Small or medium-scale focused research project (STREP)

Energy Demand Aware Open Services for Smart Grid Intelligent Automation

SmartHG

EU FP7 Project #317761

Deliverable D3.3.2

Third Year Prototype of Home Intelligent Automation Services

Deliverable due on	:	M36
Output of WP	:	WP3
WP Responsible	:	HMTI

Consortium

Participant Organization Name	Participant Short Name	Country
Sapienza University of Rome	UNIROMA1	Italy
Aarhus University	AU	Denmark
IMDEA Energía	IMDEA	Spain
A. V. Luikov Heat and Mass Transfer Institute of the National Academy of Sciences of Belarus	HMTI	Belarus
ATANVO GmbH	ATANVO	Germany
Panoramic Power	PANPOW	Israel
Solintel	SOLINTEL	Spain
SEAS – NVE	SEAS	Denmark
Kalundborg Municipality	KAL	Denmark
Minskenergo	MINSKENG	Belarus
Develco Products A/S	DEVELCO	Denmark

Document Information

Version	November 18, 2015, 05:12
Date	November 18, 2015
Contributors	UNIROMA1, AU, IMDEA, HMTI, ATANVO, PANPOW, DEVELCO
Reason for release	Third year review
Dissemination level	Public (PU)
Status	Final

Project title	Energy Demand-Aware Open Services for Smart Grid Intelligent Automation
Project acronym	SmartHG
Project number	317761
Call (part) identifier	FP7-ICT-2011-8

Work programme topic addressed	
Challenge	**6**: *ICT for a low carbon economy*
Objective	**ICT-2011.6.1** *Smart Energy Grids*
Target Outcome	**d**) Home energy controlling hubs that will collect real-time or near real-time data on energy consumption data from smart household appliances and enable intelligent automation.

Project coordinator	Enrico Tronci
E-mail	tronci@di.uniroma1.it

Contents

Executive Summary

The main objective of the SmartHG project is to develop effective Intelligent Automation Services (IASs) to minimise users energy bill for end residential users while optimising operation on the grid for Distribution System Operators (DSOs). This deliverable, together with Deliverable D4.3.2, describes the third year effective implementation of such IASs. Namely, in this deliverable we focus on the IASs working on the residential users side, i.e., on the Home Intelligent Automation Services (HIASs). Instead, Deliverable D4.3.2 focuses on the Grid Intelligent Automation Services (GIASs), which work on the DSO side. Following the design described in Deliverable D3.3.1, the HIASs developed in the third year are the following: the Energy Usage Reduction for Control (EUR-K), Energy Usage Reduction for Homes (EUR-H), Energy Usage Modelling and Forecasting for Control (EUMF-K), Energy Usage Modelling and Forecasting for Homes (EUMF-H) and Energy Bill Reduction (EBR) services. The implementation of the HIASs is based on the HIASs design described in Deliverable D3.3.1.

In the second year, for each of such HIASs we developed a Web service (for human users) and a REpresentational State Transfer (RESTful) service (for automatic communications between IASs), basing on a common and unified software architecture. In this third year, we integrated such architecture in the SmartHG Platform Technical Section, by also simplifying and clarifying the user interaction. Moreover, we made the EUR-K service a Software as a Service (SaaS), thus easing usage of such service by residential users.

The effectiveness of the prototypes described here is evaluated, together with the GIAS prototypes, in the third year iteration of the SmartHG IASs evaluation (Deliverable D5.3.1).

Chapter 1

Introduction

This deliverable describes the third year version of the prototypes for the SmartHG HIAS. To this aim, we recall that, during the second year activities, we defined a common architecture for all SmartHG IASs (see Figure 1.1). Within such architecture all IASs have a Web service and a RESTful service, in order to be used by both human users and by other (automatic) IASs. During the third year, we mainly worked in integrating such services in the SmartHG Platform developed during the third year, and most notably in the SmartHG Technical Section described in Deliverable D7.3.2. Moreover, we also updated the EBR service in order to reflect the changes in the Demand Aware Price Policies (DAPP) design described in Deliverable D4.3.1. Finally, we completed the work on the EUMF-H and EUR-H Web services.

In this deliverable, we outline the work done during the third year on the HIASs prototypes, by focusing on the one which needed most effort, i.e., on EBR.

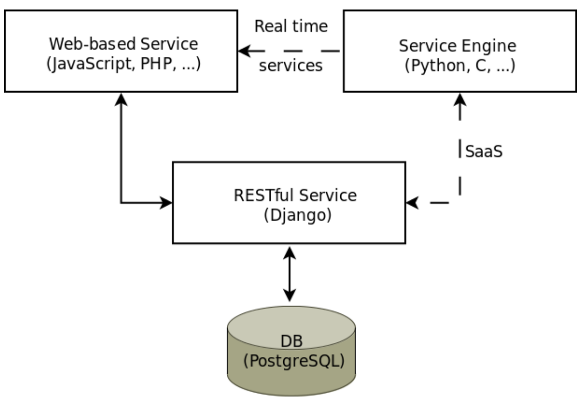

Figure 1.1: Software architecture for real-time services and for SaaS. Communications which take place for both SaaS and real-time services are shown with arrows. Dotted arrows show communications which take place either only for SaaS or only for real-time service (to allow service download). For the DB&A service, only the first two layers are used.

Chapter 2

EBR Web Service and Tarball File

In this section, we describe both the EBR Web service [1] and the EBR tarball archive file which may be downloaded from [1]. For the design of the service and the nomenclature used, we refer the reader to Deliverable D3.3.1. As for the Web service, it only allows to download the EBR tarball file, namely `ebr-2.0.tgz` (see Figure 2.1). Note that the user manual for the EBR tarball file (namely `ebr-2.0.pdf`) is also available for download.

Once `ebr-2.0.tgz` has been downloaded on a Linux machine, it is possible to unzip it by typing on a terminal the following command:

```
tar xfz ebr-2.0.tgz; cd ebr-2.0
```

This creates a new directory named `ebr-2.0`, with the following directories: `aux_files`, `example_files` and `src`. Moreover, an explanatory `README.txt` file is provided, together with a wrapper Bash script `launch.sh`.

In order to run EBR, the main requirement is to have either CPLEX or GLPK installed on the machine (i.e., either the command `cplex` or the command `glpsol` must be available in the system or user `PATH`). Given this, the user may directly launch the Bash script:

```
bash launch.sh
```

This will launch EBR with default settings, which will reproduce the BU experiment on the Kalundborg scenario described in Deliverable D5.3.1. In order to customise the input, the `launch.sh` script accepts the following command line arguments (note they are all optional: when an argument is not given, the default is used).

`-h`: prints an help message with all arguments and defaults. Des not run EBR.

`-scen` *s*: where *s* is either `BU, BC, BD, PC, PD` (see Deliverable D3.3.1 and Deliverable D5.3.1).

`-mod` *m*: where *m* is either `energy, co2, dapp` (see Deliverable D3.3.1 and Deliverable D5.3.1).

`-l` *h*: uses *h* as the number of hours to be forecasted for each charge/discharge action computation (default is 6).

-f f_1**:** uses f_1 as the Comma Separated Value (CSV) file with the aggregated demand history (default is `example_files/kalundborg/profile.csv`, which may be used as an example for the format).

-pr f_2**:** uses f_2 as the CSV file with the power profile for the given home (default is `example_files/kalundborg/pricepolicies.csv`, which may be used as an example for the format).

-pev f_3**:** uses f_3 as the CSV file with the Plug-in Electric Vehicle (PEV) characteristics (default is `example_files/kalundborg/pev.csv`, which may be used as an example for the format).

-bc Q**:** uses Q as the Energy Storage System (ESS) capacity (i.e., maximum energy storage) in kWh (default is 9).

-br R**:** uses R as the ESS maximum charge/discharge rate in kW (default is 2).

-bde α**:** uses $\alpha \in [0, 1]$ as the ESS efficiency coefficient for discharging (default is 0.82). That is, if a discharge action a is sent to the ESS for 1 hour, then the ESS discharges of a kWh, but the energy provided is $a\alpha$ instead of a kWh.

-bce β**:** uses $\beta \in [0, 1]$ as the ESS efficiency coefficient for charging (default is 0.98). That is, if a charge action a is sent to the ESS for 1 hour, then the energy provided to the ESS is a kWh, but the ESS charges of $a\beta$ kWh instead of a kWh.

-pc Q_p**:** uses Q_p as the PEV capacity (i.e., maximum energy storage) in kWh (default is 16).

-ppr R_p**:** uses R_p as the PEV maximum charge rate in kW (default is 13).

-pce β_p**:** uses $\beta_p \in [0, 1]$ as the PEV efficiency coefficient for charging (default is 0.876). That is, if a charge action a is sent to the PEV for 1 hour, then the energy provided to the PEV is a kWh, but the PEV charges of $a\beta_p$ kWh instead of a kWh.

-cc C_u**:** uses C_u as the maximum threshold for power usage in kW (default is 18).

-cp P_u**:** uses P_u as the maximum threshold for power production in kW (default is 6).

-p p**:** uses p as the number of days in the past to be used for forecast (default is 10).

-pd p_d**:** uses p_d as the discounting factor for the days in the past. Format is x_1:...:x_p, and $\sum_{i=1}^{n} x_i = 1$ must hold (default is $\frac{1}{2}$:$\frac{1}{2^2}$:...:$\frac{1}{2^9}$:$\frac{1}{2^9}$).

-ndmilps: do not delete auxiliary files for Mixed-Integer Linear Programming (MILP) problems.

Finally, the output of EBR is the log of the decided actions (for both the ESS and the PEV, when available) and their effect on the resulting home demand. Such output is stored in the CSV file `results/`$\tilde{t}$`/output/results.csv`, being $\tilde{t}$ the time-stamp at which `launch.sh` was started. Namely, `results.csv` contains the following information, for each time-slot in the execution (note that EBR actually starts to compute charge/discharge actions only after $24p$ hours):

- starting and ending time-stamps of the current time-slot t;

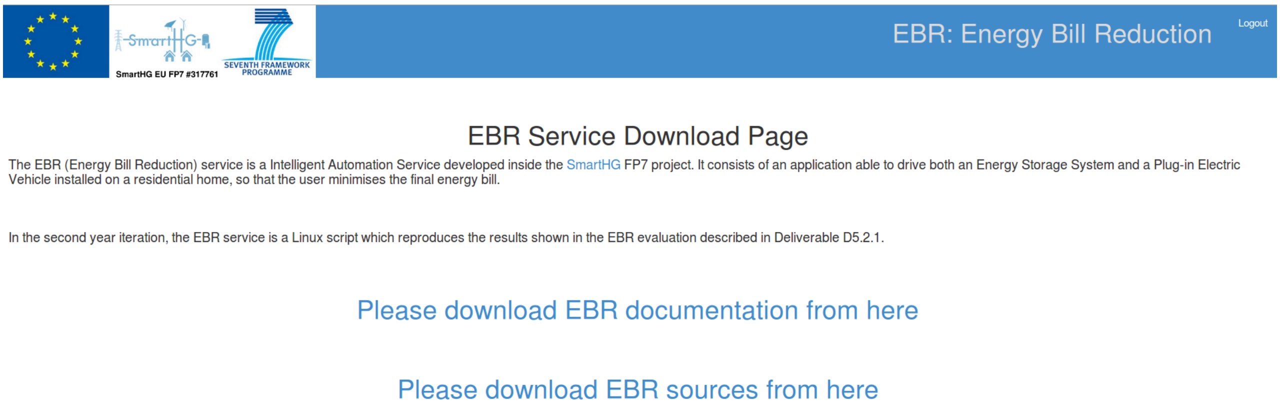

Figure 2.1: EBR service: page for download

- overall home demand $d(t)$ (in kW) without using EBR, i.e., without ESS and where PEV is not managed by EBR;
- ESS action $a_e(t)$ computed by EBR for t (in kW),
- PEV action $a_p(t)$ computed by EBR for t (in kW),
- overall home demand $\tilde{d}(t)$ (in kW) using EBR (that is, after ESS and PEV actions application), i.e., either $\tilde{d}(t) = d(t) + a_e(t) + a_p(t)$ (if $a(t) \geq 0$) or $\tilde{d}(t) = d(t) + a(t)\alpha + a_p(t)$ (otherwise);
- ESS remaining capacity $b_e(t)$ (in kWh), i.e., either $b_e(t) = b_e(t-1) + \tau a(t)\beta$ (if $a(t) \geq 0$) or $b_e(t) = b_e(t-1) + \tau a(t)$ (otherwise), being τ the length of time-slot t;
- PEV remaining capacity $b_p(t)$ (in kWh), i.e., $b_p(t) = b_p(t-1) + \tau a(t)\beta$ (if $a(t) \geq 0$) or $b_p(t) = b_p(t-1) + \tau a(t)$ (otherwise), being τ the length of time-slot t;
- cost of energy $c(t)$ (in EUR/kWh), cost of CO_2 emissions $c_o(t)$ (in EUR/kg) and cost of energy in CO_2 emissions $e(t)$ (in kg/kWh);
- cost of demand without ESS and where PEV is not managed by EBR (in EUR), i.e., $c(t)f$, being f the flat tariff;
- cost of demand with ESS (in EUR), i.e., $[c(t) + c_o(t)e(t)]\tilde{d}(t)$.

Chapter 3

Conclusions

In this deliverable we described the third year versions of the prototypes for the SmartHG EBR HIAS. W.r.t. the second year version of the EBR prototype, we removed the login page (to ease reviewers access to the service), and we changed the downloaded software as described in the EBR design see (Deliverable D3.3.1).

3.1 Impact

The Web services of each Home Intelligent Automation Service (HIAS) are important in order to allow users (mainly residential users) to actually invoke such services in a clear and user-friendly way. On the other hand, the REpresentational State Transfer (RESTful) services behind such Web services are important to allow any Intelligent Automation Service (IAS) to directly and automatically communicate with other IASs. Thanks to this architecture, the SmartHG Platform may be directly used in an effective way by its final users, thus allowing all benefits described in Deliverable D3.3.1.

Chapter 4

List of Acronyms

Bibliography

[1] "Energy Bill Reduction (EBR) Web Service: mclabservices.di.uniroma1.it/ebr/," 2014.