D4.9 – Updated USM Extensions for Storage Systems

Deliverable ID	**D4.9**
Deliverable Title	**Updated USM Extensions for Storage Systems**
Work Package	**WP4**
Dissemination Level	**PUBLIC**
Version	**1.0**
Date	**31/08/2018**
Status	**final version**
Type	**Prototype**
Lead Editor	**UPB**
Main Contributors	**UNINOVA (Vasco Delgado-Gomes), UPB (Mihaela Albu, Mihai Sanduleac, Marta Sturzeanu), ISMB (Orlando Tovar), LIBAL (Yini Xu), ALPERIA (Karen Stocker)**

Published by the Storage4Grid Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731155.

Document History

Version	Date	Author(s)	Description
0.1	2018-07-12	UNINOVA	UNINOVA inputs
0.2	2018-07-13	UPB	UPB inputs
0.3	2018-07-13	UPB	UPB inputs
0.4	2018-08-20	UNINOVA LIBAL	Updated inputs and clarifications
0.5	2018-08-20	ALPERIA	Updated inputs
0.6	2018-08-22	UPB	Added Libal contribution in the the latest version
0.7	2018-08-27	ISMB	ISMB inputs
0.8	2018-08-29	UPB	Further improvements. Version ready for internal review.
0.9	2018-08-30	UNINOVA	Reviewed version by UNINOVA
0.91	2018-08-30	ISMB	Reviewed version by ISMB
1.0	2018-08-31	UPB	Final version, ready for submission to the EC

Internal Review History

Review Date	Reviewer	Summary of Comments
2018-08-30 (v0.9)	Vasco Delgado-Gomes (UNINOVA)	Approved: • Minor corrections • It needs update regarding outdated S4G interfaces and architecture.
2018-08-30 (v0.91)	Orlando Tovar (ISMB)	Approved: • Approved: General minor connections regarding the document shape, spelling mistakes, content on pictures.

Legal Notice

The research work leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731155 - Storage4Grid project. The information in this document is subject to change without notice. The Members of the Storage4Grid Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the Storage4Grid Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material. The European Union and the Innovation and Networks Executive Agency (INEA) are not responsible for any use that may be made of the information contained therein.

Table of Contents

Executive Summary

D4.9 describes the "Updated USM Extensions for Storage Systems" details of the prototypes, developed by the Storage4Grid (S4G) project. The deliverable describes the extension connectors which were developed in the Unbundled Smart Meter (USM) to support the S4G functionalities. Similarly, to other prototypes, this document provides minimal technical documentation necessary to understand the functionalities and structure of the USM prototype with its extension connectors needed to support the S4G system at the field level. The document is an update of D4.8 and will be revisited in the subsequent D4.10 deliverable (final USM extensions) at M33.

1 Introduction

D4.9 describes the "Updated USM Extensions for Storage Systems" prototypes, developed by the Storage4Grid project. Similarly, to other prototypes, this document provides minimal technical documentation necessary to understand the functionalities, structure and deployment instructions for the prototype of interest. More detailed information can be retrieved from related documents summarized in Section 1.3.

1.1 Reading Guide

In order to help readers in quickly finding their extensions concerns of interest in this document, Table 1 provides an overview of the key topics addressed by this and associated documents.

Table 1. S4G Architecture Reading Guide

Topic	Relevant Section
General information regarding the Unbundled Smart Meter, biling, instrumentation values and sub metering with different components	Section 2 - Unbundled Smart Meter Architecture. Prototype Overview
Presentation of main extensions used between the USM and other components	Section 3 – Smart Meter eXtensions – SMX. Prototype Overview
The way in which SMX communicates with high level application	Section 4 – SMX communication and information exchange with high level application. Prototype Overview
Minimal information regarding software and requirements	Section 5 – Software dependencies and requirements
Sub-components of the prototypes related to USM	Section 6 – Sub-components
How the installation has to be done and deployments instruction related to Technical Graphical Interface, Energy Router, Electrical Vehicle and Energy Storage Systems	Section 7 – Installation/Deployment instructions
Software dependencies and requirements	Section 8 – Software dependencies and requirements
Application Programming Interface References	Section 9 – API Reference

1.2 Scope

This prototype deliverable has been developed by Task T4.4 – "Unbundled Smart Meter Extensions for Storage Systems". Further updates of this prototypes are expected to be released in D4.10 (M33).

1.3 Related documents

ID	Title	Reference	Version	Date
D2.1	Initial Storage Scenarios and Use Cases	[2]	1.1	2017-06-08
D3.1	Initial S4G Components, Interfaces and Architecture Specification	[3]	1.0	2017-08-31
D2.2	Final Storage Scenarios and Use Cases	[4]	1.0	2018-07-31
D3.2	Updated S4G Components, Interfaces and Architecture Specification	[5]	1.0	2018-08-31

[SmxGuide]	SMX Make yourself guide	SmxGuide	1.6	2018-04-07
[SMXAddressesMapping]	Manual for retrieving DLMS addresses in new meters	[SMXAddressesMapping]	1.01	2018-03-29

2 Unbundled Smart Meter Architecture. Prototype Overview

The USM is composed from (i) a Smart Metrology Meter (SMM) i.e. a certified smart meter suitable to measure, using a trusted/certified method, various parameters of the electricity transferred in the point of meter connection, most important among them being the value of the electrical energy in a given time interval and (ii) a so-called smart meter extension (SMX), hosting the SMX Core and one or more SMX modules; SMX can be seen as a combination of modular software running on a dedicated small-form Personal Computer (PC) (namely the SMX hardware), which can host plug-in components providing added-value services.
The overall, high-level structure of prototype USM Architecture is summarized in Figure 1.

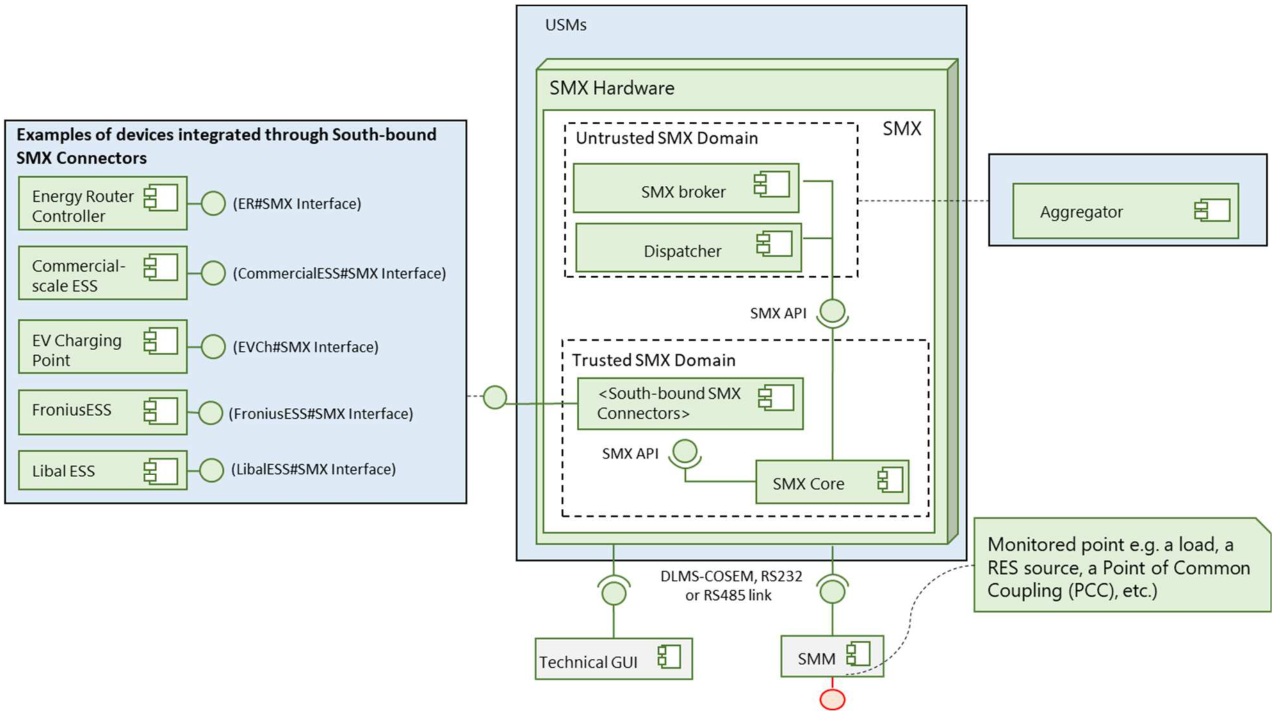


Figure 1 - The prototype structure, including an example of SMX instantiation.

The sub-components of the prototype are shortly summarized in the following sub-sections:

2.1 Smart Metrology Meter (SMM)

2.1.1 Billing values in SMM

Electricity meters are deployed for their main functionality: source of information for billing. This is usually made either by reading

- a meter index at a-priori specified intervals of time (e.g. each beginning of month), in case of single tariff or
- by reading meter indexes associated with different tariffs.

In case of smart energy meters used for active electrical energy (in opposition to the reactive electrical energy), the so-called “billing values” can be reported in digital form and read remotely. These values are usually made available in one or more of the following forms:

- Indexes of active energy: these are usually indexes which only increase their value until the index recirculates, meaning that it overpasses its maximum value, e.g. 999999.99

There can be separate indexes for each direction of the transfer of active energy during the Point of Common Coupling (PCC) where the meter is installed (to/from the customer installations), meaning separate IdxA+ and IdxA-, measured in [kWh] or [MWh]

- Indexes of reactive energies; these are also usually indexes for each direction of reactive energy, meaning separate IdxR+ and IdxR-, measured in [kvarh] or [Mvarh]
- Indexes associated with each tariff, in case of multiple tariffs;
- A list of these indexes stored at specified moments, usually at the beginning of a new month.

2.1.2 Instrumentation values in SMM

The following instrumentation values should be made available and, for S4G scope, should to be obtained directly (without additional calculations) from the SMM (also from those meters on the market which have some of the Smart Meter functionalities[li]) are:

- Voltage value (rms), on each phase, at moment k – U(k) for single-phase meters and U1(k), U2(k), U3(k) for three-phase meters, where U(k) or Ux(k) represents the rms value of the voltage available on the communication interface. To be noted that usually the time window Tw for calculating this value is not specified by the meter manufacturer. Usually these values are obtained from the secondary winding of instrument transformers and are expressed in volts [V]. Meters with indirect connection (voltage meter input connected to a voltage transformer VT and current meter input connected to a current transformer CT) may give the voltage level either as real measurements in the secondary circuits (specific to the standard 57 V range which correspond to 100 V phase-to-phase voltage of VT) or calculated as to correspond to the primary circuit values, i.e. secondary values multiplied by the VT ratio.
- Current value (rms) on each phase, at moment k – I(k) for single-phase meters and I1(k), I2(k), I3(k) for three-phase meters, where I(k) or Ix(k) represents the rms value of the current available on the communication interface.
- Active power value on each phase at moment k – P(k) for single-phase meters and P1(k), P2(k), P3(k) for three-phase meters. To be noted that usually the time window TW for calculating this value is not specified by the meter manufacturer. In case that the SMM transfer quantities to be calculated from other instrumentation values.
- Reactive power value on each phase at moment l– Q(k) for single-phase meters and Q1(k), Q2(k), Q3(k) for three phase meters. To be noted that usually the time window Tw for calculating this Q value is not specified by the meter manufacturer.
- Power factor m
- Frequency f of the network.

2.2 S4G Instrumentation Components

Similar with the main meter used for billing the (a.c.) electrical energy, presented as SMM, other components of the local system hosting energy storage need also specific instrumentation values. However, the additional instrumentation values may not be necessary enforced by metrology, as being sub-metering information, thus not relevant for the point of common coupling where the contractual relation between the local system and the DSO is sealed. It is of paramount importance to correlate the real time values (from the ER, for example) with quasi-real time values (from the storage system, for example) and instrumentation values (information averaged over Tw period).

2.2.1 Sub-metering local ESS

Similar with the SMM, local ESS needs its instrumentation values to be acquired by SMX through specific interface. A commercial or vendor specific ESS can be integrated by using either a special SW extension which can read the instrumentation values or another USM having as main role the provision of the electrical data. In S4G project are considered to be used USMs, which bring the same type of data to be read for all these points. The "Remote USM" are then the preferred devices to be connected to the south-bond connector in the main USM.

The following data are needed for the S4G project:

- The active power $P(t_k)$ on each phase of the meter connection; t_k is the reporting moment, with acceptable synchronization features; the highest reporting rate is given by $t_k - t_{k-1} = 1$ s; in S4G applications we might need to use different reporting rates (up to 15 min)
- The rms value of the voltage $U(t_k)$ in the metered point: t_k is the reporting moment, with acceptable synchronization features; the highest reporting rate is given by $t_k - t_{k-1} = 1$ s; in S4G applications we might need to use different reporting rates (up to 15 min)
- Active energy A+ and A- (in both directions): defined by multiplication of the active power with the reporting time: $A+=P+(t_k - t_{k-1})$

To be noted that the exact details of the extension will be presented in the next phase of development, namely in D4.10 [M33].

2.2.2 Sub-metering EV

EVs charging parameters need also to be read by the S4G system, to help the optimisation of using the storage resources. For this purpose, in S4G project are used USMs, bringing the same type of data to be read for all these points with the help of EV charging connectors. The extension is reading the "Remote USM" in the main USM.

The following data are needed for the project:

- The active power P(k) on each phase of the meter
- The voltage U(k) in the metering point
- Active energy A+ and A- (in both directions)

2.2.3 Sub-metering ER

The ER will measure its electrical data and make it available through the ER SMX South-bound adapter. The data is available in the ER is described in section 9.2. Please note that in cases where the ER energy needs to be billed (or any other reason) an USM will be considered.

2.2.4 Sub-metering PV

PV production is present in all pilot sites and is considered to be paired with the storage resources. PV are separate resources which are difficult to be interfaced with SMX, therefore the project is using an USM, as it is a common solution in previous cases.

The following data are needed for the project:

- The active power P(k) on each phase of the meter
- The voltage U(k) in the metered point
- Active energy A+ and A- (in both directions)

2.2.5 Sub-metering Fronius

The Fronius devices used in the project need to be also monitored also in terms of electrical data. Instead of USMs, the SMX extension connectors developed by Storage4Grid will provide instrumentation values directly from the measured internal data of the Fronius devices. The needed data is separated for each type of Fronius equipment.

2.2.5.1 Sub-metering Fronius Symo (hybrid)

The Fronius hybrid controls both local PV and storage resources. The following data are needed for the project:

- The active power P(k) on each phase of the AC interface towards the DSO network
- The voltage U(k) on each phase of the AC interface towards the DSO network
- Power Ppv, for the PV production
- Power Pess, for the storage part
- Active energy A+ and A- (in both directions)

For the situation that A+ and A- are not directly available, an integration module which uses P(k) will be implemented as extension connectors for USM, in order to obtain similar information.
The exact data available through the extension connectors will be based on the data availability from the Fronius hybrid inverter (FroniusHI).

2.2.5.2 Sub-metering Fronius for Storage

The Fronius inverter equipment controls the storage resources needed to provide the following data:

- The active power P(k) on each phase of the AC interface towards the DSO network
- The voltage U(k) on each phase of the AC interface towards the DSO network
- Active energy A+ and A- (in both directions)

For the situation that A+ and A- are not directly available, an integration module which uses P(k) will be implemented as an extension connector for USM, in order to obtain similar information.
The exact data available through the extension will be based on the data availability from the Fronius inverter.

3 Smart Meter eXtensions – SMX. Prototype Overview

3.1 Extensions for integration of local EV charging station (HLUC – 2)

A dedicated extension, namely the "EV Charging Point Connector", has been developed to integrate controllable charging points compatible with the OCPP open standard.
Such controllable charging points are normally deployed in public charging stations, but within the S4G project one of such charging points was deployed in a residential scenario.

In order to preserve operation of pre-existing EV SCADA systems, this extension works as a "transparent proxy", allowing in-bound connection from the EV SCADA systems, which are transparently forwarded to the charging point. The "transparent proxy" mode has been introduced because many commercial charging point implementation only allow one single in-bound connection.

The deployment schema for the extension is depicted in Figure 2. It has to be observed that picture does not represent the full list of components involved in this type of scenarios (e.g. the Data Broker, the DSF Data Warehouse (DSF-DWH), etc.), but it is only provided to highlight the main remote components to be deployed when the "EV charging Point Connector" extension is used.

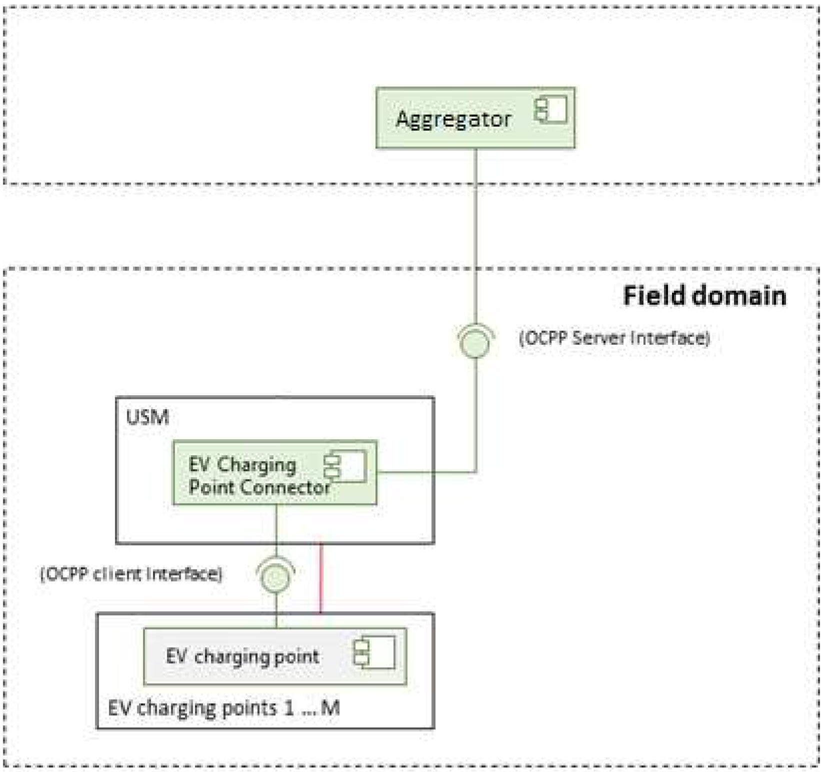


Figure 2 - EV Charging Point Connector

There are three interfaces designed for this extension which is named *EV charging point connector*. They are OCPP client interface, OCPP server interface and SMX#Aggregator Interface as shown in Figure 2 - EV Charging Point Connector. For this version, only OCPP client interface and OCPP server interface are defined. SMX#Aggregator Interface will be defined in the next version.

3.2 Extensions regarding the inter-connection with GUI for technical users

During the phase 2, test sites development and installation of the SMXs at the partner's premises, it was identified the need of a technical interface. In the testing stage, obtaining an instantaneous reaction from the analysed connection was important to identify possible errors. Considering this, a Technical Graphical User Interface was realized as an additional interface. This includes different tabs which can ease the monitoring of a correct functioning of the SMX and the correct communication between SMM and SMX.
An important tab of the interface is named *LESSAg*. This one is associated to storage systems, and enables the visualization of a planning for the electrical energy storage.

Also, for technical objectives an extension related to Raspberry Pi (SMX) health monitoring was developed. It can be installed on SMX and through an WEB GUI the status and other relevant information will be exposed for all the SMX installed and alive during the project. This tool run twice a day and control the amount of free disk space, raising an alert via email if the free percentage is less than 15% of the overall memory etc.

3.3 Extensions for integration with Grid-side ESS (HLUC – 3)

The integration of the grid-side ESS is realized through Aggregator component placed in Communication Layer. GESSCon communicates with the Aggregator and then the Aggregator communicates with SMX and LESSAg. More information regarding these interactions can be found in deliverable D3.2 [M21].

3.4 Extensions for integration with Residential ESS (HLUC – 2 and HLUC – 3)

The integration of the residential ESS is realized through Aggregator component placed in Communication Layer. PROFESS offers a flexible optimization setting environment for controlling ESS. More information regarding these interactions can be found in deliverable D3.2 [M21].

3.5 Extensions for integration with ER (HLUC – 1)

The extension to connect USM with the ER was designed to allow the monitor and control of the ER. More information about the API is described in is described in section 9.2. The deployment schema for the extension is depicted in Figure 3.

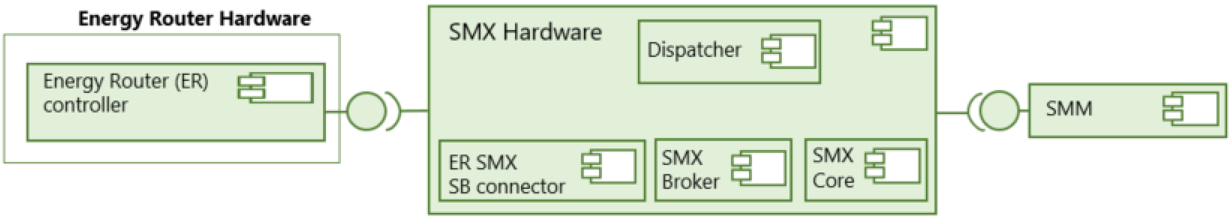


Figure 3 - ER extension

The ER Connector is implemented in the trusted SMX domain and the communication is made through the real-time database of SMXCore.

4 SMX communication and information exchange with high level application. Prototype Overview

The communication with the GESSCon and GEVChCon components will use MQTT messaging to exchange data between SMX and Aggregator, followed by OGC based communication to the high level, as presented in Figure 4.

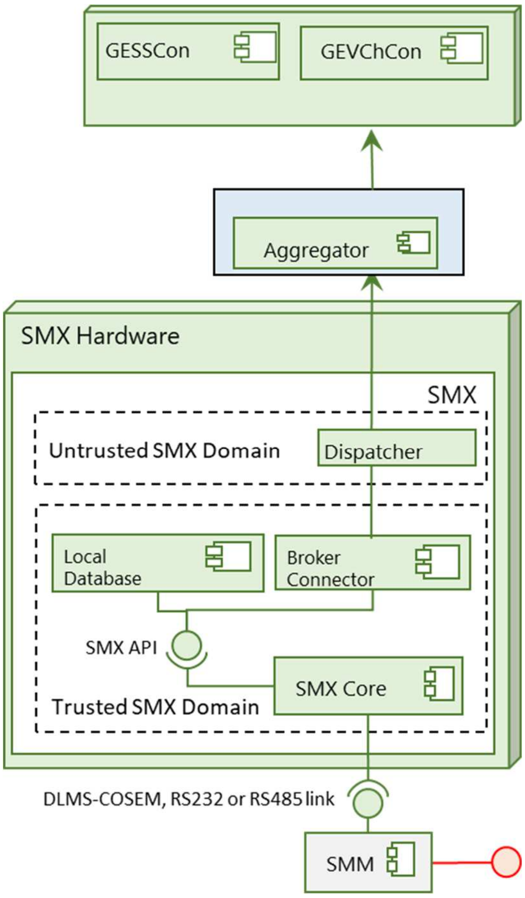


Figure 4 - High level applications connectors

The information exchange with the trusted domain is based on precise RBAC configuration, allowing that only necessary data is sent to the agents running in the untrusted domain.

5 Software dependencies and requirements

5.1 Minimal requirements for SMX

SMX is a Linux platform that runs on a compact Single Board Computer (SBC). The Raspberry Pi 3 Model B+ SBC has been chosen due to its quad-core architecture running at 1.4 GHz, its memory capacity of 1 GB of RAM and its storage capacity of up to 32 GB on SD-Card. In the S4G project, it is considered as sufficient a 16 GB SD-Card, which allows the installation of the Linux operation system and of the project related applications, while still allowing the recording of a high-resolution log of different variables.

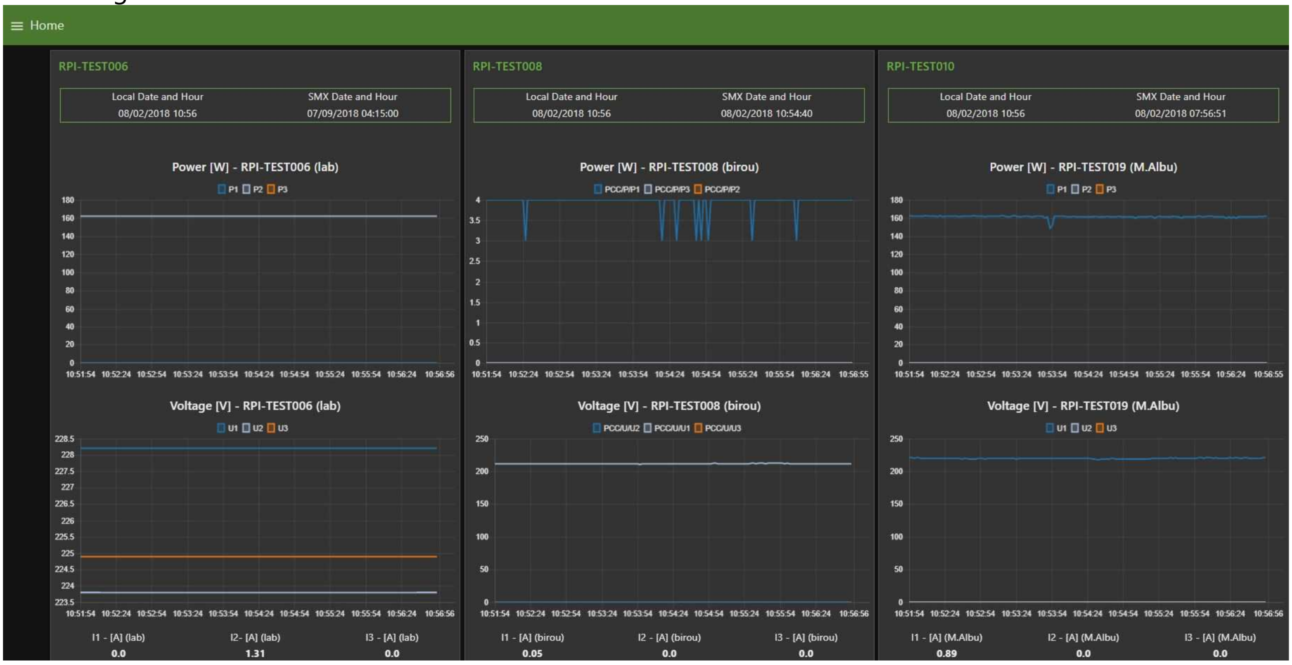
5.2 Advanced requirements for SMX

The advanced requirements for SMX consist in the capabilities of the LESSAg/PROFESS/PROFEV component, a software component in charge of running site-wise ESS control algorithms. It receives in quasi-real-time all available information from local devices: ER which will transmit information on elements connected to its available ports: (the ESS system, PV, DCbus bar), the (a.c.) load-side energy meters, and the (a.c.) PCC with the DSO.

6 Sub-components

6.1 Local Technical GU Interface SMX South-bound connector

This interface is able to show in real time on an associated web link data recorded by SMX at a certain measurement point by subscription to different topic. The latest version of the Local Technical Interface can be seen in Figure 5


≡ Home
Power [W] - RPI-TEST006 (lab)
P1 P2 P3
Power [W] - RPI-TEST008 (birou)
PCC/P/P1 PCC/P/P3 PCC/P/P2
Power [W] - RPI-TEST019 (M.Albu)
P1 P2 P3
Voltage [V] - RPI-TEST006 (lab)
U1 U2 U3
Voltage [V] - RPI-TEST008 (birou)
PCC/U/U2 PCC/U/U1 PCC/U/U3
Voltage [V] - RPI-TEST019 (M.Albu)
U1 U2 U3
10:51:54 10:52:24 10:52:54 10:53:24 10:53:54 10:54:24 10:54:54 10:55:24 10:55:54 10:56:24 10:56:56

RPI-TEST006

Local Date and Hour	SMX Date and Hour
08/02/2018 10:56	07/09/2018 04:15:00

I1 - [A] (lab)	I2- [A] (lab)	I3 - [A] (lab)
0.0	1.31	0.0

RPI-TEST008

Local Date and Hour	SMX Date and Hour
08/02/2018 10:56	08/02/2018 10:54:40

I1 - [A] (birou)	I2 - [A] (birou)	I3 - [A] (birou)
0.05	0.0	0.0

RPI-TEST010

Local Date and Hour	SMX Date and Hour
08/02/2018 10:56	08/02/2018 07:56:51

I1 - [A] (M.Albu)	I2 - [A] (M.Albu)	I3 - [A] (M.Albu)
0.89	0.0	0.0

Figure 5 - Local Technical Interface

6.2 Energy Router SMX South-bound connector

This connector receives commands (set-points) from the PROFESS and publishes real-time data in the SMX Event broker (SMX-EB). The Connector receives the set-points and forward them to the ER Controller, through the ERController#ERConnector interface using the IEC61850-90-7 data model. More detailed information about this interaction and interface can be found in Section 9.

6.3 EV Charger SMX South-bound connector

This connector uses OCPP to communicate with a specific electric vehicle charging point. It receives commands (set-points) from the PROFEV, trough MQTT, and publishes data in the SMX Event broker attached to the specific EV charging point. The connector receives the set-points and forward them to the EV charger, through the SMX#EV Charger interface. Besides, the connector reads data from EV charger about the charging point status (i.e. EV charging point status, charging point time) as well as measured electrical parameters. Retrieved data from EV charger are used by components in the *Edge Layer* such as the *Residential GUI*. A detailed description of this connector will be provided in the next version of this document.

6.4 ESS (Modbus) SMX South-bound connector

This connector receives commands (set-points) from the PROFESS and publishes real-time data in the SMX Event broker (SMX-EB). The Connector receives the set-points and forward them to the ESS, through the ESS#ESSConnector interface using the Modbus protocol. More detailed information about this interaction and interface can be found in Section 5.

6.5 Weather Forecast and Load Prediction SMX connector

The connector for weather forecast – solar radiation connector developed in S4G project makes uses of the API provided by PVGIS. Through this API, the connector receives information of solar radiation and PV generation of a defined place. More details are information can be found in deliverable D5.4. – Updated DSF Connectors.
Local predictions will be made through publishing topics in the SMX-Broker from the PROFESS.

7 Installation/Deployment instructions

7.1 Local Technical GU Interface SMX South-bound connector

The graphical interface was developed using an open-source program developed for Linux distributions called *Node-Red*[ii]. It allows a quick, easy and convenient configuration of an interface by interconnecting some nodes specific to Internet-based Things (IoT) platforms. The basic operating system for *Raspberry Pi* is *Raspbian OS*, and it comes with a pre-installed version of the *Node-Red*. However, it is recommended to update the original version, and an additional module called Node-Red-Dashboard[iii] was used.
In order to install an interface on the SMX following steps are required:

- Connect to your SMX (Raspberry Pi) using *putty.exe* and update node-red using the following command inside putty: <update-nodejs-and-nodered> and then start node-red as a system service using: <sudo systemctl enable nodered.service>. First time it is required to start it manually, so just type: <node-red-start>
- In your web browser, navigate to the node-red page using the SMX IP - you can find out which IP your SMX has using: <hostname –I> (*capital i, not small letter L)* and install node-red-dashboard by clicking on the Menu button (found in the upper-right corner of the interface) >> Manage pallette >> Install nodes >> type in: node-red-dashboard >> install it.
- Import from Clipboard the code for the desired interface (Menu >> Import >> Clipboard >> paste the received code there and click on "import on current flow") or, another option is to connect to your SMX through WinSCP (or similar software), navigate to */home/pi/.node-red/* and delete the file which is named *flows your-SMX-name.json* (if existing), copy the received *.json* file and rename it to have the same name pattern as original deleted one (*flows_your-SMX-name.json*). E.g.: *flows_RPI-TEST005.json.*
- Restart node-red using the following sequence in command line (putty):<node-red-stop>, <node-red-start>.
- Open your interface at hostname:1880/ui. E.g.: 192.168.1.4:1880/ui

7.2 Energy Router SMX South-bound connector

The ER SMX SB connector was implemented using a Raspberry PI 3 Model B[IIiv] (RPI3). The SMX image needs to be installed on the RPI3 using the instructions document available to the consortium [SmxGuide]. The Stretch Raspbian version was used.
The developed software used in this prototype can be found in the Storage4Grid Git official repository as SB-SMX-ER-connector.
Copy the 'AuxFiles_v2.0' folder to the SMX and paste it at /home/pi/S4G/. Rename the folder to 'SB_SMX_ErConnector'. Edit the 'ErConnectorConfig.properties' file to comply with the system. To automatically start the ER Connector when the SMX starts, use the instructions available for the consortium and the 'ErConnector.service' file. Please do not edit the xml file as is a key file for the application.

7.3 EV Charger SMX South-bound connector

The EV Charger SMX South-bound connector is a python-based software component. The python dependencies are managed by Miniconda3. In order to install this connector, the Miniconda3 dependencies should be installed by running the "*install.sh*" script. Besides, a dedicated service should be created following the systemD specifications. Further installation details will be provided in the next version of this deliverable.

7.4 ESS (Modbus) SMX South-bound connector

The Modbus SMX SB connector was implemented using a RPI3. The SMX image needs to be installed on the RPI3 using the instructions document available to the consortium [SmxGuide]. The Stretch Raspbian version was used.

The developed software used in this prototype can be found in the Storage4Grid Git official repository as SB-SMX-Modbus-connector.

Copy the 'AuxFiles_v1.0' folder to the SMX and paste it at /home/pi/S4G/. Rename the folder to 'SB_SMX_ModbusConnector'. Edit the 'ModbusConnectorConfig.properties' file to comply with the system. To automatically start the Modbus Connector when the SMX starts, use the instructions available for the consortium and the 'ModbusConnector.service' file. Please do not edit the xml file as is a key file for the application.

8 Software dependencies and requirements

Table 2. Software Dependencies

Dependency	License	Role
Apache httpd, version 2.4.26	Apache version 2.0	Used to register and expose web-pages for the GUI, as well as proxying content over SSL channels
OpenIEC61850, version 1.5.0	Apache License 2.0	This software is used to exchange data between the ER Connector and the ER using the IEC61850 protocol.
Jamod, version	Apache version 2.0	This software is used to exchange data between the ESS Connector and the ESS using the Modbus protocol.
Java-OCA-OCPP[v]	MIT	A client and server library of Open Charge-Point Protocol from openchargealliance.org. Used in the "Extensions for integration of local EV charging station"
SMXCore	GPL	Open source Core application running in the trusted domain, implementing the real-time database, the communication with SMM and with all extensions running around SMXCore, by using also a RBAC system to preserve security and privacy for the used data
Node-RED	No license needed	Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

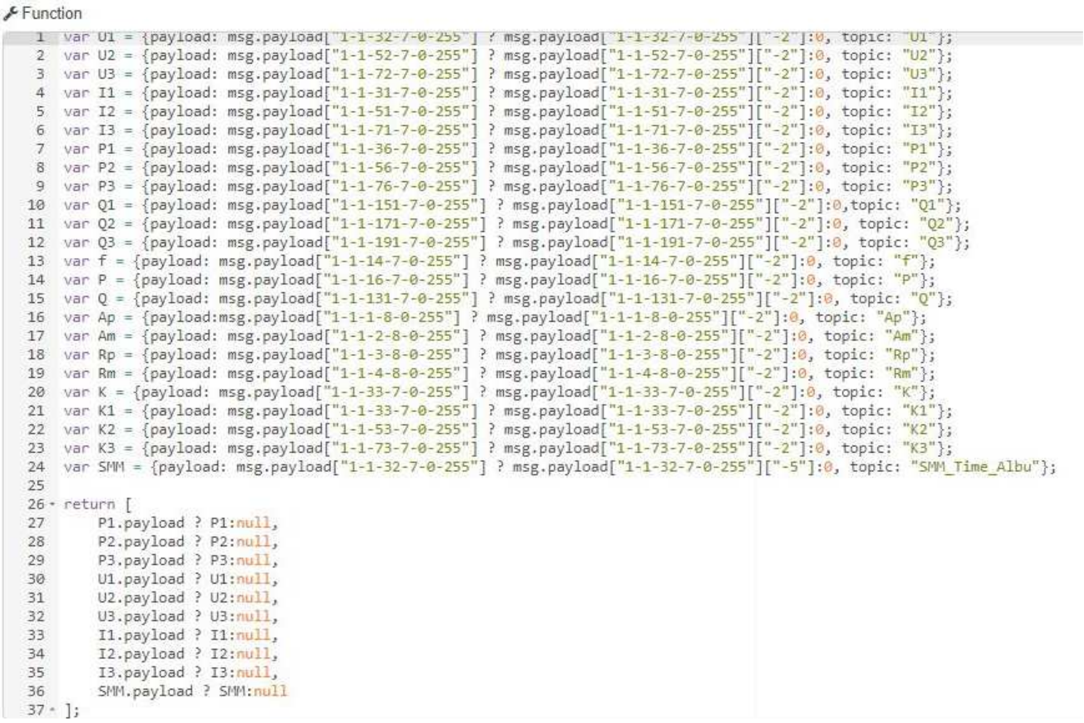
9 API Reference

9.1 Local Technical GUI Interface SMX South-bound connector

Table 3 Parameters recorded by SMX and presented on the interface

Readed and recorded data	Parameter on interface
SySDate: The date of the system	Yes
SySTime: The time of the system	Yes
Q: Reactive Power [kvar];	No
Q1: Reactive Power on phase 1 [kvar];	No
Q2: Reactive Power on phase 2 [kvar];	No
Q3: Reactive Power on phase 3 [kvar];	No
P: Active Power [kW];	Yes
P1: Active power on phase 1 [kW];	Yes
P2: Active power on phase 2 [kW];	Yes
P3: Active power on phase 3 [kW];	Yes
U1: Voltage on phase 1 [V];	Yes
U2: Voltage on phase 2 [V];	Yes
U3: Voltage on phase 3 [V];	Yes
I1: Current on phase 1 [A];	Yes
I2: Current on phase 2 [A];	Yes
I3: Current on phase 3 [A];	Yes
K1: Power factor on phase 1 [-];	Not yet
K2: Power factor on phase 2 [-];	Not yet
K3: Power factor on phase 3 [-];	Not yet
Ap: Consumed active el. energy [kWh];	Yes
Am: Produced active el.energy [kWh];	Yes
Rp: Consumed reactive el. energy [kvarh];	Yes
Rm: Produced reactive el. energy [kvarh];	Yes
f: Frequency [Hz];	Yes
SySCpuLoad: Level of charge of the system process unit	Not yet

The Node-RED interface receives by subscribing to different topics as mqtt messages presented in Figure 6 and Figure 7



```
var U1 = {payload: msg.payload["1-1-32-7-0-255"] ? msg.payload["1-1-32-7-0-255"]["-2"]:0, topic: "U1"};
var U2 = {payload: msg.payload["1-1-52-7-0-255"] ? msg.payload["1-1-52-7-0-255"]["-2"]:0, topic: "U2"};
var U3 = {payload: msg.payload["1-1-72-7-0-255"] ? msg.payload["1-1-72-7-0-255"]["-2"]:0, topic: "U3"};
var I1 = {payload: msg.payload["1-1-31-7-0-255"] ? msg.payload["1-1-31-7-0-255"]["-2"]:0, topic: "I1"};
var I2 = {payload: msg.payload["1-1-51-7-0-255"] ? msg.payload["1-1-51-7-0-255"]["-2"]:0, topic: "I2"};
var I3 = {payload: msg.payload["1-1-71-7-0-255"] ? msg.payload["1-1-71-7-0-255"]["-2"]:0, topic: "I3"};
var P1 = {payload: msg.payload["1-1-36-7-0-255"] ? msg.payload["1-1-36-7-0-255"]["-2"]:0, topic: "P1"};
var P2 = {payload: msg.payload["1-1-56-7-0-255"] ? msg.payload["1-1-56-7-0-255"]["-2"]:0, topic: "P2"};
var P3 = {payload: msg.payload["1-1-76-7-0-255"] ? msg.payload["1-1-76-7-0-255"]["-2"]:0, topic: "P3"};
var Q1 = {payload: msg.payload["1-1-151-7-0-255"] ? msg.payload["1-1-151-7-0-255"]["-2"]:0,topic: "Q1"};
var Q2 = {payload: msg.payload["1-1-171-7-0-255"] ? msg.payload["1-1-171-7-0-255"]["-2"]:0, topic: "Q2"};
var Q3 = {payload: msg.payload["1-1-191-7-0-255"] ? msg.payload["1-1-191-7-0-255"]["-2"]:0, topic: "Q3"};
var f = {payload: msg.payload["1-1-14-7-0-255"] ? msg.payload["1-1-14-7-0-255"]["-2"]:0, topic: "f"};
var P = {payload: msg.payload["1-1-16-7-0-255"] ? msg.payload["1-1-16-7-0-255"]["-2"]:0, topic: "P"};
var Q = {payload: msg.payload["1-1-131-7-0-255"] ? msg.payload["1-1-131-7-0-255"]["-2"]:0, topic: "Q"};
var Ap = {payload:msg.payload["1-1-1-8-0-255"] ? msg.payload["1-1-1-8-0-255"]["-2"]:0, topic: "Ap"};
var Am = {payload: msg.payload["1-1-2-8-0-255"] ? msg.payload["1-1-2-8-0-255"]["-2"]:0, topic: "Am"};
var Rp = {payload: msg.payload["1-1-3-8-0-255"] ? msg.payload["1-1-3-8-0-255"]["-2"]:0, topic: "Rp"};
var Rm = {payload: msg.payload["1-1-4-8-0-255"] ? msg.payload["1-1-4-8-0-255"]["-2"]:0, topic: "Rm"};
var K = {payload: msg.payload["1-1-33-7-0-255"] ? msg.payload["1-1-33-7-0-255"]["-2"]:0, topic: "K"};
var K1 = {payload: msg.payload["1-1-33-7-0-255"] ? msg.payload["1-1-33-7-0-255"]["-2"]:0, topic: "K1"};
var K2 = {payload: msg.payload["1-1-53-7-0-255"] ? msg.payload["1-1-53-7-0-255"]["-2"]:0, topic: "K2"};
var K3 = {payload: msg.payload["1-1-73-7-0-255"] ? msg.payload["1-1-73-7-0-255"]["-2"]:0, topic: "K3"};
var SMM = {payload: msg.payload["1-1-32-7-0-255"] ? msg.payload["1-1-32-7-0-255"]["-5"]:0, topic: "SMM_Time_Albu"};

return [
    P1.payload ? P1:null,
    P2.payload ? P2:null,
    P3.payload ? P3:null,
    U1.payload ? U1:null,
    U2.payload ? U2:null,
    U3.payload ? U3:null,
    I1.payload ? I1:null,
    I2.payload ? I2:null,
    I3.payload ? I3:null,
    SMM.payload ? SMM:null
];
```

Figure 6 - A Node-RED function description

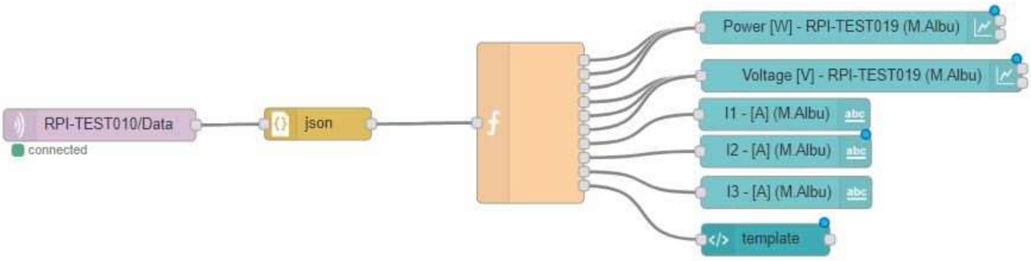
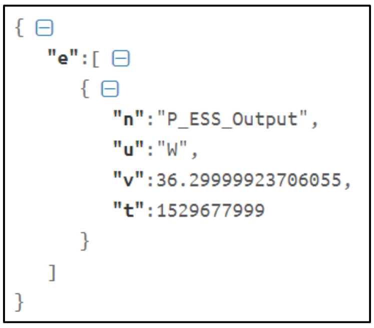


Figure 7 - Node-RED structure for an SMX through .json messages

9.2 Energy Router SMX South-bound connector API

The PROFESS (or other components, e.g. PROFEV, LESSAg) can send set-points to both ERs using the MQTT topics described in Table 43. Table 54 shows the three-phase ER set-points while Table 65 shows the single-phase ER set-points. To define a set-point, the PROFESS should publish in the corresponding topic using the template described in Figure 8.



```
{ ⊟
   "e":[ ⊟
      { ⊟
         "n":"P_ESS_Output",
         "u":"W",
         "v":36.29999923706055,
         "t":1529677999
      }
   ]
}
```

Figure 8 - SENML JSON message template.

“n” can have different values, according to the PROFESS optimisation output, namely:

- P_ESS_Output
- P_Grid_Export_Output
- Q_Grid_Export_Output
- P_Grid_Import_Output
- Q_Grid_Import_Output
- Q_PV_Output
- P_PV_Output

Table 4. Topics to define ER set-points.

Parameter	Units	Topic
DERStart	1 = Standby 2 = PV On 3 = PV Off 4 = Load balancing 5 = Pref Control	/PROFESS/SMX/EnergyRouterInverter/DRCC1.DERStr.ctlNum
DERStop	0 = Converter is switched off.	/PROFESS/SMX/EnergyRouterInverter/DRCC1.DERStop.ctlNum
PBatRef	W	/PROFESS/SMX/EnergyRouterInverter/ZBTC1.BatChaPwr.setMag.f
PPvRef	W	/PROFESS/SMX/EnergyRouterPV/MMDC1.Watt.subMag.f

Table 5. Topics to define three-phase ER set-points.

Parameter	Units	Topic
PGridRef	W	/PROFESS/SMX/EnergyRouterInverter/MMXU1.TotW.subMag.f
PaGridRef	W	/PROFESS/SMX/EnergyRouterInverter/MMXU1.W.phsA.subCVal.mag.f
PbGridRef	W	/PROFESS/SMX/EnergyRouterInverter/MMXU1.W.phsB.subCVal.mag.f
PcGridRef	W	/PROFESS/SMX/EnergyRouterInverter/MMXU1.W.phsC.subCVal.mag.f
QGridRef	VAr	/PROFESS/SMX/EnergyRouterInverter/MMXU1.TotVAr.subMag.f
QaGridRef	VAr	/PROFESS/SMX/EnergyRouterInverter/MMXU1.VAr.phsA.subCVal.mag.f

QbGridRef	VAr	/PROFESS/SMX/EnergyRouterInverter/MMXU1.VAr.phsB.subCVal.mag.f
QcGridRef	VAr	/PROFESS/SMX/EnergyRouterInverter/MMXU1.VAr.phsC.subCVal.mag.f

Table 6. Topics to define single-phase ER set-points.

Parameter	Units	Topic
PGridRef	W	/PROFESS/SMX/EnergyRouterInverter/MMXN1.Watt.subMag.f
QGridRef	VAr	/PROFESS/SMX/EnergyRouterInverter/MMXN1.VolAmpr.subMag.f

The same template defined in Figure 8 can be used to read real-time values in the topics defined in Table 76. Table 87 shows the three-phase ER topics, while Table 98 shows the single-phase ER topics.

Table 7. Topics to receive ER real-time values.

Parameter	Units	Topic
SoC	%	/ER/SMX/EnergyRouterInverter/ZBAT1.VolChgRte.instMag.f
PBat	W	/ER/SMX/EnergyRouterInverter/ZBTC1.BatChaPwr.setMag.f
Ppv	W	/ER/SMX/EnergyRouterPV/MMDC1.Watt.instMag.f

Table 8. Topics to receive three-phase ER real-time values.

Parameter	Units	Topic
PGrid	W	/ER/SMX/EnergyRouterInverter/MMXU1.TotW.instMag.f
PaGrid	W	/ER/SMX/EnergyRouterInverter/MMXU1.W.phsA.instCVal.mag.f
PbGrid	W	/ER/SMX/EnergyRouterInverter/MMXU1.W.phsB.instCVal.mag.f
PcGrid	W	/ER/SMX/EnergyRouterInverter/MMXU1.W.phsC.instCVal.mag.f
QGrid	Var	/ER/SMX/EnergyRouterInverter/MMXU1.TotVAr.instMag.f
QaGrid	Var	/ER/SMX/EnergyRouterInverter/MMXU1.VAr.phsA.instCVal.mag.f
QbGrid	Var	/ER/SMX/EnergyRouterInverter/MMXU1.VAr.phsB.instCVal.mag.f
QcGrid	VAr	/ER/SMX/EnergyRouterInverter/MMXU1.VAr.phsC.instCVal.mag.f

Table 9. Topics to receive single-phase ER real-time values.

Parameter	Units	Topic
PGrid	W	/ER/SMX/EnergyRouterInverter/MXN1.Watt.instMag.f

QGrid	VAr	/ER/SMX/EnergyRouterInverter/MMXN1.VolAmpr.instMag.f

9.3 EV Charger SMX South-bound connector

The following information, in Table 9 and Table 10, may change since the component is underdevelopment.

Table 10. Topics to receive data from EV Charger

Parameter	Units	Topic
EV_status	1= EV status 2= charging point time (s)	/EV/SMX/status
EV_measurement	1= Power(kW) 2= Voltage (V) 3= State of charge (SoC) % 4=Enable/Disable	/EV/SMX/measurement

Table 11. Topics to send data to EV Charger

Parameter	Units	Topic
EV_command	1= load profile (kW) 2=Enable/Disable	/SMX/EV/command

9.4 ESS (Modbus) SMX South-bound connector

The PROFESS (or other components, e.g. PROFEV, LESSAg) can send set-points to the ESS using the template described in Figure 8. The message should be sent to the MQTT topics described in the following format: "/PROFESS/SMX/" +register number. For instance, if a set-point is to be sent to the register 1234, the message should be sent to the MQTT topic "/PROFESS/SMX/1234".

Similar method should be used to read real-time values from the ESS using the Modbus protocol. Subscribing to the topic "/ESS/SMX/2345" it receives in near-real time the value of the 2345 register.

The connector has a configuration file were the topics can be changed if needed. Also, the registers description is made using a xml file, enabling the use of this connector in several devices, changing the Modbus registers mapping.

9.5 Residential GUI SMX South-bound connector

The Residential GUI SMX South-bound connector is referred to the *Residential GUI (back-end)*. This connector is implemented by the *Aggregator* component. Further details are described in D6.8.

Table 12. Topics to send data to the Residential GUI

Parameter	Units	Topic
Residential GUI data	*<SENSOR_ID> KEY1=<value1>,KEY2=<value2>,...,KEYn=<valuen>*	RESIDENTIAL/GUI

	datatype=<dataType> *<timestamp>*	

In order to show time-frame data, the *Residential GUI* relies on the *Aggregator DWH* which is based on TICK influx. The APIs to query data form the *Aggregator DWH* are based on influxDB.
This is an example of query with curl to a specific SMX from a time window: *"curl -G 'http://<url>/query?db=S4G-DWH-AGGREGATOR' --data-urlencode 'q=select * from "S4G-GW-EDYNA-0014" where time > 2018-06-01 00:00:00'' AND time < 2018-08-29 23:59:59'''*.

10 Conclusions

This deliverable presents the "Updated USM Extensions for Storage Systems" prototype, developed by the Storage4Grid project. Similarly, to other prototypes, this document provides minimal technical documentation necessary to understand the functionalities, structure and deployment instructions for the prototype of interest. The document presents the main extensions needed in the SMX for implementing a complete chain of functionalities to support efficient and optimized used of storage resources and other important components involved in the support process.
More detailed information regarding the implementation of the extensions will be presented in the following deliverable D4.10.

Acronyms

Acronym	Explanation
API	Application Programming Interface
DSO	Distribution System Operator
ER	Energy Router
ESS	Energy Storage System
EV	Electrical Vehicle
GUI	Graphical user interface
JSON	JavaScript Object Notation
MQTT	Message Queue Telemetry Transport
PROFESS	Professional Realtime Optimization Framework for Energy Storage Systems
PROFEV	Professional Realtime Optimization Framework for Electric Vehicles
RPI3	Raspberry PI 3 Model B
OCPP	Open Charge Point Protocol
PCC	Point of common coupling
PV	Photovoltaic
RBAC	Role based access control
RMS	Root mean square
SBC	Single Board Computer
SCADA	Supervisory control and data acquisition
S4G	Storage4Grid
SMM	Smart Metrology Meter
SMX	Smart Meter eXtension
XML	Extensible Markup Language
SW	Software
USM	Unbundled Smart Meter

List of figures

List of tables

References

[i] Mihaela Albu, Mihai Sănduleac, Carmen Stănescu, 2016, "Syncretic use of smart meters for Power Quality monitoring in emerging networks", IEEE Transactions on Smart Grid, Volume: PP, Issue: 99

[ii] "Node-RED" [online]. Available: https://nodered.org/ [Accessed 05-12-2017].

[iii] "node-red-dashboard - Node-RED" [online]. Available: https://flows.nodered.org/node/node-red-dashboard. [Accessed: 05-12-2017]

[iv] Raspberry PI 3 Model B, https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ [Accessed: 31 July 2017].

[v] https://github.com/ChargeTimeEU/Java-OCA-OCPP