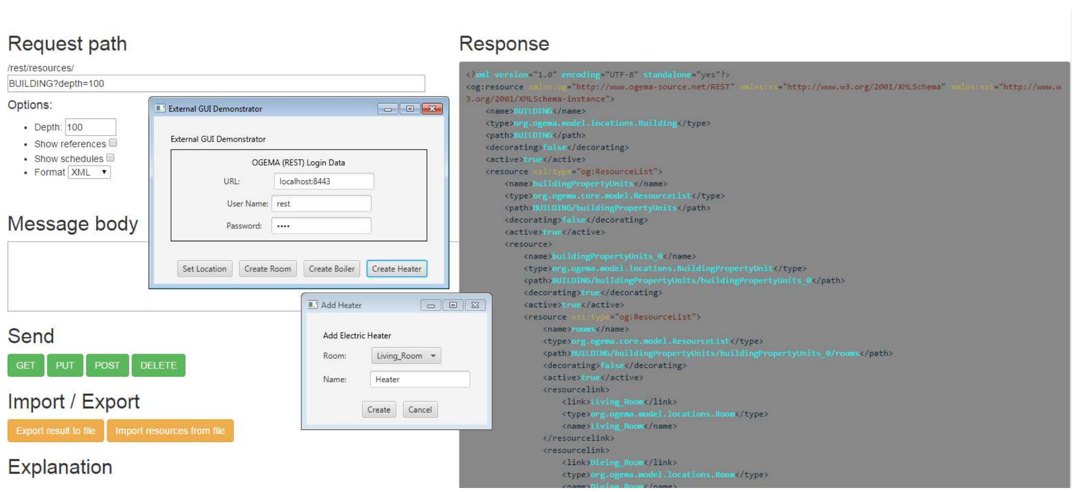


Figure 23: OGEMA REST tool GUI page used to check and update settings provided by 'External GUI Demonstrator'

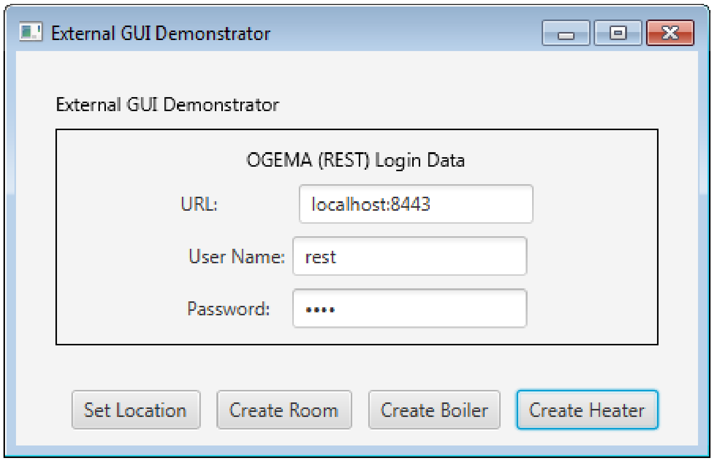


Figure 24: username and password in 'External GUI Demonstrator'

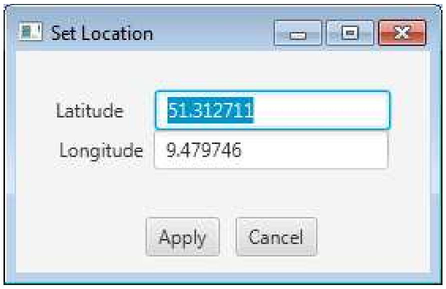
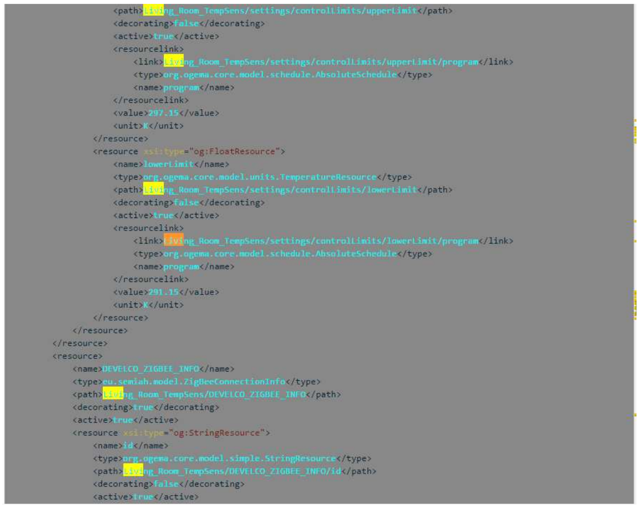


Figure 25: Latitude and Longitude for Household

Figure 26: Adding an Electric Heater

The settings can be read and updated using e.g. OGEMA ref implementation REST




```xml
                <path>Living_Room_TempSens/settings/controlLimits/upperLimit</path>
                <decorating>false</decorating>
                <active>true</active>
                <resourcelink>
                    <link>Living_Room_TempSens/settings/controlLimits/upperLimit/program</link>
                    <type>org.ogema.core.model.schedule.AbsoluteSchedule</type>
                    <name>program</name>
                </resourcelink>
                <value>297.15</value>
                <unit>K</unit>
            </resource>
            <resource xsi:type="og:FloatResource">
                <name>lowerLimit</name>
                <type>org.ogema.core.model.units.TemperatureResource</type>
                <path>Living_Room_TempSens/settings/controlLimits/lowerLimit</path>
                <decorating>false</decorating>
                <active>true</active>
                <resourcelink>
                    <link>Living_Room_TempSens/settings/controlLimits/lowerLimit/program</link>
                    <type>org.ogema.core.model.schedule.AbsoluteSchedule</type>
                    <name>program</name>
                </resourcelink>
                <value>291.15</value>
                <unit>K</unit>
            </resource>
        </resource>
    </resource>
    <resource>
        <name>DEVELCO_ZIGBEE_INFO</name>
        <type>eu.semiah.model.ZigBeeConnectionInfo</type>
        <path>Living_Room_TempSens/DEVELCO_ZIGBEE_INFO</path>
        <decorating>true</decorating>
        <active>true</active>
        <resource xsi:type="og:StringResource">
            <name>id</name>
            <type>org.ogema.core.model.simple.StringResource</type>
            <path>Living_Room_TempSens/DEVELCO_ZIGBEE_INFO/id</path>
            <decorating>false</decorating>
            <active>true</active>
```


Figure 27: 'Living_Room' in XML definition using REST tool

The XML setting needs to be read and updated to be active. These settings are then again linked to the real defined HW that is being used.

5.1.3 Environment Configuration

OGEMA with drivers are installed on local PC with E.g. Window Java 1.8, using Maven. FXML GUI is compiled and run using E.g. Sping with Maven using README file description per item to define the room with temperature setting. The Develco SmartAmm Developer Utility is used to learn/register in the Temperature Sensor and the SmartPlug and how these shall report temperature and Watt usage.

5.1.4 Software Build

StarterKit SW (SQUID, Sensor, SmartPlug) is used as delivered. README files per item is followed for OGEMA, Zigbee Driver and SmartAmm Driver to build SW. GUI example is built and run e.g., using Spring with Maven. The REST GUI is used from within OGEMA. The front-end Software is delivered by Fraunhofer and an executable (EXE file) and somewhere in the front end GUI both for VPP front-end and back-end release labels exist. A development/SIT applik19 and an ST/pilot applik20 SW exist on the Fraunhofer SaaS servers.

5.2 Front-End IWES.VPP Test Tool from Fraunhofer IWES.VPP

This set setup accesses the IWES.VPP back-end directly, but does not have a Flexible Forecast interface. It is possible to send 'Time Series' load for a household and load limits can be set. Considering that if a household (possibly a Prosumer - ProduserConsumer) is a powerplant that consumes or delivers power, then a direct mapping for levels and households can be embedded.

5.2.1 Infrastructure – Hosting Environment

The test environment runs on a local Windows machine. The IWES.vpp client can be downloaded from:

https://applik-d19iwes.fraunhofer.de:8443/vppcore/home.xhtml

(user and password can be provided to Devoteam by jarle.einar.qvigstad@devoteam.com). Integration environment configuration is done according to Annex D.

5.2.2 Software Build

The Software is delivered by Fraunhofer and is available in the front end GUI. Both release labels: a development applik19 and a pilot applik20 exist SW on the servers. These labels should as a service and should be frozen for applik21 to maybe applik22 and release notes should be sent when updates are done in a controlled manner.

6 System Integration and System Test tasks

6.1 System Integration (SI)

6.1.1 Principle

As explained in the System and Integration Test plan and specifications, the System integrator's task will consist in integrating the different parts of the back-end system and re-verify the main operation and new and changed operation iteratively. It means verifying that any new implemented and accepted feature, interfaces or APIs can be added to the central software repository, can be configured and built where the main operations run/works.

6.1.2 Entry Criteria

- Back-end system documentation ready
- Working back-end system IT infrastructure environment
    - All features, components, interfaces implemented, feature-tested and accepted
    - Failure rate: No fatal and/or very serious failure are reported from feature testing
- Used and stubbed interfaces ready

6.1.3 Exit Criteria

- Software release successfully built and installed/upgraded
- Failure rate: No fatal and/or very serious failure are reported
- Release notes delivered

6.1.4 Tasks

- Build and package software modules according to the Integration & verification plan schedule
- Write a release note about the content of the package and open issues if any
- Report any failure in the issue tracking system
- Inform the Test team (Test leader) a new Software build is ready for verification

6.1.5 Planning

All components and modules will be integrated.

6.2 System Test (ST) Tasks

6.2.1 Principle

Once the integration is finished, the verification of the back-end integrated parts can start within the same iteration. This verification will be split into 2 phases: system integration testing and system testing.

6.2.2 Entry Criteria

- Detailed test instructions detailing the framework ready for back-end
- Integration and system tests reviewed and approved, including the selected feature tests proposed by the development teams.
- Software builds release build with no fatal or very serious failures.
- Release notes for the software build to verify
- Test environment ready (test tool, test machines)

6.2.3 Exit Criteria

- All Integration executed. Pass rate: 100% (Pass rate of executable Test Cases)
- All system tests executed. Pass rate: 50% (Pass rate of executable Test Cases)
- Failure rate: No fatal and/or very serious failures are reported
- Test completion report delivered for each phase

6.2.4 Tasks

- Prepare the test environment and test scripts
- Execute tests and document detailed tests according to the integration & verification plan schedule
- Report any issue in the issue tracking system
- Report test results

6.2.5 Planning

All components, modules will be integrated.

7 Conclusions

Aggregator infrastructure integration and verification of the SEMIAH project provided the server infrastructure for both the system integration stage and for the production stage in the back-end system of SEMIAH. These servers were integrated and then defined as a back-end software release. The back-end software release, based on SaaS release labels and bitbucket commit labels, was tested prior to labels was set but just slightly after the back-end software release was defined, due to test setup problems and lack of time. A list of test cases relevant for a back-end software release is found in Annex part.

8 References

[1] SEMIAH DoW
https://dms-prext.fraunhofer.de/livelink/livelink.exe?func=ll&objaction=overviewversion&objid=3862362&vernum=1

[2] SEMIAH-WP3-D3.1-Verification_and_Validation_Plan.docx

[3] SEMIAH-WP3-D3.2-System_Requirements_and_Functional_Specifications.docx
https://dms-prext.fraunhofer.de/livelink/livelink.exe/overview/4024558

[4] SEMIAH-WP6-D6.1- System and Integration Test Specification.docx

[5] SEMIAH-WP5-D5.4- Back-end System Release

9 Annex A - Test Cases

Extraction from D6.1 Annex A Integration Test Cases [4].

9.1 Integration Test Cases – Test of APIs/Interfaces

Table 1 Test Cases from D.6.1 Annex A Integration Test Cases

System Under Test	Interface	Test Cases	Expected result	Verified NotStarted/ Passed/ Passed-Partly/ NotExec/ NotApplic/ Postponed
	iFlexForecast	Verify the ability of the GVPP to receive forecasts	The GVPP is able to accept and use received forecast information	**Passed**
		Verify the ability of the GVPP to receive measurements	The GVPP is able to accept and use received measurements information	**Passed**
		Verify the ability of the GVPP to receive constraints	The GVPP is able to accept and use received constraints information	**NotStarted**
		Verify the ability of the GVPP to receive electricity load aggregation	The GVPP is able to accept and use received aggregated electricity load information	**NotStarted**
		Verify the ability of the GVPP to receive load scheduling	The GVPP is able to accept and use received load scheduling information	**Partly**
	iHousehold	Verify the ability of the GVPP to receive Flexibility offers from the	The GVPP is able to accept and use received Flexibility offers from the household appliances through the HEMG	**NotApplic**

		household appliances through the HEMG		
		Verify the ability of the GVPP to receive Measurements from the household appliances through the HEMG	The GVPP is able to accept and use received measurements information from the household appliances through the HEMG	**Partly**
		Verify the ability of the GVPP to receive Reports from the household appliances through the HEMG	The GVPP is able to accept and use received reports from the household appliances through the HEMG	**NotApplic**
		Verify the ability of the GVPP to provide Schedules to the household appliances through the HEMG	The schedules are provided according to specification	**NotApplic**
	iHouseholdCollection	Verify the ability of the GVPP to receive Flexibility offers for a collection of e.g., households and/or appliances through the HEMG	The GVPP is able to accept and use received Flexibility offers from collection of e.g., households and/or appliances through the HEMG	**Passed**
		Verify the ability of the GVPP to receive Measuremen	The GVPP is able to accept and use received measurements information from collection of e.g., households and/or appliances through the HEMG	**Partly**

		ts for a collection of e.g., households and/or appliances through the HEMG		
		Verify the ability of the GVPP to receive Reports for a collection of e.g., households and/or appliances through the HEMG	The GVPP is able to accept and use received reports from collection of e.g., households and/or appliances through the HEMG	**NotStarted**
		Verify the ability of the GVPP to provide Schedules for a collection of e.g., households and/or appliances through the HEMG	The schedules are provided according to specification	**Passed**
	iSemiahOp	Verify the ability of the GVPP to allow operation and maintenance of the SEMIAH system over the iSemiahOp interface	The SEMIAH system can be operated and maintained through the provided interface	**NotStarted**
	iVppOp	Verify the ability of the GVPP to allow operation and maintenance of the IWES.vpp	The IWES.vpp components planned for used in the SEMIAH pilot can be operated and maintained	**Passed**

		components planned for use in the SEMIAH pilot over the iVppOp interface		
SEMIAH Algorithms	iAlgorithms (consumer)	Verify the ability of the SEMIAH Algorithms to provide flexibility schedules to the GVPP	1. The Algorithm is able to accept and use received information to produce new schedules	**NotStarted**
			2. The schedules are provided according to specification	**NotStarted**

10 Annex B - Test Cases

Extraction from the D.6.1 Annex B System Test Cases

Table 2 Test Cases from D.6.1 Annex B System Test Cases

Main Actors	Test Cases	Expected result	Verified NotStarted/ Passed/ Passed-Partly/ NotExec/ NotApplic/ Postponed
Electricity energy supplier	Schedule a global power profile move in the consumption pattern of my customers	I can minimize the acquisition cost of energy	**NotStarted**
		I can adapt my global power profile to my intermittent generation	**NotStarted**
		I can minimize the balance energy for my balance group	**NotStarted**
		I can successfully respond to call control reserve activation	**NotStarted**
	Schedule a global power profile move in the generation pattern of my customers	I can minimize the acquisition cost of energy	**NotStarted**
		I can adapt my global power profile to my intermittent generation	**NotStarted**
		I can minimize the balance energy for my balance group	**NotStarted**
		I can successfully respond to call control reserve activation	**NotStarted**

	Schedule a global power profile move in the storage pattern of my customers	I can minimize the acquisition cost of energy	**NotStarted**
		I can adapt my global power profile to my intermittent generation	**NotStarted**
		I can minimize the balance energy for my balance group	**NotStarted**
		I can successfully respond to call control reserve activation	**NotStarted**
	Manage the capacity reserve for power profile moves on my customers' premises	I can increase / decrease the margin for energy purchase / sales operations	**NotStarted**
		I can be sure to dispose of a big enough margin to clear the balance energy within a calculation period	**NotStarted**
		I can reliably offer bids for control reserve	**NotStarted**
	Be informed on the success / failure of my global power profile moves	I have a feedback on my power profile move requests	**NotStarted**
	Be informed on power profile move operations requested by DSOs on my customers' premises	I can take actions to mitigate the effect of these operations	**NotStarted**
	Integrate the flexibility service into my market operation	The flexibility service can be used in a way similar to other market instruments.	**NotStarted**
	Package a flexibility product for my clients (communication, rewarding scheme, contract...)	I can increase the loyalty of my current clients and acquire new ones.	**NotStarted**
	Verify the ability to request a demand reduction on a day when supplies are too low	This is a general user story. The details can come later. The request for reduction (towards the prosumers) must come as late as possible, based on the latest predictions	**NotStarted**
Prosumer	Allow external entities to automatically exploit the flexibility of my processes	My electricity bill is reduced.	**NotStarted**
Virtual power plant owner	Verify the ability, as a virtual power plant owner, to trade the flexible energy capacity from on the market	It shall be possible, as a virtual power plant owner, to trade the flexible energy capacity from on the market	**NotStarted**

11 Annex C - Test Cases

Extraction from the D.6.1 Annex C User Acceptance Test Cases

Table 3 Test Cases from D.6.1 Annex C User Acceptance Test Cases

Test Cases	Verified NotStarted/ Passed/ Passed-Partly/ NotExec/ NotApplic/ Postponed
Verify that the project has developed a novel Information and Communication Technology (ICT) infrastructure for the implementation of Demand Response (DR) in households.	**NotStarted**
Verify that the DR infrastructure enables shifting of energy consumption to off-peak periods.	**NotStarted**
Verify that the DR infrastructure enables shifting of energy consumption to periods with high generation of electricity from Renewable Energy Sources (RES).	**NotStarted**
Verify that ICT framework is open.	**NotStarted**
Verify that the project has developed a centralized system for DR service provisioning.	**NotStarted**
Verify that the DR system is based on electricity consumption	**NotStarted**
Verify that the DR system supports aggregation.	**NotStarted**
Verify that the DR system supports forecasting.	**NotStarted**
Verify that the DR system supports scheduling.	**NotStarted**

12 Annex D - Environment Integration Configuration

From a SEMIAH Integration point of view the Fraunhofer VPP delivery with the front-end-client is evaluated. This tool is released as defined in D5.4 Back-end System Release [5]. Firstly, the how-to operate the setup and then find out what this setup can be used for with regards to integration and verification. The rest of this annex will just show a series of figure showing setup and configuration and usage of the tool. After installing and starting the tool, the Connector in Figure 28appears. Click 'Connect'.

Figure 28: FRAUNHOFER IWES.vpp Connector selecting profile at connection time

The password for the tool needs to be given and 'OK' is then pressed.

Figure 29: Providing password

Then select 'Topology'

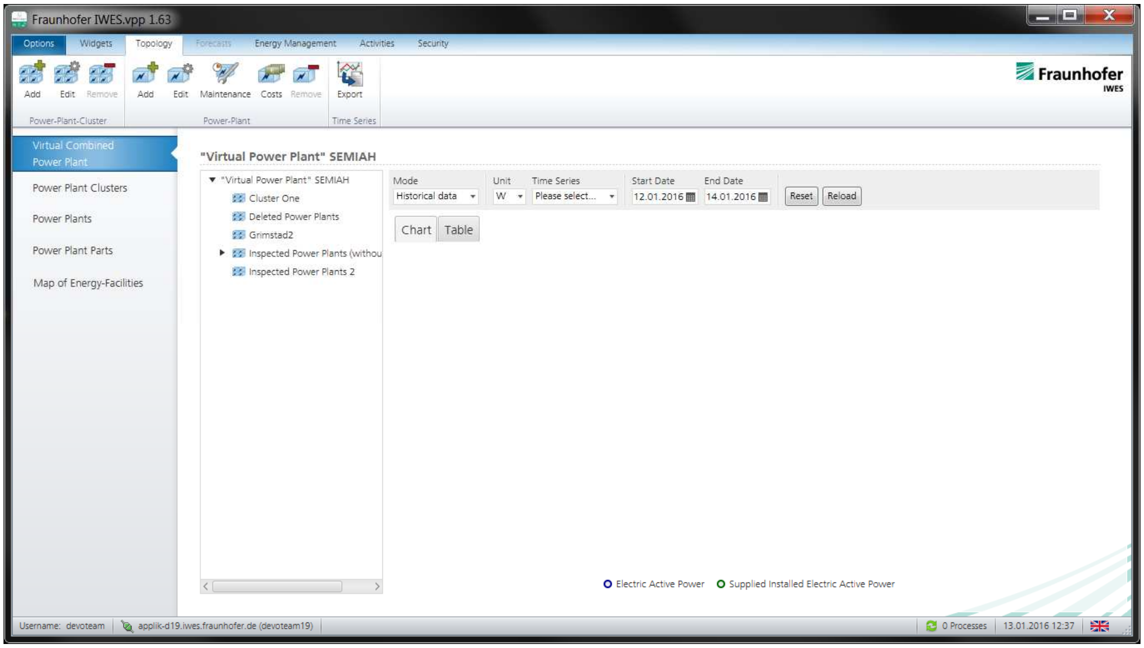


Figure 30: Initial page after connecting

Click Power Plants like in Figure 31

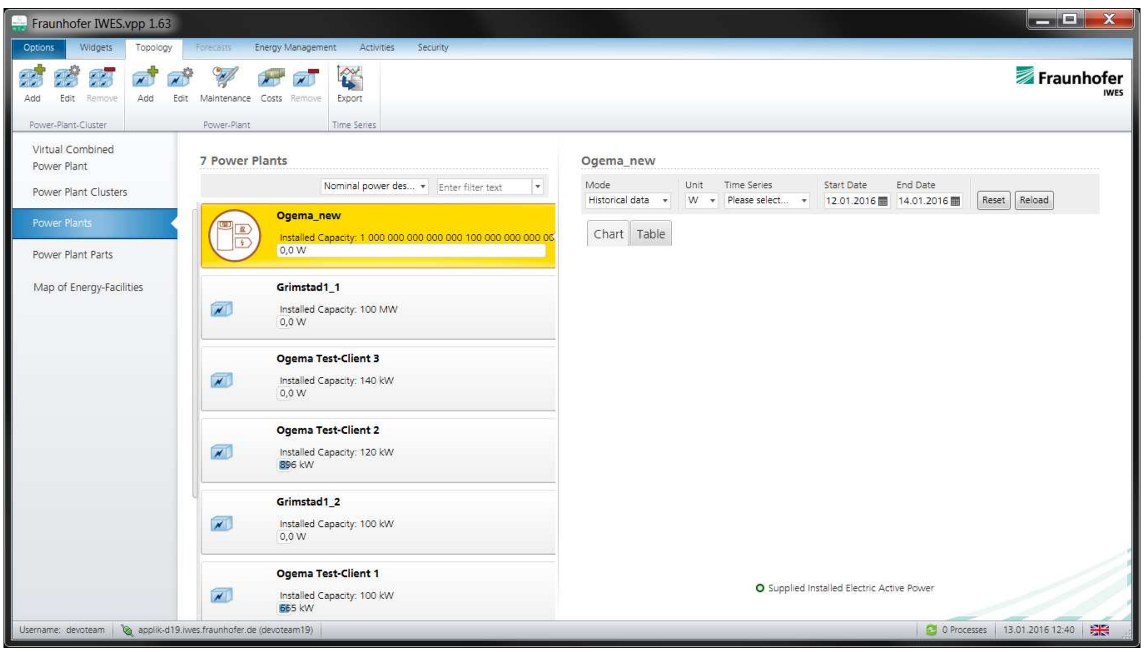


Figure 31: Topology and Power Plants (households) view

Add a new Power-Plant (or household) as shown in Figure 32.

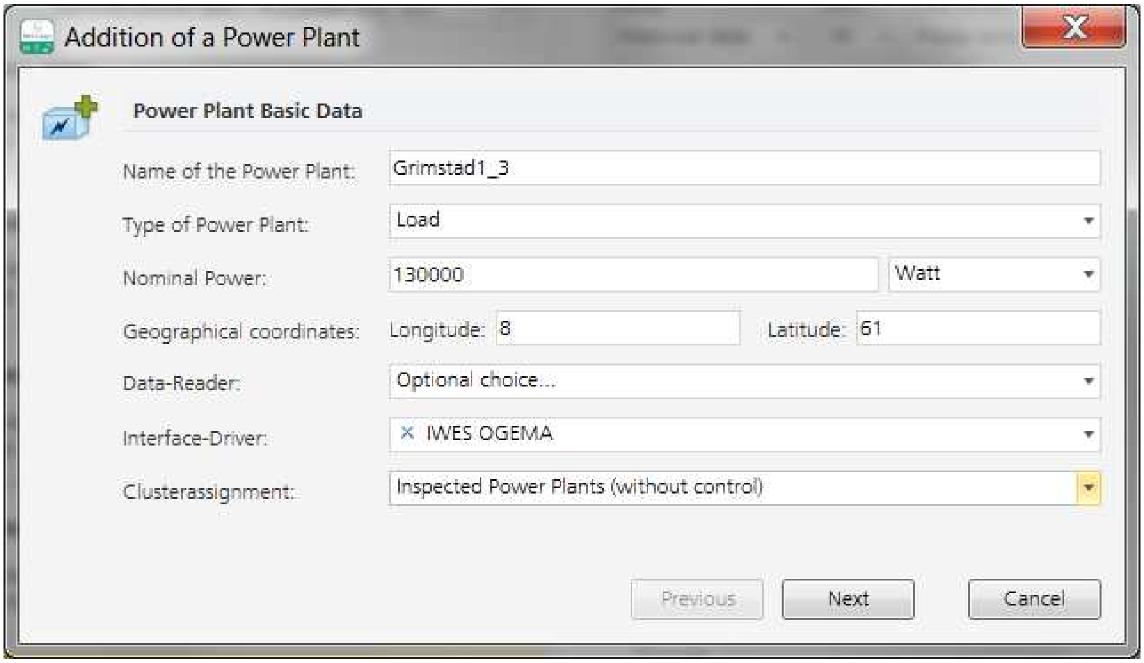


Figure 32: Adding a new Household

Defining the household Username and household Password that is normally used by OGEMA to access VPP can be seen in Figure 33.

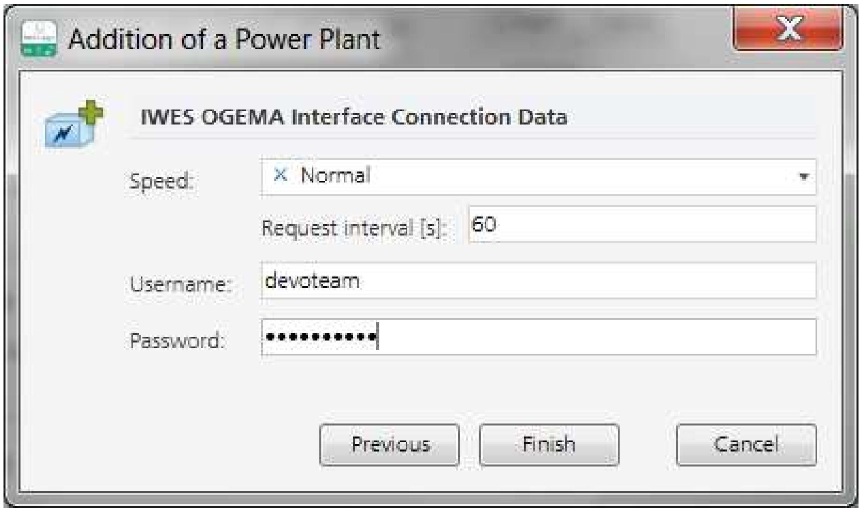


Figure 33: the OGEMA user and password related to the household

Notice in Figure 34 that the new household is defined.

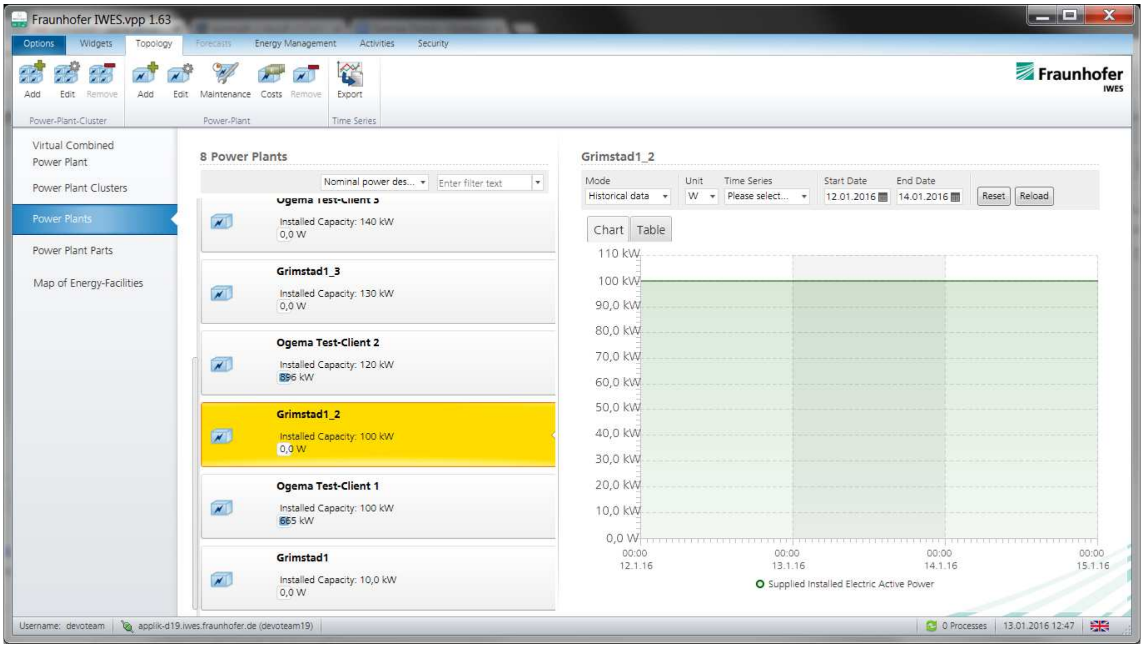


Figure 34: Updated list of Powerplants and households

Click the new defined household and notice the 'Nominal Power' defined matches the curve in Figure. Click Export Time Series -> Select File export of time series.

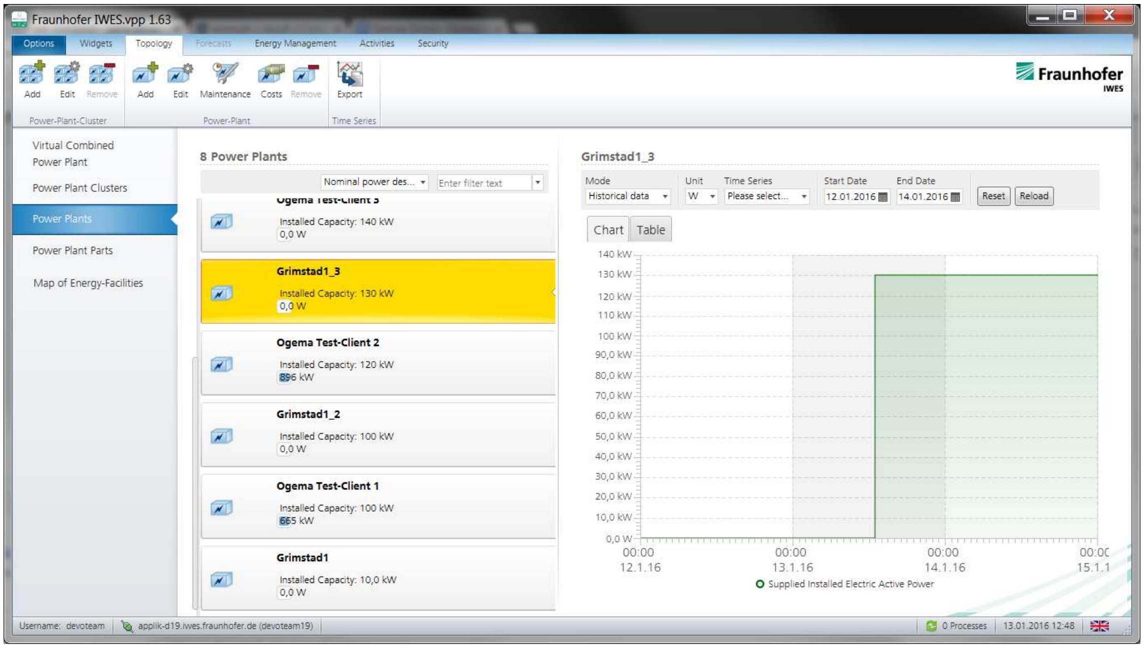


Figure 35: Showing here the new household

Notice new GUI window like Figure 36

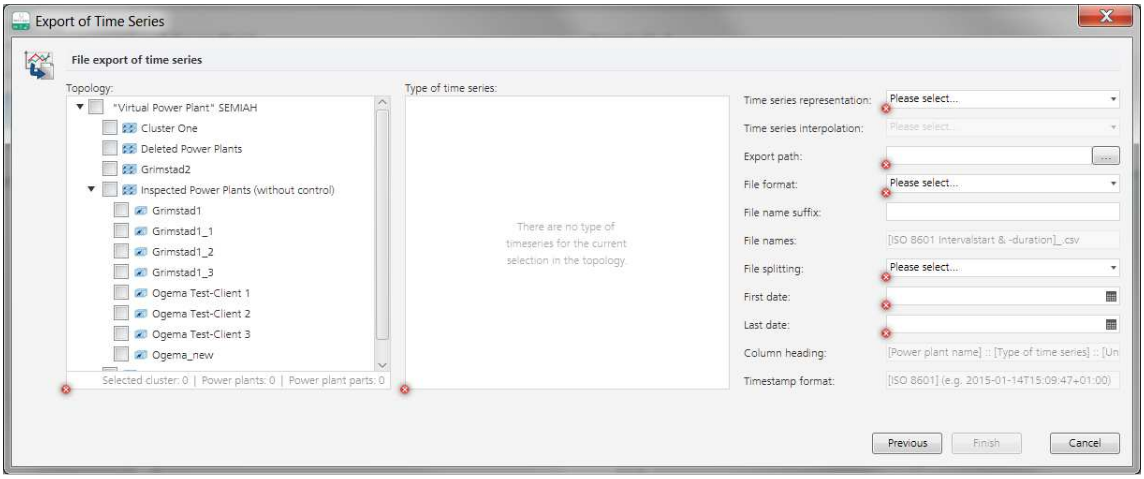


Figure 36: pressing Export of Time series

Fill in data likeFigure 37.

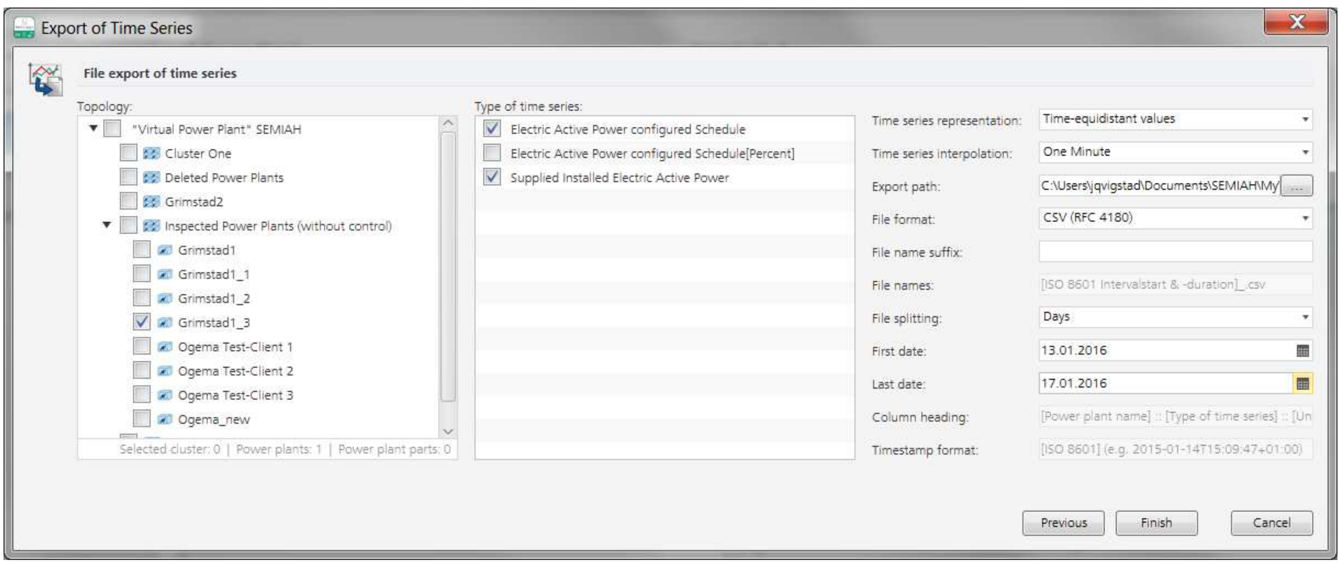


Figure 37: Selecting the household and setting export time series attributes

Create and select a directory where time series data is to be stored like shown in Figure 38.

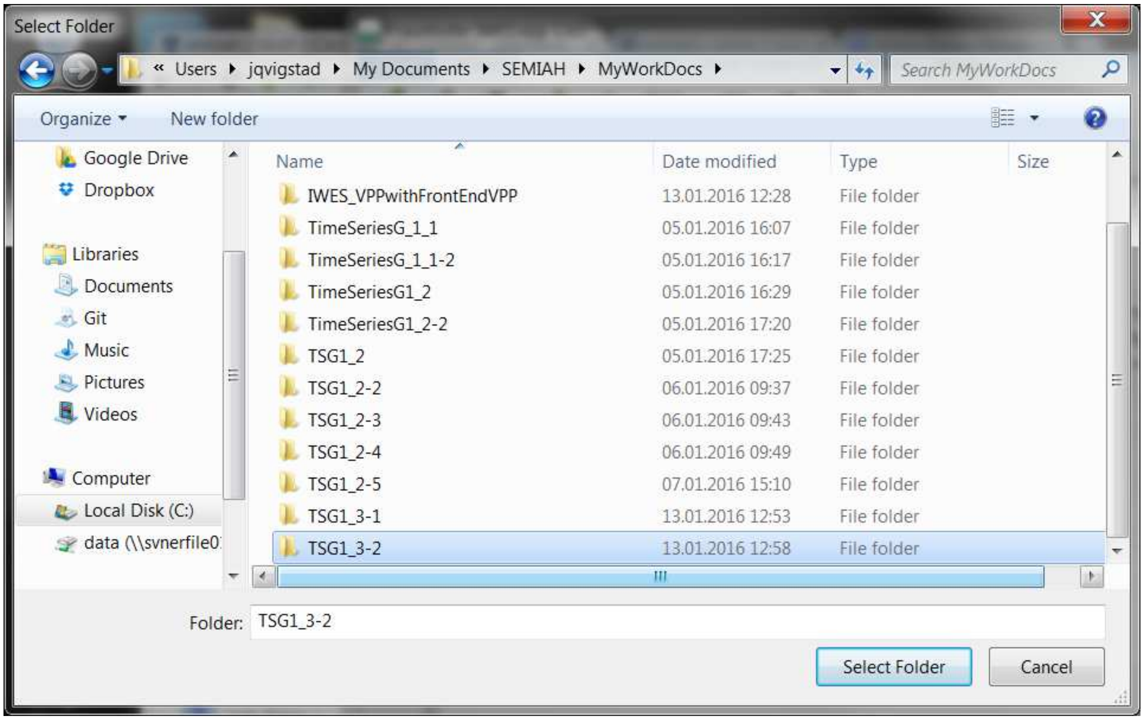


Figure 38: Selecting the path where the household timer series data is to be stored in a directory

Finally, export by pressing Finish button. Notice that export of time series occurs like shown in Figure 39.

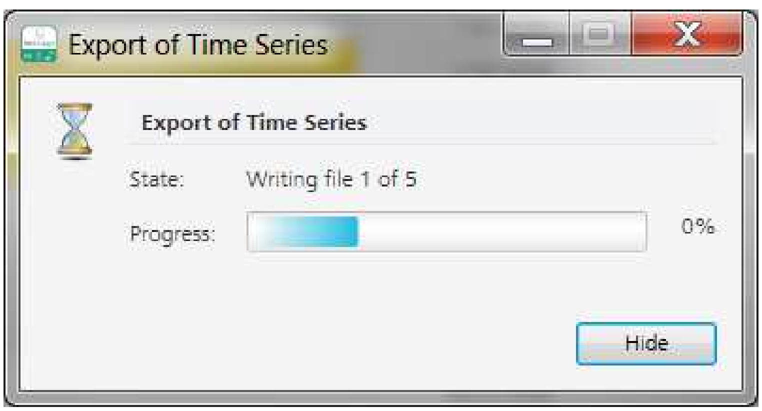


Figure 39: Writing of XML(or csv) data files in the defined directory

Notice that some data has been exported to e.g. csv files in a path as shown inFigure 38.

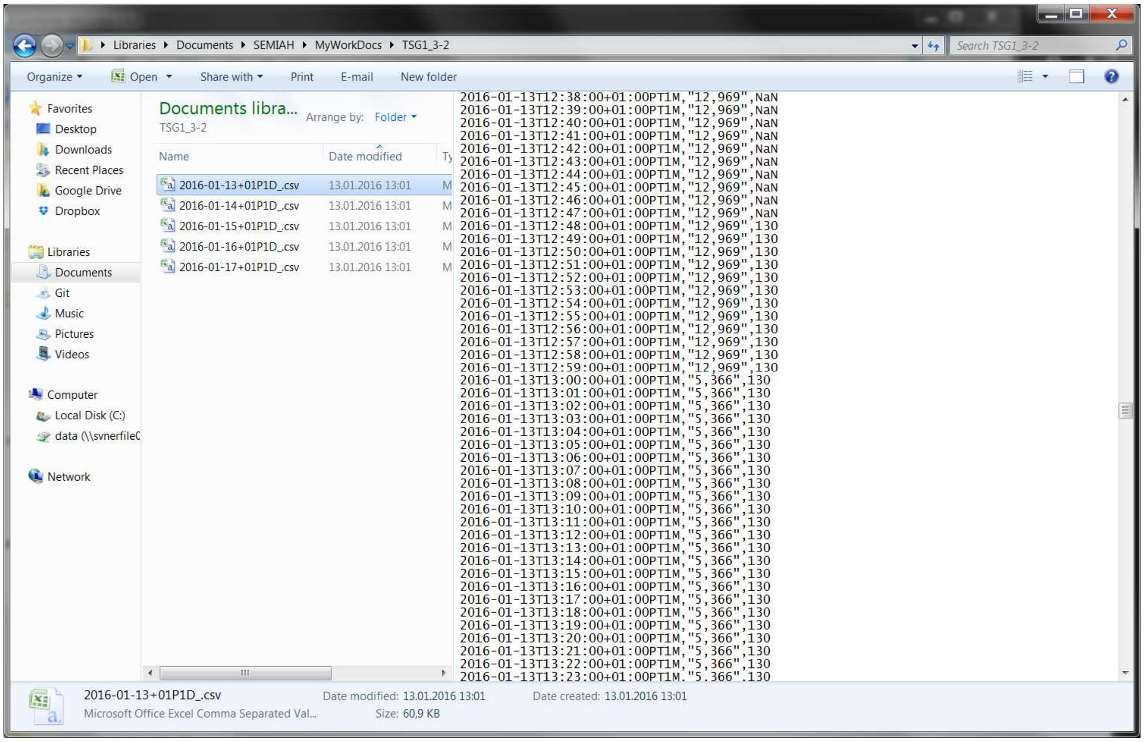


Figure 40: Contents for csv file

Click out and then into the created power plant and then select a time series in Figure 41.

->Select view settings for the 'Time Series' exported data. Then 'Electric Active Power configured Schedule.

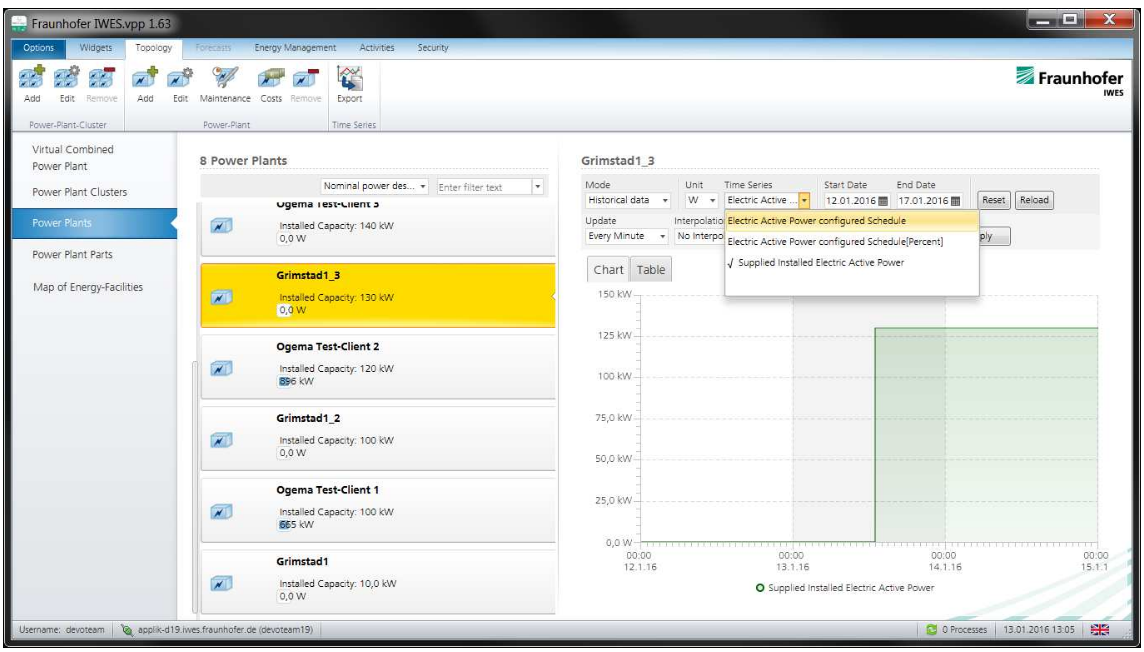


Figure 41: Supplied Installed Electric Active Power curve

Notice that the time series data for the household is drawn and displayed like shown in Figure 42.

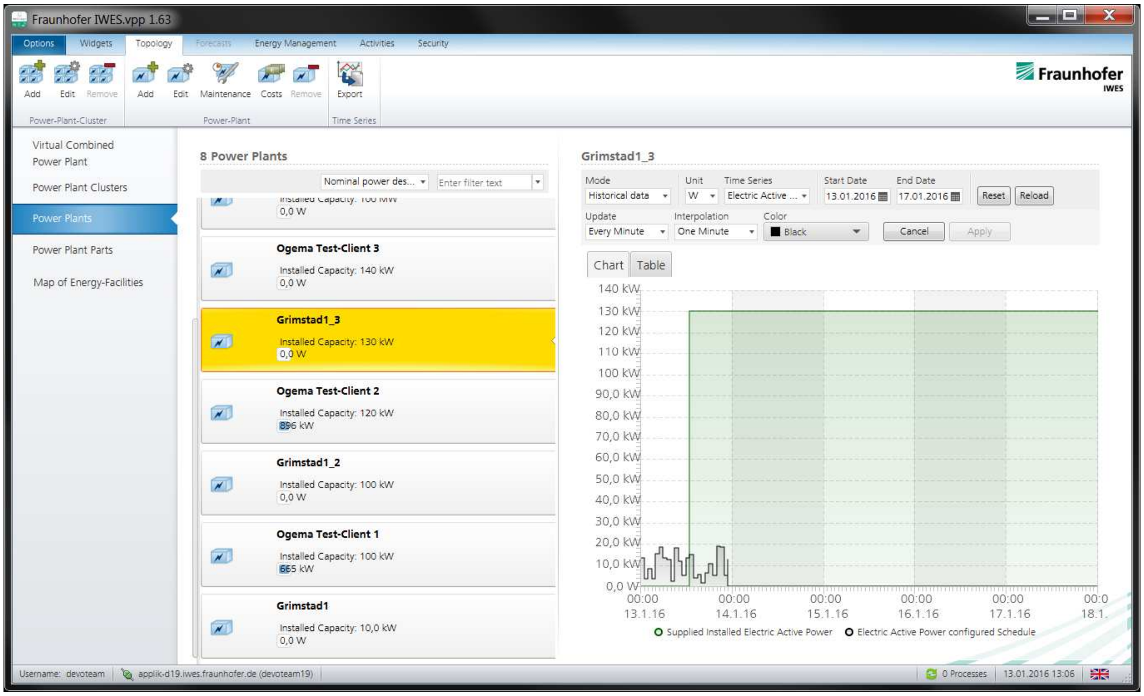


Figure 42: Time series data from the XML curve

Ensure that 'Time Series' is marked with a hook called 'Electric Active Power configured Schedule' as shown in Figure 43.

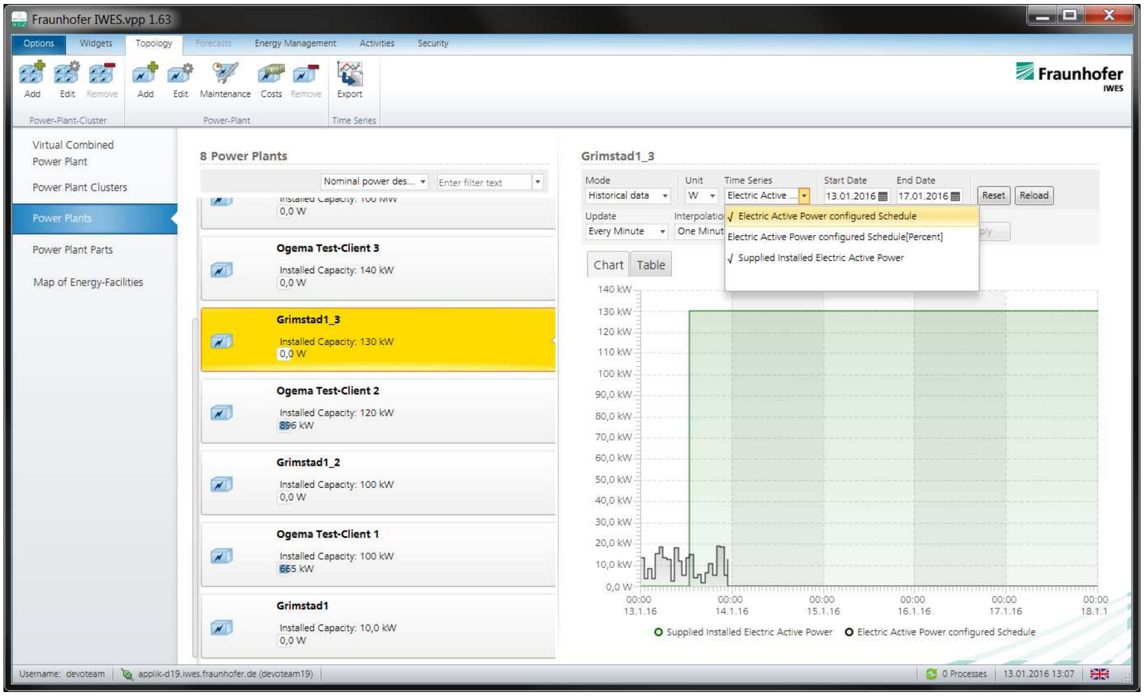


Figure 43: Active Power configured Schedule also selected for visualization

There is also a direct restful interface

https://applik-d19.iwes.fraunhofer.de:8443/RESTfulService/
https://applik-d20.iwes.fraunhofer.de:8443/RESTfulService/

available to login in Figure 44.

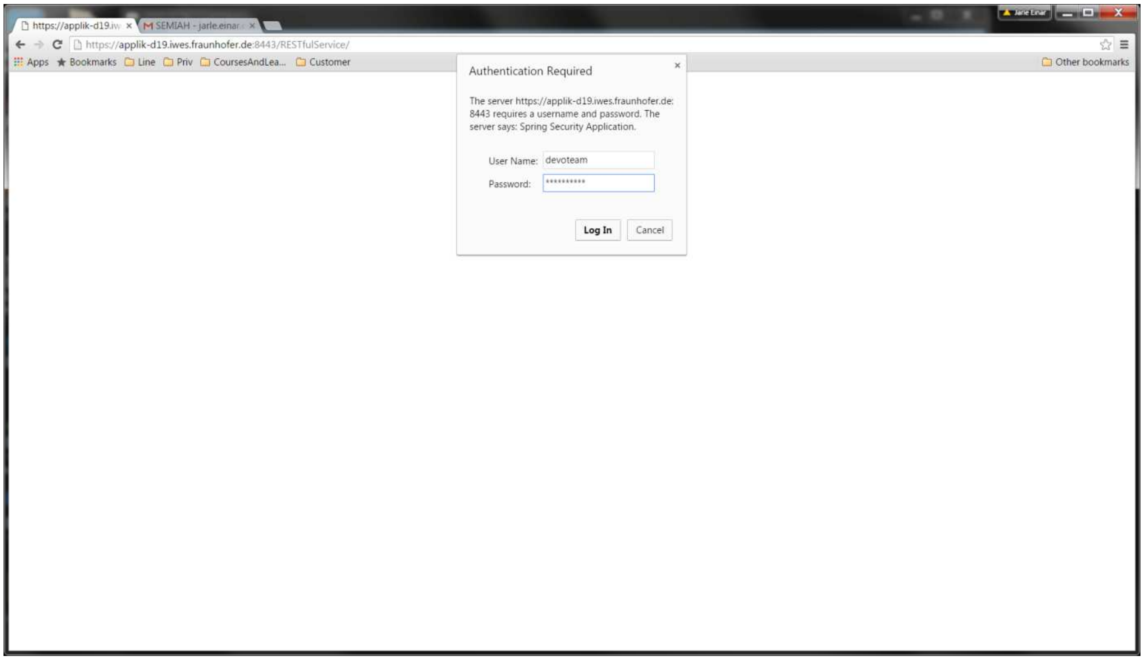


Figure 44: RESTfulService with login security

The RESTful Service of VPP can be browsed like in Figure 45.

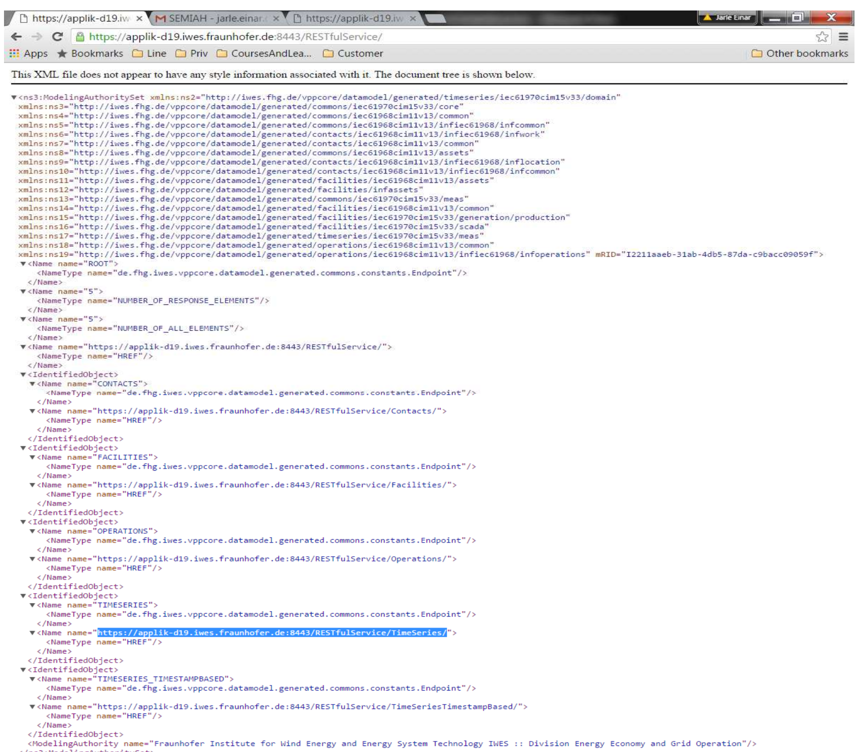


Figure 45: Initial page for RESTfulService