Front-end OGEMA gateway	
Document ID:	SEMIAH-WP4-D4.1-v2.22
Document version:	2.22
Document status:	Final
Dissemination level:	PU
Deliverable number:	D4.1
Deliverable title:	Font-end OGEMA gateway
WP number:	4
Lead beneficiary:	8 (HES-SO)
Main author(s):	D. Gabioud (HES-SO), G. Basso (HES-SO), Y. Maret (HES-SO), J. Ringelstein (IWES),Benoit Cosendey (Netplus), Poul Møller Eriksen (DEVELCO)
Nature of deliverable:	P
Delivery date from Annex 1:	M20
Actual delivery date:	2016-02-04 (M24)
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

The SEMIAH infrastructure for Demand Response in households is made up of a front-end subsystem and of a back-end subsystem. The front-end subsystem handles individual households whereas the back-end subsystem aggregates households for grid and market services. This document presents the front-end subsystem designed and developed in the frame of Tasks T4.1 and T4.2.

The front-end subsystem is made up of the Home Energy Management Gateway, which is based on the OGEMA “operating system” and of the front-end server implementing database, provisioning and web services. Two generations of the front-end subsystem are presented: the field test front-end subsystem, which is already in operation to gather experience with real households, and the full front-end subsystem that will host the distributed part of the SEMIAH intelligence. The full front-end system is running in a laboratory environment.

Abbreviations

API	Application Programming Interface
D	Deliverable
EC	European Commission
CIM	Common Information Model
DNS	Domain Name Service
DSO	Distribution System Operators
DoW	Description of Work
GUI	Graphical User Interface
GVPP	Generic Virtual Power Plant
HEMG	Home Energy Management Gateway
ICT	Information and Communication Technology
IoT	Internet of Things
ISP	Internet Service Provider
SSL	Secure Socket Layer
WP	Work Package

Contents

List of Figures

List of Tables

1 Introduction

1.1 SEMIAH architecture reminder

The SEMIAH infrastructure has been divided in a front-end part and in a back-end part: the front-end part encompasses all elements related to individual households, whereas the back-end part relates to the aggregation of households and to interfaces to system operation and markets.

The SEMIAH architecture is presented in document D3.2 – System Requirements and Functional Specifications. The Figure 7 of this document, which is copied in Figure 1 below, presents a layered model for SEMIAH.

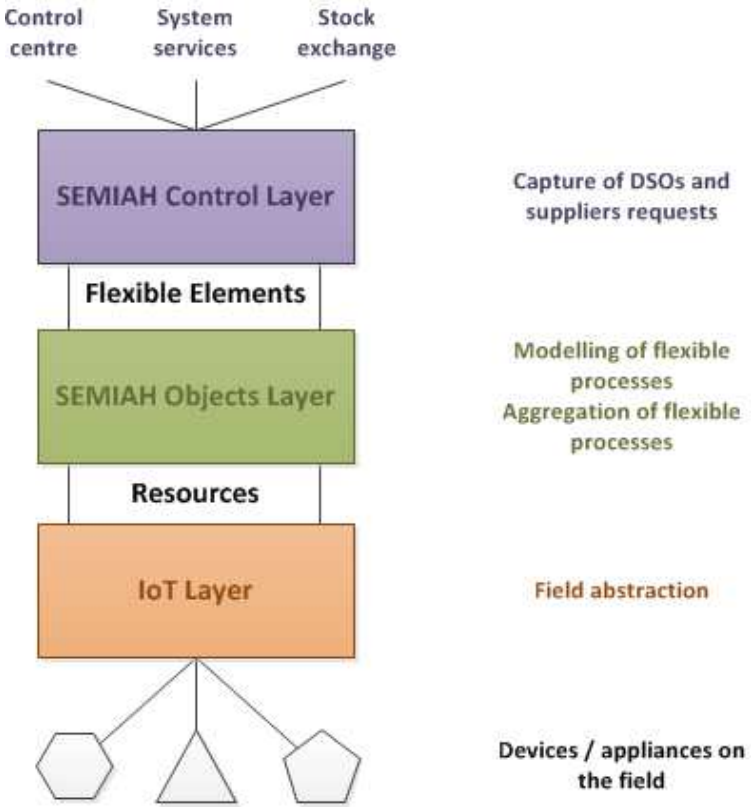


Figure 1: SEMIAH layered model (from D3.2)

The IoT Layer belongs to the SEMIAH front-end, whereas the SEMIAH Control Layer is part of the back-end. The SEMIAH Objects Layer is shared between front-end and back-end: objects related to households or to households elements are related to the front-end, and objects aggregating households are related to the back-end.

The present document describes shortly the SEMIAH front-end prototype developed in the frame of Tasks T4.1 and T4.2.

In a first approach, it seems logical to implement the front-end in the distributed HEMGs and that is basically the case for SEMIAH. However some front-end functions are implemented centrally because deployment and operation are easier.

1.1.1 The IoT layer

The IoT Layer provides an object-oriented view of appliances related to demand response: heat pumps, direct electrical heating systems, boilers, photovoltaic generation units, household electrical meters, room temperature sensors…

1.1.2 The SEMIAH Objects layer

The SEMIAH Objects Layer aims to provide an abstract view of flexible entities like a single process, a household, a feeder... The main object implemented in the SEMIAH front-end infrastructure is the household.

Two concepts are associated with household objects:

- the Flexibility Forecast component, which expresses the flexible consumption of the household in the near future, and
- the Real Time Appliance Scheduler component, which schedules appliances in real-time.

Flexibility forecasting is implemented by a single component for all households and is therefore treated as a back-end component even if it generates a forecast per household.

1.1.3 Context for the SEMIAH front-end prototype

On could say in a simplistic fashion that a working Demand Response system requires an infrastructure and intelligence. Infrastructure deals with sensors, actuators, gateways, protocols, distributed information systems… whereas intelligence covers the control logic.

This document present the prototype infrastructure developed for the SEMIAH front-end. That infrastructure has been designed to be (as far as possible) independent of the intelligence, which is under development in WP5.

1.2 The two front-end flavours

1.2.1 Why two front-end flavours?

In SEMIAH, two generations of front-end infrastructure have been developed:

- Gathering hands-on experience with the real behaviour (as opposed to behaviour in simulation) of electrical space heating systems and sanitary hot water systems is crucial for the success of the SEMIAH pilot. Hence a prototype version of the SEMIAH system is already deployed in Swiss and Norwegian buildings during the winter 2015 – 2016. This pre-pilot infrastructure will monitor buildings, under normal operation and in case of simple demand response events. A *field test front-end system* has been developed for that purpose.
- A prototype version of the *full front-end system* has been developed. The ICT part of the full front-end system is available, but the hosted intelligence is still being developed in simulation and has not been ported to the full front-end system.

The field test front-end system can be considered as a proof of concept deployed in real environments. Encountered problems will be analysed, and relevant fixes will be performed. Once validated, field test front-end features will be added step by step (agile process) to the full front-end, which is running in laboratory as of today.

1.2.2 Field test front-end system

The field front-end system is schematically represented on Figure 2.

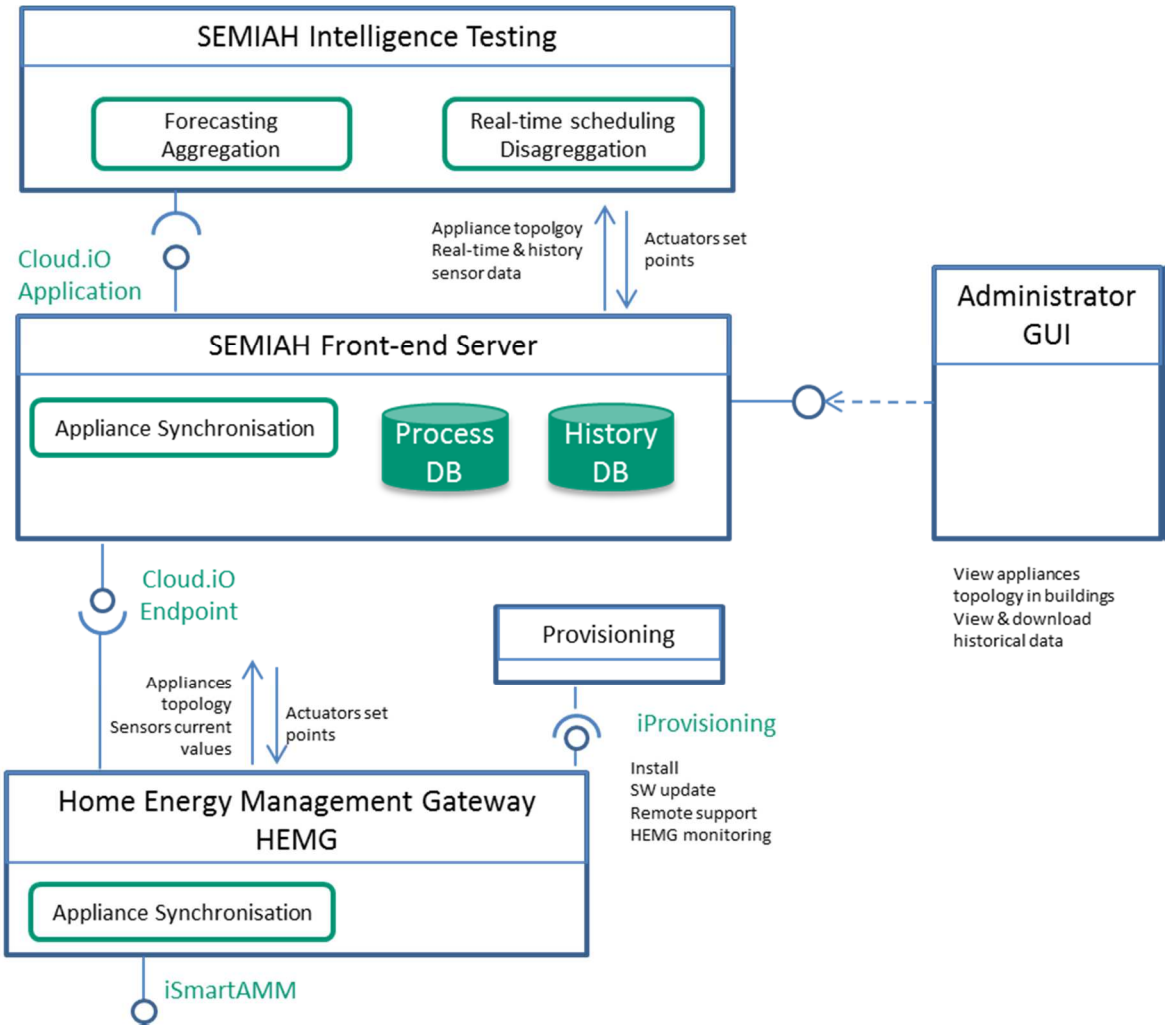


Figure 2: Block schema of the field test front-end system

Basically, the HEMG (in its field test version) contains only a subset of the IoT layer. The current status of sensors and actuators is mapped in a server side database called the Process Database. Past values are stored in a second database called the History Database. Actuators set points are essentially power relay enabling / disabling the supply of electrical loads.

For the development and validation of the SEMIAH intelligence, an interface allows test programs or scripts to query historical values, to receive real time sensor values and to set actuators set points in real time.

A web server provides a short description of the monitored households and gives access to a graphical representation of historical values.

1.2.3 Full front-end topology

Figure 3 shows part of the full SEMIAH technical architecture as introduced in D4.2. Differing from D4.2, the front-end server from the field test front-end system (cp. section 1.2.2) was added to the picture.

The core components of the back-end are:

- the generic virtual power plant (GVPP) which is connected to a multitude of households,
- the Flexibility Forecast, and
- the SEMIAH algorithms

The flexibility forecast is essentially delivering predictions of the household's load flexibility to the GVPP. The GVPP again aggregates this with additional information, e.g. grid constraints from the DSO or measurements from the households directly, and subsequently either calculates operation schedules for the households internally or delivers information to the external SEMIAH algorithm component, which in turn computes operation schedules and sends them back to the GVPP. Schedules are then forwarded from the GVPP to the households. Each schedule describes the range in which the household load power shall stay for a future time period (e.g. one day).

The core components of the front-end are:

- the Home Energy Management Gateway (HEMG), and
- the front-end server.

From the schedules received from the GVPP, the HEMG calculates individual operation plans for controllable electric devices such that the target household load range is met. This step is called "disaggregation". The HEMG controls and supervises operation of the devices, e.g. room heaters or tap water boiler, according to the plans. Also, it collects operation measurement data, e.g. room temperatures and actual values of device consumption, and sends them to the flexibility forecast module. By this way, a closed loop system is set up. The User GUI is providing the end customer (household inhabitants) access to the HEMG, while the provisioning is providing basic functions for remote HEMG software maintenance and updating.

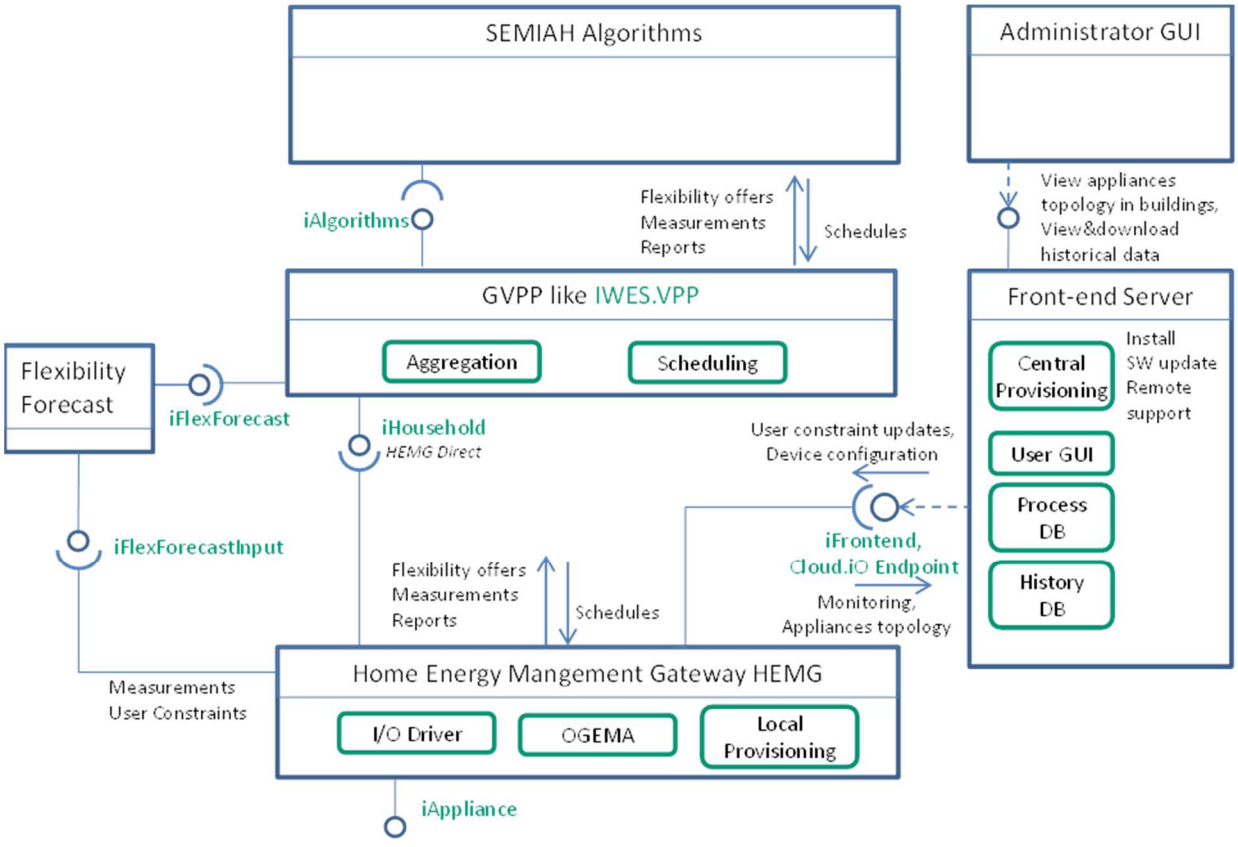


Figure 3: SEMIAH Full front-end technical architecture

1.3 Use of cable modems as Home Energy Management Gateways

Some cable modem manufacturers already integrate a Zigbee interface in some of their modems. Such modems could therefore work as HEMG and be an interesting alternative to the Develco Squid.link device, hence reducing the number of devices in households (HEMG and cable modem in one device).

Different solutions are used by manufacturers to separate the cable modem function and the supplementary non-core services (HEMG for example): the supplementary services are hosted by a dedicated processor, by a dedicated virtual machine or by a proprietary “sands box” framework.

The following modem manufacturers have already products which possibly could be integrated in the SEMIAH project:

- Icotera FTTH Gateway I6800
- Hitron CGNV4 Docsis 3.0 eMTA Wi-Fi Gateway
- Arris DOCSIS Gateway (model to be defined)

However, in most of the cases, these manufacturers presently have no readily usable realization. They are very interested in discussing (or even collaborating) with the SEMIAH project to see how they could integrate the SEMIAH application in their modems.

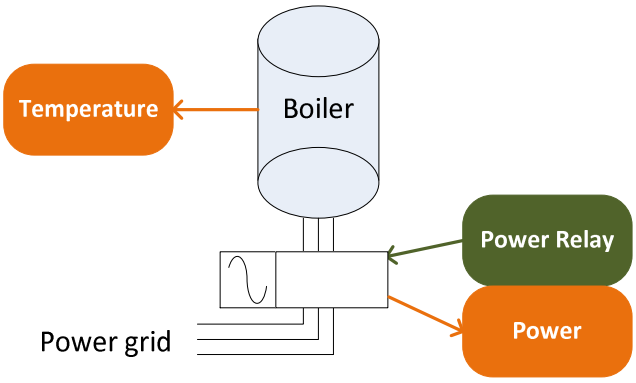
Although we acknowledged their interest, from our point of view, these discussions must be delayed to a point where the SEMIAH concept works correctly on the Develco Squid.link platform. When this is ready, then it would be possible to take up the contacts with the modem providers, based on our working concepts.

2 Hardware setting in households

2.1 Typical SEMIAH household topology

Demand response in households can be implemented by controlling either thermal appliances (space and water heating if performed electrically, cooling), white appliances or in general electrical (high) power consumer products and appliances like e.g. electrically vehicles.

SEMIAH addresses all these categories of appliances. However the SEMIAH pilot is limited to the control of thermal appliances producing heat for space heating and hot water preparation. Figure 4 presents a sample topology of a SEMIAH setting in a pilot household.



(a)

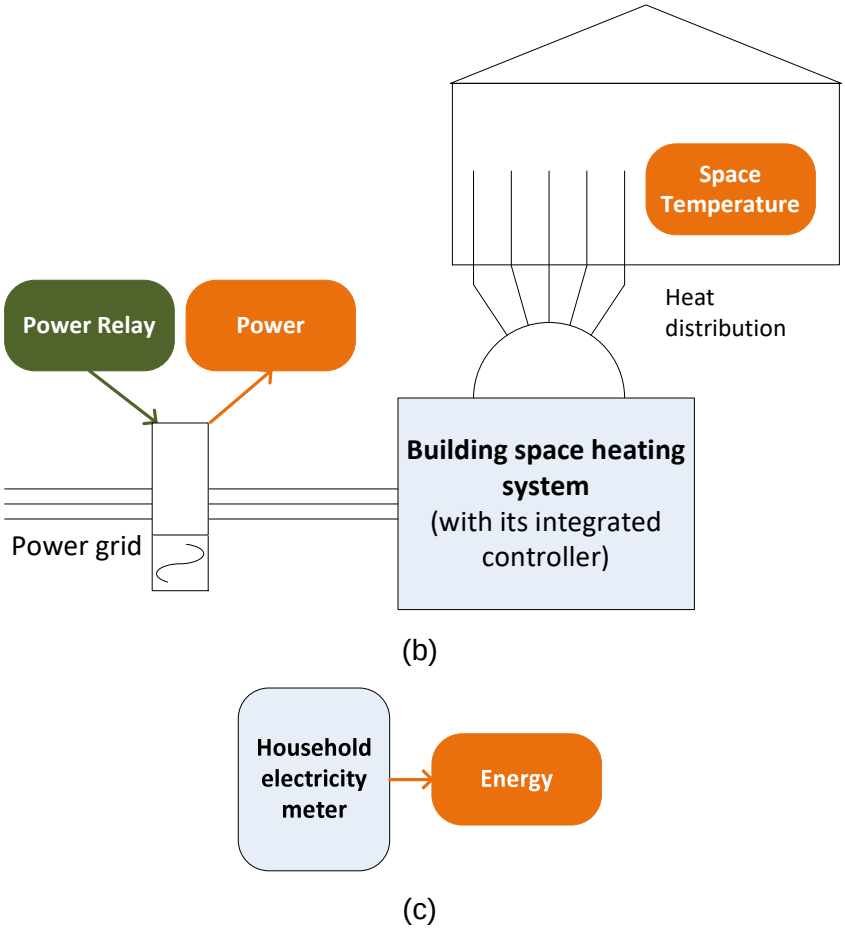


Figure 4: Typical setting in a SEMIAH household (orange box: sensor; green box: actuator)

Further types of sensors / actuators can be added if useful.

2.2 List of supported Zigbee devices

The current version of the SEMIAH household infrastructure supports the types of Develco Zigbee devices listed in Table 1.

Develco product ID	Device name	Device Role in SEMIAH
MGW101-DP3	Gateway	HEMG
ZHOT101	ZigBee Occupancy, Temperature & Light Sensor	Detection of occupancy, light & temperature.
ZHEMI101	External meter interface (EMI)	Real-time measurement and reporting of the household's power consumption (Pulse counter).
SMMZB310	Prosumer meter (triple 3-phase meter)	Monitoring and live reporting of the home's solar cell energy usage and output.
ZHDR201	Smart phase relay 16A	Remote power control for clusters of devices and wireless stream of consumption data (230V).
ZHDR201	Modified smart phase relay 16A	Remote power control for clusters of devices and wireless stream of consumption data. Control command of HP or boiler (<230V).
ZHWR202	Smart plug 1 phase 16A	Monitoring and live reporting of consumption of plugged device.
ZSMR101	Relay 30A	Monitoring, live reporting and control of consumption.
(Specific product developed for SEMIAH)	Temperature probe	Monitoring and live reporting of boiler tank temperature.

Table 1 Zigbee devices used for the SEMIAH front-end

2.3 Per household infrastructure

The control infrastructure in each building is sketched on Figure 5 (logical view) and on Figure 6 (topology in building for space and water heating control). Each household will be equipped by a HEMG provided by DEVELCO, which works as Zigbee access point. Basically any type of Develco Zigbee devices (and most probably also any type of Zigbee devices compliant with the Smart Energy or Home Automation profiles although this feature has not been tested) can be included in the household infrastructure.

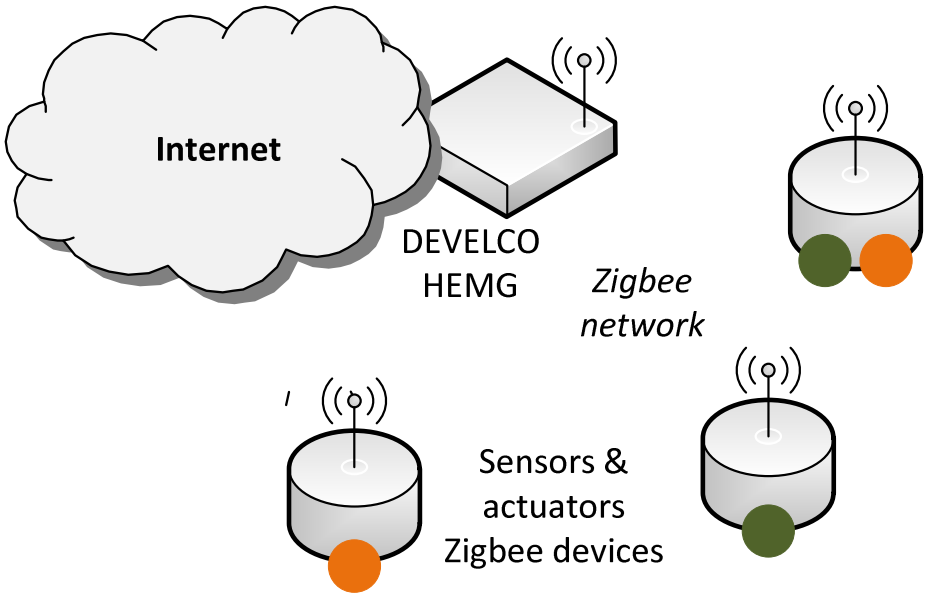


Figure 5: Topology of the control infrastructure in SEMIAH households

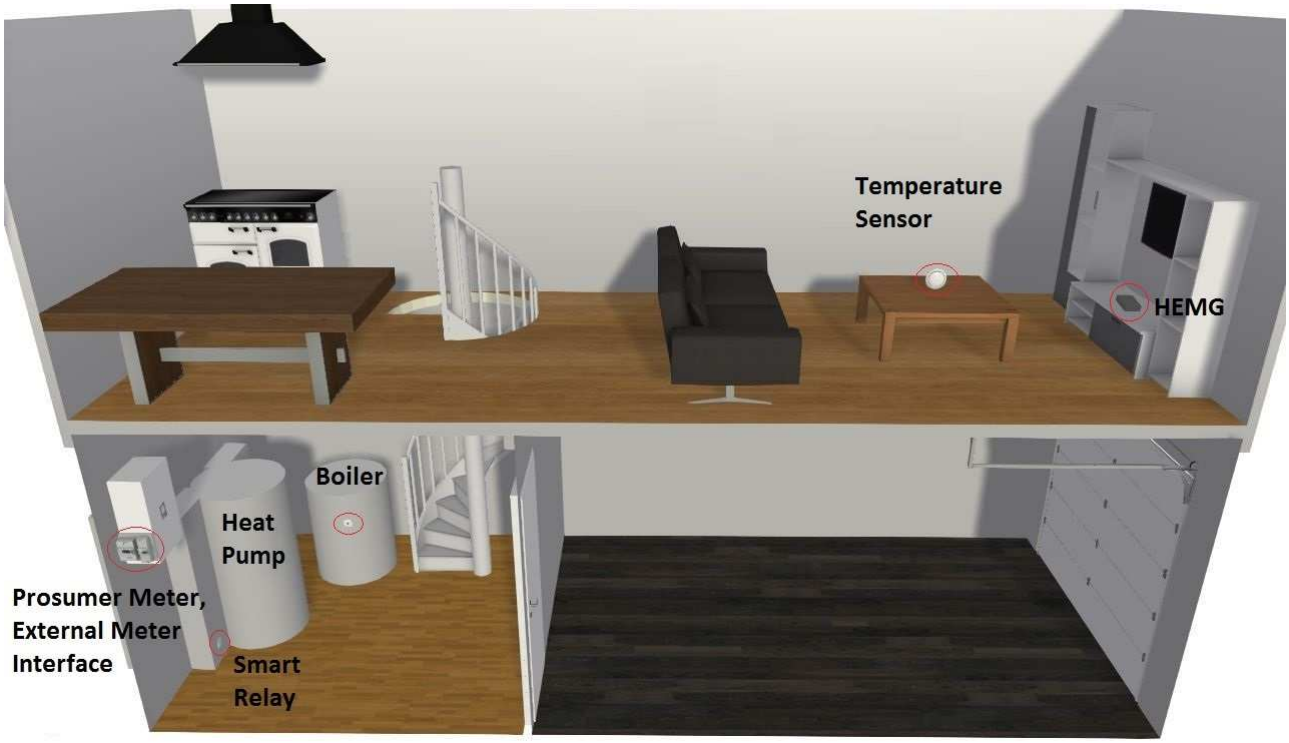


Figure 6: Topology in building for space and water heating

2.4 Hardware features of the Develco HEMG

The DEVELCO Squid.link HEMG has been developed in the frame of T4.2. Its hardware and firmware features are summarized in Table 2:

Processor	Processor type
Size	10 x 10 x 2,8 cm
Processor	ARM926EJ – 454 MHz
RAM	DDR2 128 MB (optional 256MB)
Flash	NAND 256 MB (optional 512 MB)
Ext. memory	Optional uSD for memory up to 32 GB
Power supply	5V adapter (optional Power over Ethernet) Typ < 2W
Ethernet	10/100 Mbit (optional PoE)
USB	USB2 Host (hidden under the box)
ZigBee	ZigBee HA1.2 certified PAN Coordinator (combined Interface)
WLAN – Optional	IEEE802.11 b/g/n – AP and Station (multiple networks)
Wireless M-Bus - Optional	EN13757 compliant – mode T and C (868 MHz) Master
Z-wave - Optional	868 or 916 MHz module (ZM5304) – Master modem software
3G - Optional	HUAWEI E3533 3G USB dongle (in hidden USB compartment)

Table 2: Features of the DEVELCO HEMG

3 Field test front-end system

3.1 Components of the field test front-end system

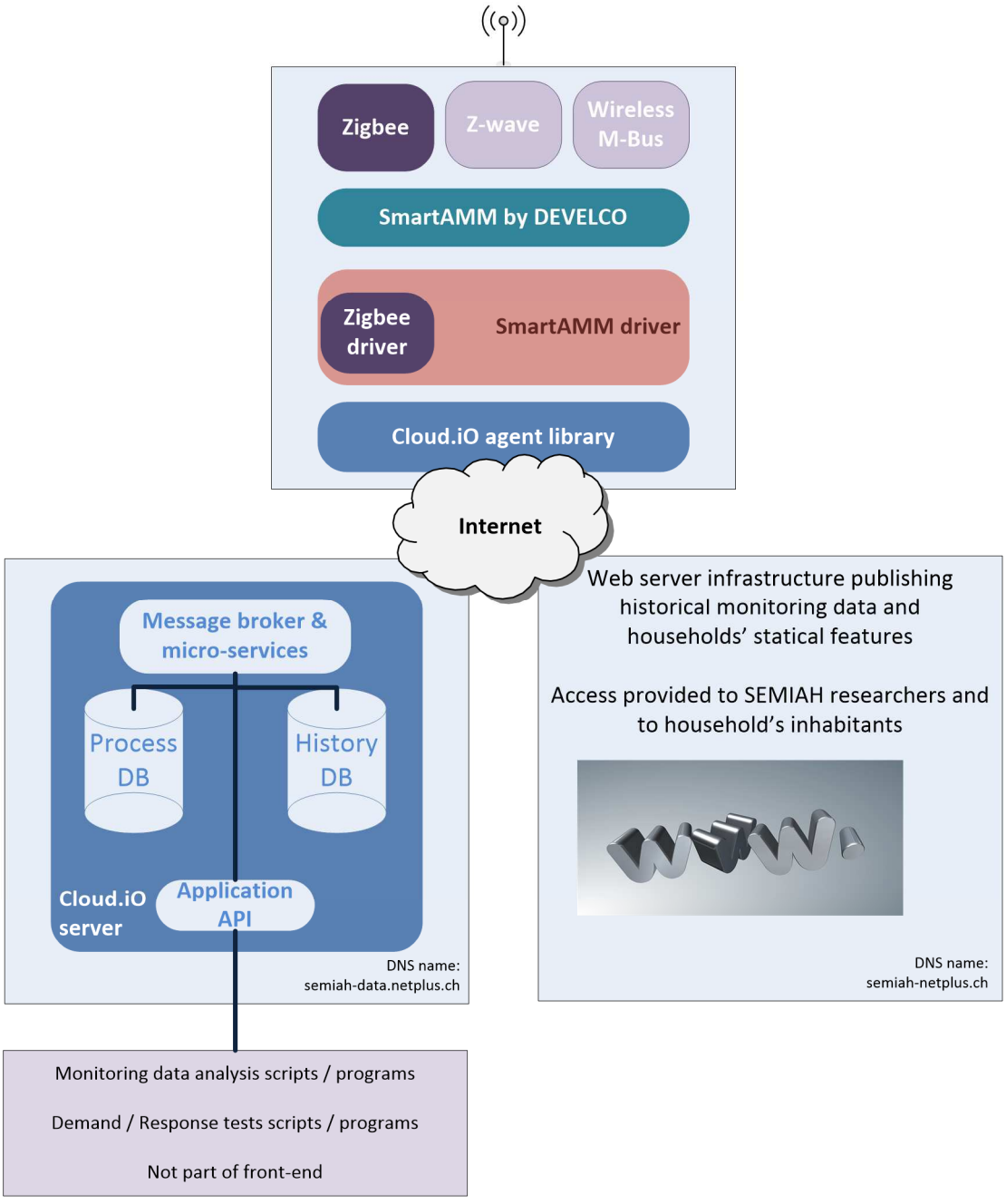


Figure 7: Components of the field test front-end system

3.2 The SmartAMM & Zigbee drivers

A stream socket communication channel can be established to the "SmartAMM by DEVELCO" component as represented on Figure 7. Communication on this stream is governed by a proprietary protocol named "SmartAMM". Basically, SmartAMM encapsulates all frames exchanged on the local wireless interfaces (Zigbee, Z-Wave, Wireless M-Bus) as well as the changes of state of the unique HEMG button.

The SmartAMM driver is a software component developed in the frame of SEMIAH acting as a local stream client for the “SmartAMM by Develco component” taking over the role of a stream server (i.e. client and server are both hosted by the HEMG).

Handling of the Zigbee protocol is managed in the Zigbee driver, a further Java component developed for SEMIAH. By design, the driver can perform all Zigbee standard operation on any Zigbee device.

3.3 Virtual Private Network (VPN)

Virtual Private Networks (VPNs) provide a private and secure networking over a public unsecure network like the internet.

Even if the SEMIAH security concept does not require a VPN, HEMGs, the Cloud.iO server and test and development computers are linked by a VPN for the field test front-end phase. A VPN is not primarily used for security reasons but to allow test and development computers accessing HEMGs from a console at any time, even if the HEMGs are located behind a NAT (Network Address Translation) firewall.

It is foreseen to stop using a VPN for the full front-end system, if a sufficient level of security and reliability can be reached.

3.4 The Cloud.iO framework

3.4.1 Overview

The Cloud.iO framework[1] is an IoT framework developed by HES-SO (development is not part of SEMIAH) and used in SEMIAH to synchronize the HEMGs with the front-end server.

Cloud.iO orchestrates field-proven internet open-source components to build a secure and flexible IoT framework.

Some features of the framework are listed below:

- Cloud.iO can support any data model and can possibly enforce a given (freely defined) data model.
- Cloud.iO communication is based on a messaging service (and not on the client-server model), i.e. the HEMGs and the front-end server can exchange data at any time.
- Cloud.iO communication is secured.
- Access rights to field data points can be controlled centrally through a specific API (function in development phase).

3.4.2 Endpoint API

In SEMIAH, the Endpoint API is implemented in the HEMG. Through it, sensor data can enter Cloud.iO and actuator data can be received from Cloud.iO.

3.4.3 Databases

The Process Database stores the current status of the SEMIAH system. The current status is composed of two types of data:

- an image of the current values of the field data as captured by each HEMG, and

[1] http://cloudio.hevs.ch

- the list of on line and off line HEMGs and devices.

The History Database stores all past field monitoring data and provides access to them.

3.4.4 Application API

Through the Application API, applications can:

- query the Process and History databases,
- subscribe to get notified of any change of field data point,
- be notified in real time on the changes on subscribed data points, and
- update the set point of any actuator in real time (to be developed).

3.4.5 Security

Each Cloud.iO host (HEMG, Cloud.iO server, application accessing data) feature an X.509 certificate signed by a private Certificate Authority (CA) and a corresponding private key. Links between Cloud.iO hosts are implemented by SSL/TLS connections based on mutual certificate based authentication, in order to protect transferred data.

Cloud.iO also implements the necessary tools to enforce privacy rules by explicitly granting applications read and/or write access rights to given set of data.

3.5 Web platform

The web platform is a state-of-art LAMP (Linux Apache, MySQL, PHP) environment. Web site is under construction and will feature following functions:

- host description of field test houses, and
- provide graphical and raw access to monitoring data.

Access is password protected. Inhabitants and SEMIAH researchers may access the whole site. This policy is acceptable for the field test as concerned inhabitants are employees of SEMIAH consortium entities aware of this situation and agreeing with it.

3.6 Status of the field test front-end system per November 2015

In November 2015, the 10 Swiss sites participating to the field test are being equipped with the SEMIAH front-end infrastructure. The Develco Zigbee devices installed in each household participating to the field test are listed in Annex A.

The following are being deployed using an agile method:

- Monitoring data are collected in the Process and History databases.
- A web server with username / password protected access displays the household static features (house situation, thermal envelope features, space and hot water heating system features) in an unstructured format and provides access to households monitoring data history.
- An API for real-time SEMIAH intelligence test script is under development.

The central part of the SEMIAH front-end is hosted by NETPLUS on two virtual machines with expandable capabilities:

- The server part of Cloud.iO (Process and History Databases, message broker, Cloud.iO micro-services) is hosted on the server with DNS name semiah-data.netplus.ch.
- The web server infrastructure is hosted on a second server with DNS name semiah.netplus.ch.

4 Full front-end system

4.1 Components of the full front-end system

The software components of the full front-end as relevant in the SEMIAH context are shown in Figure 8. The figure is detailing the “Home Energy Management Gateway” component in Figure 3. Topmost, it shows connections to the back-end level, namely the flexibility forecast, GVPP and the front-end server. All elements below the dotted line are part of the HEMG and bundled by the OGEMA platform, which is detailed in section 4.2. Software modules running on this platform (“apps”) are shown as boxes, and a part of the data resource tree is shown beneath. The drawn through lines in the resource tree indicate sub-resource relations, the dotted lines indicate references or decorators. Some of the apps are connected to resources by arrow lines which indicate information flows. For reasons of clarity, some sub-resource relations were not represented by lines, but by path names. Also, only exemplary connections are shown.

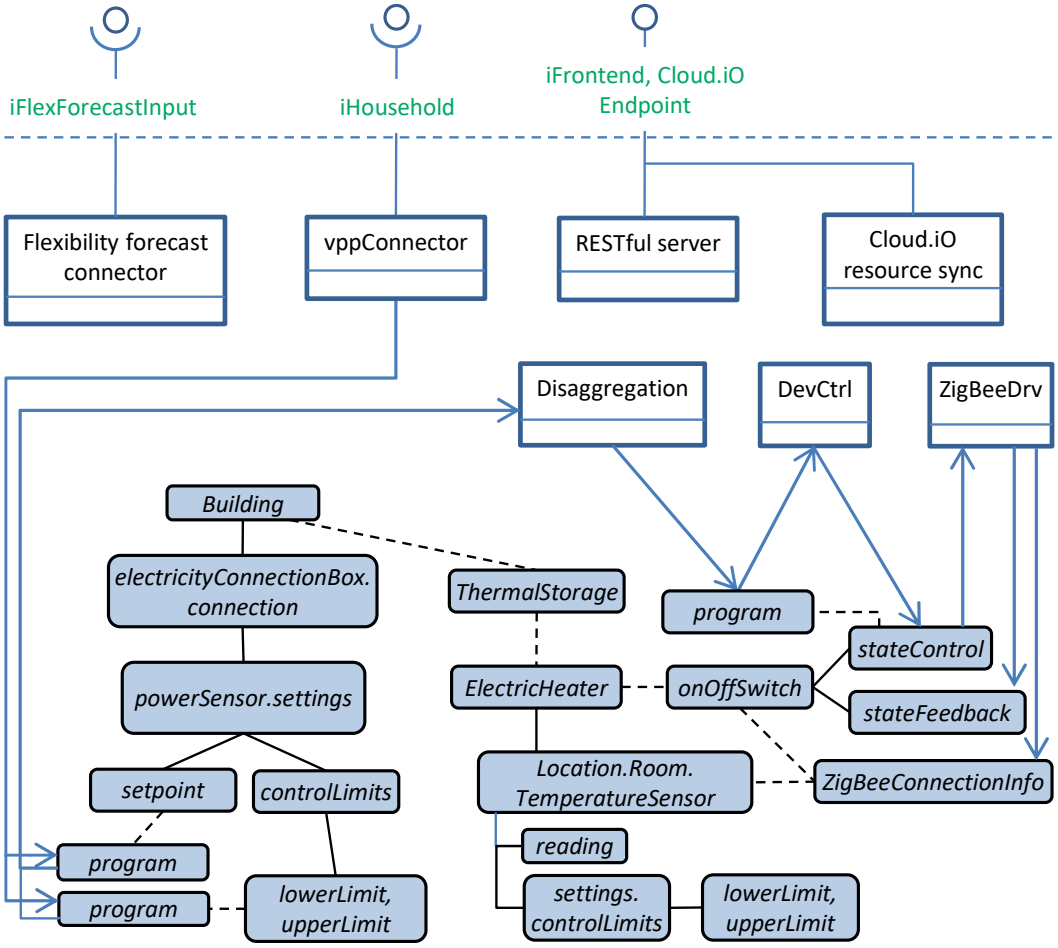


Figure 8: HEMG components

The function of the OGEMA SmartAMM driver module (“ZigBeeDrv” block in Figure 8) is described in subsection 3.2, and the functions of the other software modules in conjunction with the resources as well as the back-end are outlined in subsection 4.4.

4.2 OGEMA

OGEMA[2] (Open Gateway Energy MAnagement) is an open software platform that acts as an operating system for automated building control and energy management. It can be applied in households, commercial and industrial environments. OGEMA links the customer's loads and generators to building automation and energy management applications. By providing a manufacturer- and hardware-independent platform, OGEMA allows energy flows within end customer premises to be optimized with high degree of modularity. To be able to execute software applications from various sources on a single embedded computer a common execution environment is defined to which all these applications are deployed. OGEMA uses Java and OSGi as widely accepted software standards that provide such cross-platform execution environment. OGEMA has a modular structure and consists of the following layers:

- OGEMA Services: central services like resource administration, user interface, web interface etc.,
- Resources: data structures according to the OGEMA data model representing the connected devices,
- Applications ("apps") which implement functionality, and
- Communication Drivers, which are a special type of Applications.

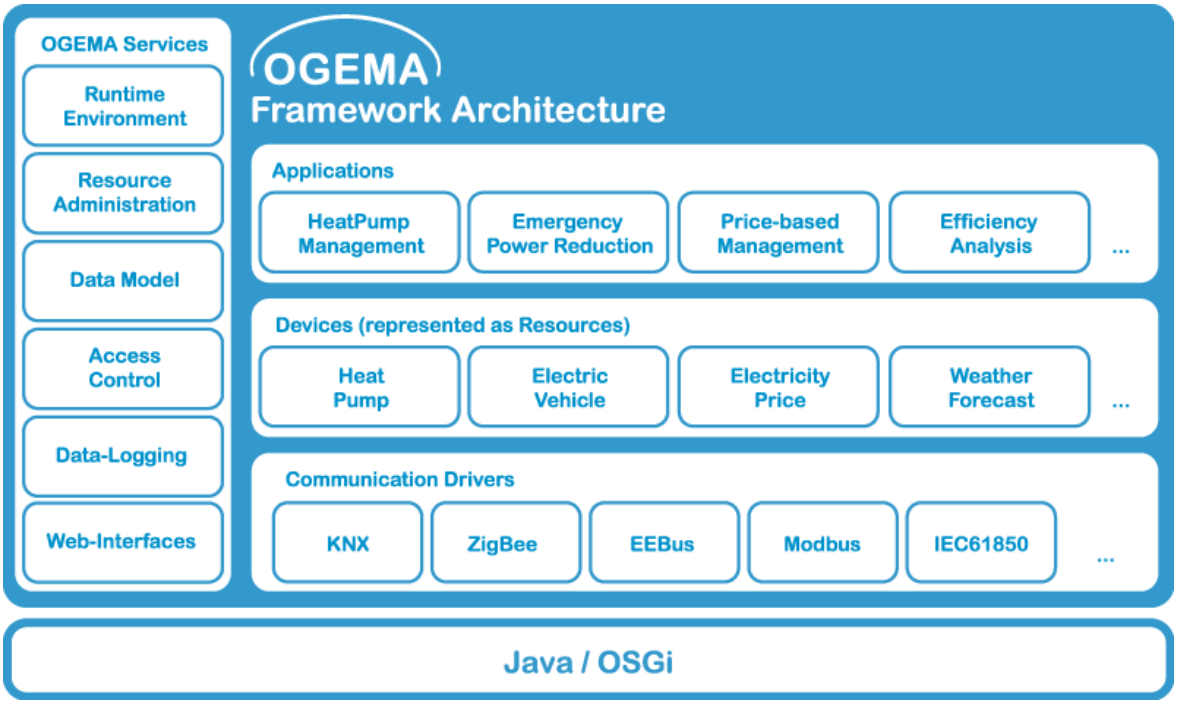


Figure 9: OGEMA Framework Architecture

The Resources follow a well-defined OGEMA standard data model. For SEMIAH, this standard data models was extended by project specific elements as documented in D4.2. As a standard machine-to-machine interface, OGEMA provides a RESTful server for reading, writing, creating, deleting and connecting resources over secure HTTPS connections. This feature is also indicated in Figure 8 and is essentially used for realization of the external prosumer GUI (cp. section 4.5) by the front-end server. The resource data model inherent to OGEMA follows an object oriented approach, leading to a tree-like hierarchical structure. For example, the resource type for representing a heat pump is an extension of the generic type *HeatGenerator* and contains a sub-resource representing the heat pump's electric connection. The plurality of resources hence may be viewed as a resource tree, which in the SEMIAH case typically contains a "Building" resource as root (as seen in Figure 8). The

[2] http://www.ogema.org

rest of the tree must be considered specific to each individual building (e.g. household), which are unknown until actual on-site installation is prepared or taking place. The resource tree is persistently stored in a graph database and automatically reconstructed during startup of the OGEMA system. Hence, it is important to understand that resources can either be created by this startup process or during runtime by the apps or by a REST client, e.g. the user may create a resource representing a new physical device through the user GUI, or the ZigBee driver may create a new resource representing a newly detected ZigBee node. Resources must be arranged at the correct places in the resource tree in order for the apps to find them at places expected. This needs special attention for resources which are connected to others by references (dotted lines in Figure 8 resource tree).

Programming paradigms often seen in OGEMA apps is event or timer based programming. OGEMA apps are threads that take action when started or stopped. During runtime, they are typically sleeping and listening to changes in parts of the resource tree. If such changes are detected by an app, e.g. a new resource becomes available or the value of an existing resource changes, it may be triggered to carry out an action. Another trigger is the elapsing of timers the app has created. The app's action again may include modifying the resource tree or changing values of existing resources, subsequently triggering other apps to take action again.

4.3 The OGEMA driver for SmartAMM

The OGEMA driver for SmartAMM creates two types of OGEMA resources:

- **Specific OGEMA resources for the management of the Zigbee network**: By setting these management resources appropriately, Zigbee devices can be discovered and added in the Zigbee trust center and binding with attributes can be performed.
- **OGEMA resources for energy management**: Bound Zigbee attributes must be linked to OGEMA resources of the corresponding types. As example, an OGEMA resource of type Temperature will be instantiated after a binding with the temperature attribute embedded in a Zigbee temperature sensor.

Before operation, the OGEMA resources for energy management have to be linked to appropriate resources. For example, the above mentioned Temperature resource should be linked to the appropriate Room resource, to indicate that the Temperature resource mirrors the current temperature of a specific Room. This is done by means of an external component which uses a RESTful service offered by the standard OGEMA framework, allowing for resource creation and linkage.

4.4 Front end components and features

In this subsection, front-end software module functions and their relations to back-end components are discussed looking at Figure 3 and Figure 8. The main front-end functionality is described best when considering several modules interacting with the back-end level in a closed loop cycle starting with the "flexibility forecast connector" module of the front-end.

The ***flexibility forecast connector*** uses a RESTful service provided by the flexibility forecast component of the back-end level in order to transmit JSON encoded objects from the front-end (acting as a client) to the back-end (acting as a server). As soon as the according building has received an id from the back-end GVPP component, the flexibility forecast connector will listen to all rooms and electric heaters present or newly created at the HEMG. The rooms are decorated with the id from the GVPP, and contain - by standard data model - a temperature sensor which features upper and lower temperature control limits. For the electric heaters, which are connected to a room resource by their *Location* property, consumed power of the heater and temperature of the connected room are recorded. The following data is sent to the back-end flexibility forecast component:

- the recorded power and temperature values (using a timer)
- the user-set schedules for the room temperature control limits (using an event listener, listening to changes made by the user)
- furthermore, an information object containing building location, id, and information about room dimensions and electric heater nominal powers is sent if a new household is created or the room or heater configuration is changed (using event listeners).

The detailed calculation done by the flexibility forecast component on the back-end level is not covered by this document, but it basically processes the data mentioned above in order to calculate some information about each building's flexibility regarding electric consumption caused by heating. The flexibility forecast offers this information to the GVPP. The GVPP again aggregates this information across all buildings and uses internal or external SEMIAH algorithms in order to calculate three schedules individual to each building. The schedules define an upper and lower limit for the buildings electric power consumption over time, as well as an ideal desired consumption within those limits. The schedules are provided towards the HEMG through the GVPP's iHousehold interface which is password secured and utilizing the CIM model standard (IEC 61970).

The HEMG software module ***vppConnector*** accesses this interface. As soon as a resource specifying connection details ("VPPConfig") to the GVPP is found at the HEMG, the module starts a get, simple get or put task depending on the connection details. Such connection detail resources are decorated onto each (parent) resource containing information originating from the GVPP, namely building or flexibility forecast ID's, and schedules for electricity consumption as mentioned above. Using a timer, the tasks fetch according information from the GVPP and write it into the parent resources. The tasks may be configured to do this only once or continuously, the interval being specified by the VPPConfig details. Looking at Figure 8, this mechanism is used by the vppConnector to write schedules ("programs") specifying upper, lower and desired (expected) consumption for the building into according sub-resources of the electric power sensor of the buildings electricity connection.

The consumption schedules thus obtained from the GVPP are referring to the whole building consumption, not to individual electric devices. Hence, device individual schedules have to be generated from them. This task is carried out by the ***disaggregation*** module. This module acts in an event based fashion, listening to building, boiler and electric heater resources on the HEMG. As soon as a building resource is detected, it registers listeners to schedules specifying upper, lower and expected building consumption. If (eventually newly created) boilers or heaters are detected, it is checked if device individual upper, lower and expected power consumption transients already exist. If not, they are created. Individual schedules for these transients are computed for each boiler and heater known by a disaggregation task as soon as all three building-level schedules were updated. The algorithms used for this are currently implemented as a stub, but will be completed until the field test start. Subsequently, schedules for the switching state of the devices are calculated (referred to as "program" written by the disaggregation in Figure 8).

The individual heating device switching schedules are then used by the ***DevCtrl*** module which assigns a controller task to each device. Using a timer, each controller continuously sets the switching state of the device as planned by the disaggregation by writing the *stateControl* resource of the device's *onOffSwitch*. Through its location, the heating device is connected to a room which is assumed to feature a temperature sensor. If the temperature of such heated room is out of the limits specified, the DevCtrl switches the heating device on or off regardless of the planned switching state. The ex-post recorded room temperatures are again sent to the flexibility forecast back-end component by the flexibility forecast connector (see above), resulting in a closed-loop control.

To facilitate remote access to the resource graph, the OGEMA framework provides a ***RESTful server*** interface supporting communication using XML as well as JSON, and other communication requirements according to RFC2616[3]. Each request to this interface must be https-encrypted and

[3] "RFC2616" [Online]. Available: http://www.w3.org/Protocols/rfc2616/rfc2616.html

supply a username identifying the user (or application) whose permissions will be used for the access, and the correct password for the user. Requests sent via REST are then performed with that user's access rights. Using this interface and provided the access rights allow for the corresponding action, resources can be read, written, created, attached to others (eventually creating references or decorators), or deleted.

The ***Cloud.iO resource sync*** application is used to synchronize the OGEMA-level resources (e.g. temperature sensor values) with the front-end server using Cloud.iO. This allows the front-end server to record real-time and historical sensor values, which is already the case during the field test- phase (cp. Section 3.4)

From the explanation above it is obvious that the front-end system can only work if resources representing the buildings physical components are correctly created and arranged in the resource tree. Since it is unknown which physical components are present until installation takes place, resource configuration must be done by the user or admin GUI. The resources that need to be created and set-up include the thermal storages, rooms, electric heaters, boilers, the sensors, the onOffSwitches, and according ZigBee Devices. Cross-references between resources, e.g. to correctly link an electric heater and a temperature sensor with a room, are also made through this external GUI that supports the installation process (cp. section 4.6). Also, the GUI should provide means to delete parts of the resource tree in order to allow for corrections of incorrect resource arrangements. All of this can be done by using a RESTful service offered by the OGEMA framework which allows for resource access by authorized users using an SSL encrypted connection. This mechanism is the standard OGEMA machine-to machine interface.

4.5 Prosumer GUI

The pilot participant GUI of the full front-end system will be a development of the web platform of the field test front-end system (section 3.5).

The user GUI will be web based and the web server will be a further development of the web platform presented in paragraph 3.5. Development will be done in the frame of WP7, according to pilot needs.

4.6 GUI to support deployment by electricians

This GUI is meant to support the installation process by an electrician.

During the pre-pilot based on the field test front-end, deployment will be done by electricians and engineers. The latter will get a better understanding of the electricians' needs. Hence they will be able to specify and then develop a GUI to support deployment of pilot installations.

5 Provisioning concept

The following concept for the provisioning is foreseen. The steps are adapted from the modems provisioning in an ISP environment.

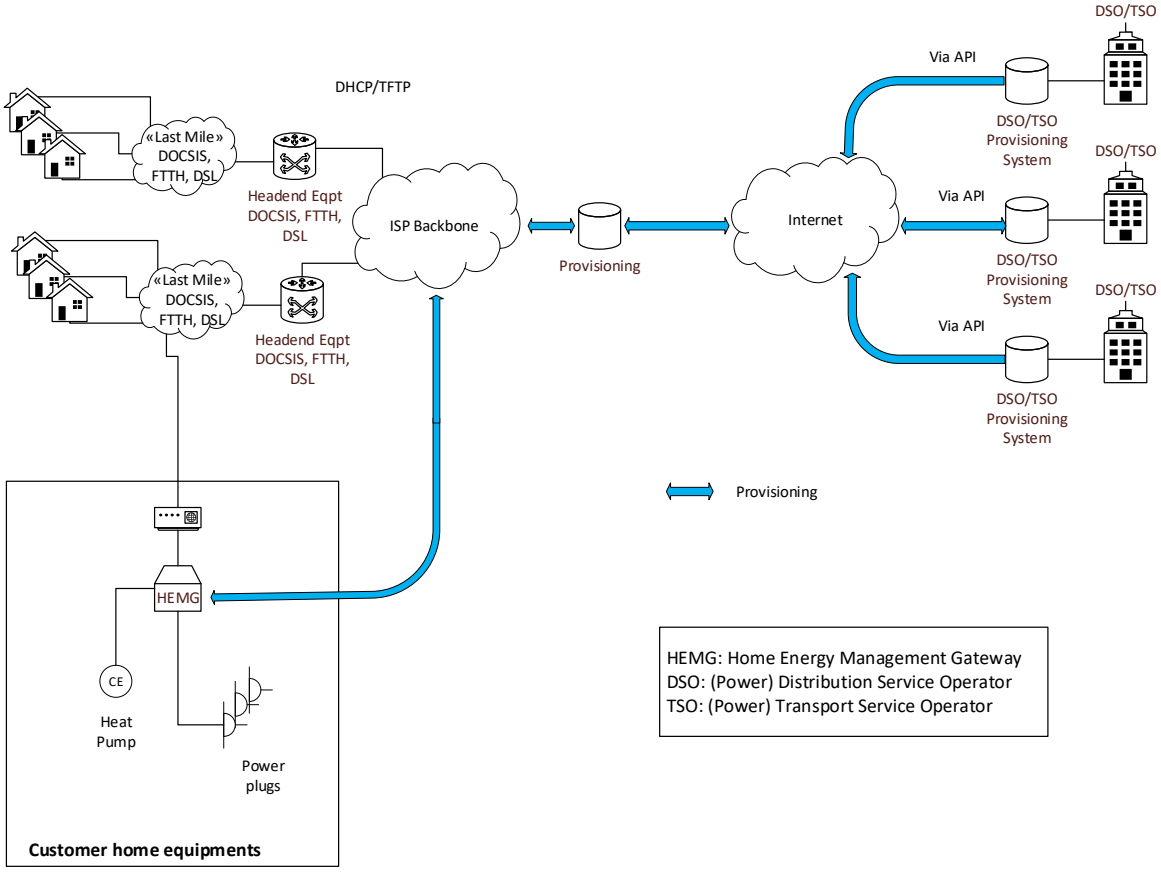


Figure 10 Provisioning concept

The following steps are foreseen:

1. The DSO provisioning system enters the necessary data on the SEMIAH provisioning database via dedicated API.
2. The HEMG receives an IP address (public or private) from ISP (or internal equipment) DHCP server.
3. The HEMG announces its presence to the provisioning system with a unique ID. This is done via a TCP provisioning connection, which will be kept open also for the monitoring.
4. The provisioning system checks the ID to be sure that this HEMG is a recognized one.
5. The HEMG checks if a software update is required and, if required, performs it using the provisioning system.
6. Finally the HEMG connects to the GVPP and front-end servers and enters the standard operation mode.

6 List of front-end software components

Software component name	Licensing model	Internet	Role in SEMIAH
SmartAMM driver	Closed source yet; open source licensing foreseen	Bitbucket: https://bitbucket.org/semiah/smartamm-library-java	Connection to the SmartAMM by DEVELCO component
Zigbee driver	Closed source yet; open source licensing to come	Bitbucket: https://bitbucket.org/semiah/zigbee-library-java	Universal Zigbee driver
OGEMA driver for SmartAMM	Closed source yet; open source licensing foreseen	Bitbucket: https://bitbucket.org/semiah/semiah-ogema	Links SmartAMM and OGEMA resources
Cloud.iO	Open source (MIT license)	http://cloudio.hevs.ch Github: https://github.com/cloudio-project	Front-end server base component HEMG communication library
OGEMA	GPL version 3 (final licensing model for commercial licenses under evaluation.)	http://www.ogema.org Github: https://github.com/ogema/	“Operating System” for HEMG

Table 3 Repositories and licensing model for SEMIAH front-end components

7 Annex A: List of ZigBee devices installed for the field test

Building #	Building type	Space heating	Domestic water heating	PV	Relays for ripple control	Modified relay	Smart plug	Relay standard	3x 3 phase meter	LED Probe	Temp. probe	Multi-sensor	Gateway
1	Multi-family house	Direct electrical central heater	Electrical boiler	No	Water boiler, electrical heating, washing machine			6			1	2	1
2	Single family house	One heat pump for both tasks		No		1				1	1	2	1
3	Single family house	One heat pump for both tasks		No	Heat pump	1	1		1		1	2	1
4	Single family house	Heat pump	Heat pump	Yes		1			1		1	2	1
5	Multi- family house	One heat pump for both tasks		No		1			1		1	2	1
6	Single family house	One heat pump HP for both tasks		No		1				1	1	2	1
7	Double single family house	Electrical floor heating	Heat pump, boiler500 l	8k W	Water boiler, storage heating	4			1		1	2	1
8	Chalet	Electrical heating 15 kW, water tank	Electrical boiler 200 l	~10 kW	Water boiler, storage heating full & reduced power	3			1		1	2	1
9	Single family house	Heat pump	Heat pump	No		1				1	1	2	1
10	Single family house with ancillary use	Electrical radiators	Electrical boiler	No				7	0		1	2	1