*local **E**lectricity retail **M**arkets for **P**rosumer smart grid p**OWER** services*

Deliverable nº:	**D4.2**
Deliverable name:	**Functional and technical documentation of relevant API-functions**
Version:	**1.0**
Release date:	**09/03/2016**
Dissemination level:	**PU**
Status:	**Draft**
Author:	Schneider - Cristóbal Cordobés
Contributors:	Berraquero eSmart - Bjørn Asvard Olsen
	CITCEA - Pau Lloret Gallego

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No 646476.

Document history:

Version	Date of issue	Content and changes	Edited by
0.1	11/11/2015	First draft version	Cristóbal Cordobés
0.2	25/02/16	Major changes	Bjørn Asvard Olsen
0.3	01/03/16	Completive changes	Cristóbal Cordobés
0.4	01/03/16	Prosumer info. added	Bjørn Asvard Olsen
0.5	02/03/16	Document change	Cristóbal Cordobés / Pau Lloret
0.6	04/03/16	Final draft	Bjørn Asvard Olsen / Cristóbal Cordobés

Peer reviewed by:

Partner	Contributor
UPC	Pau LLoret
eSmart Systems	Jo Morten Sletner

Deliverable beneficiaries:

WP / Task	Responsible
WP3	UPC
WP5	eSmart Systems
WP7	Schneider Electric

Table of contents

Executive summary 6

1 General Architecture 7

2 Interfaces for communication integration 9

2.1 DER - SCADA interface 9

2.1.1 Architecture Data Management 10

2.1.2 EV charging station Modbus mapping 11

2.1.3 Inverter for generation Modbus mapping 13

2.2 SCADA – SESP interface (Azure Event Hub Send Event) 15

2.2.1 The API description of the Azure event hub REST API is as follows:Request 15

2.2.2 Response 16

2.3 Prosumer – SESP interface (UPC & eSmart) 16

3 Use Cases API´s 17

3.1 Electric Vehicle Charging Station (EV_CS) 17

3.1.1 Messages from EV_CS to SESP 17

3.1.2 Messages from SESP to EV_CS 21

3.2 Generation Resources 24

3.2.1 Messages from generation to SESP 24

3.2.2 Messages from SESP to Generation 27

3.3 Storage 31

3.3.1 Messages from Storage to SESP 31

3.3.2 Messages from SESP to Storage 34

3.4 Load Resources 37

3.4.1 Messages from load resources to SESP 37

3.4.2 Messages from SESP to Load resources 40

3.5 Prosumers 44

3.5.1 Messages from local controller to SESP 44
3.5.2 Messages from SESP to local controller 46
3.6 Conclusions 46
3.7 References 47

Abbreviations and Acronyms

Acronym	**Description**
DER	Distributed Energy Resources
DR	Demand Response
DSO	Distributed System Operator
EV	Electrical Vehicle
IaaS	Infrastructure as a Service
iEMS	intelligent Energy Management System
IT	Information Technology
LC	Local Controller
PaaS	Platform as a Service
SCADA	Supervisory Control and Data Acquisition
SESP	Smart Energy Service Provider
SM	Smart Meter

Executive summary

The aim of this document is to describe the APIs and message formats that needs to be implemented together with software delivery “Cloud based control system for SESP, phase 1”. This constitutes the first version (SESP v1.0) of the system that will be the primary operational tool for the EMPOWER Smart Energy Service Provider (SESP).

The original conceptual design of the EMPOWER SESP platform includes three communicating entities, namely the metering cloud, the control cloud, and the market cloud (local market place).

The proposed IT architecture for the SESP platform supports all of these three elements in a unified technical solution that minimise communication and data duplication overheads and at the same time improve flexibility and maintainability of the final system.

The required communication paths for the “Cloud based control system for SESP, phase 1” will be:

- Distributed energy resources (DER) to SESP
    - DER’s such as generators, storage, electrical vehicle charging station will send technical information to SESP in a periodically manner.
    - A SCADA system will be used as a gateway between all DER’s and SESP.
- SESP to Distributed energy resources (DER)
    - The SESP shall have the possibility to send messages to the DER’s. Types of messages are either a request for variables (time series for a given property at a given DER), or control signals for demand response.
- Local controller (gateway) to SESP
    - The local controllers will communicate directly with SESP by using the SESP IoT service (Azure event hub).
- SESP to local controller
    - Any control signals from SESP to load resources controlled by the local controller will be sent directly from SESP to the local controller.

1 General Architecture

The IT architecture used for the EMPOWER SESP system is based on Microsoft Azure cloud computing big data PaaS solutions.

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort.

Azure is a cloud computing platform and infrastructure, created by Microsoft, for building, deploying and managing applications and services through a global network of managed datacenters. It provides both PaaS and IaaS services and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems.

Platform as a service (PaaS) is a category of cloud computing services that provides a platform allowing customers to develop, run, and manage web applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an application.

The main elements of the architecture are data storage components (blue cylinders), data transfer and processing components (orange cylinders) and services (hexagons).

Many different types of data are going to be stored, from dynamic metering data to static configuration data and customer data, and the architecture provides a number of dedicated storage solutions.

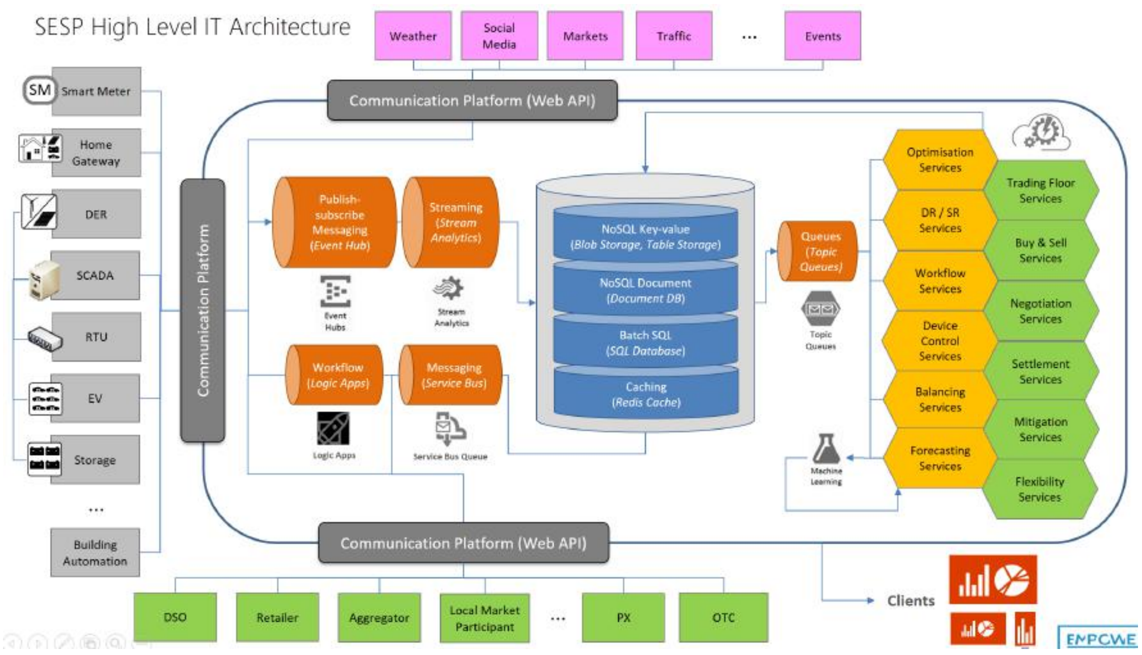


Figure 1: SESP High Level IT Architecture

Two types of communication channels will be used between SCADA and SESP:

- Azure Event Hub will be used for all messages from SCADA to SESP
- A SCADA web service will be utilized by SESP for sending control signals

1.1.1.1 Azure Event Hubs

Azure Event Hub will be the communication channel for all messages from SCADA to SESP (such as real-time values from the various DER's).

Azure Event Hub is a highly scalable data ingress service that can handle millions of events per second. This enables processing and analysis of massive amounts of data produced by connected devices and applications. Event Hubs acts as the "front door" for an event pipeline, and once data is collected into an event hub, it can be transformed and stored using any real-time analytics provider or batching/storage adapters. Event Hubs decouples the production of a stream of events from the consumption of those events, so that event consumers can access the events on their own schedule.

Event Hubs is an processing service that provides event and telemetry ingress to the cloud at massive scale, with low latency and high reliability. This service is especially useful for:

- Application instrumentation
- User experience or workflow processing
- Internet of Things (IoT) scenarios

An Event Hub is created at the namespace level in Service Bus, similar to queues and topics. Event Hubs uses AMQP and HTTP as its primary API interfaces.

The Event Hubs security model meets the following requirements:

- Only devices that present valid credentials can send data to an Event hub.
- A device cannot impersonate another device.
- A rogue device can be blocked from sending data to an Event hub.

The Event Hubs security model is based on a combination of Shared Access Signature (SAS) tokens and event publishers. An event publisher defines a virtual endpoint for an Event Hub. The publisher can only be used to send messages to an Event hub. It is not possible to receive messages from a publisher.

Typically, an Event hub employs one publisher per device. All messages that are sent to any of the publishers of an Event hub are queued within that Event hub. Publishers allow fine-grained access control and throttling.

Each device is assigned a unique token, which is uploaded to the device. The tokens are produced such that each unique token grants access to a different unique publisher. A device that possesses a token can only send to one publisher, but no other publisher. If multiple devices share the same token, then each of these devices shares a publisher.

1.1.1.2 Web services

A SCADA web service will be utilized by SESP for sending control signals and by demand from SESP to SCADA for specific meter values from one or more DER's.

A "HTTP Action" within Azure Logic App will be utilized on the SESP side for the communication with the SCADA web services, and an Azure Logic App "HTTP Listener" will be the endpoint for any returned ACK messages from SCADA to SESP.

2 Interfaces for communication integration

This chapter describes the different interfaces that are involved in data acquisition and integration between Field devices and SESP Platform (metering, smart Home, DER, Load Resources devices)

2.1 DER - SCADA interface

This is the communication between DER devices and SCADA System. Examples of DER devices will be charging points for electrical vehicle and generation resources (solar, wind, inverters).

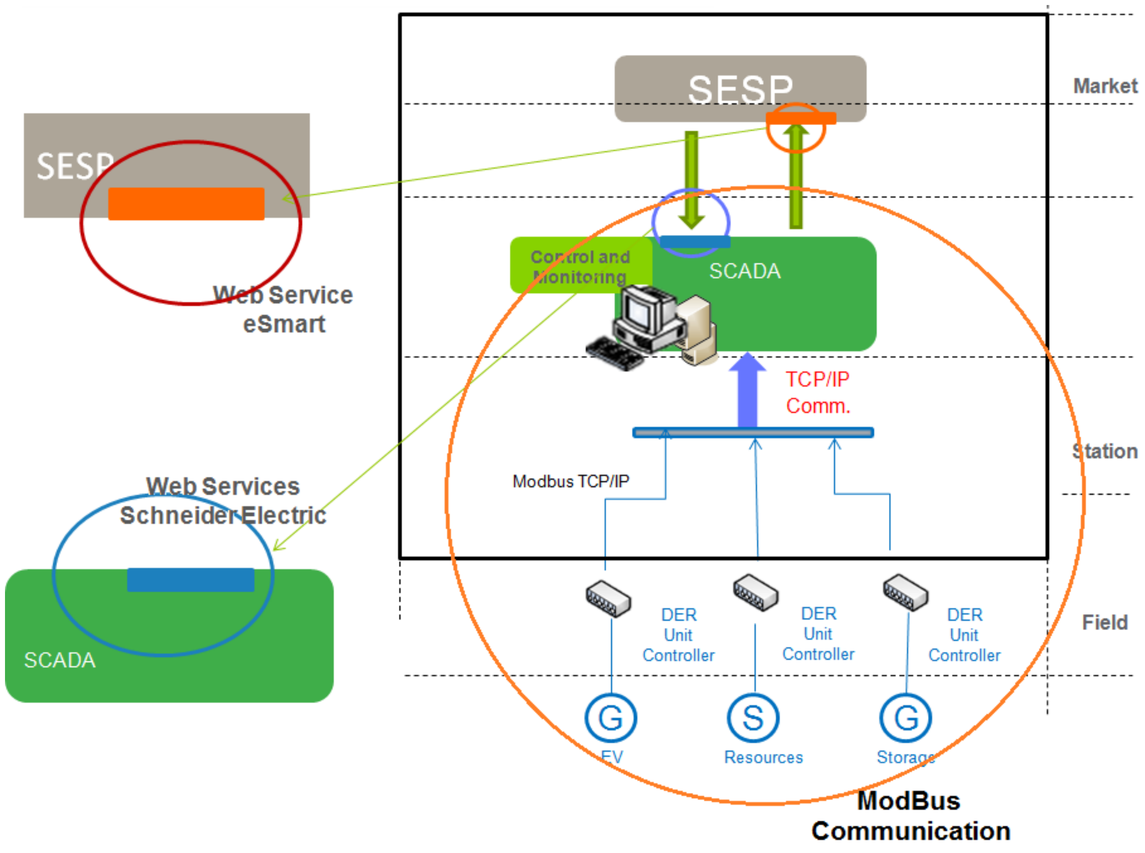


Figure 1: DER – SCADA interface

2.1.1 Architecture Data Management

Communications with the DER devices are very depending on technology. Although these communications will be implemented using Modbus protocol, every manufacture has its own memory maps, addresses, variables, commands, etc.

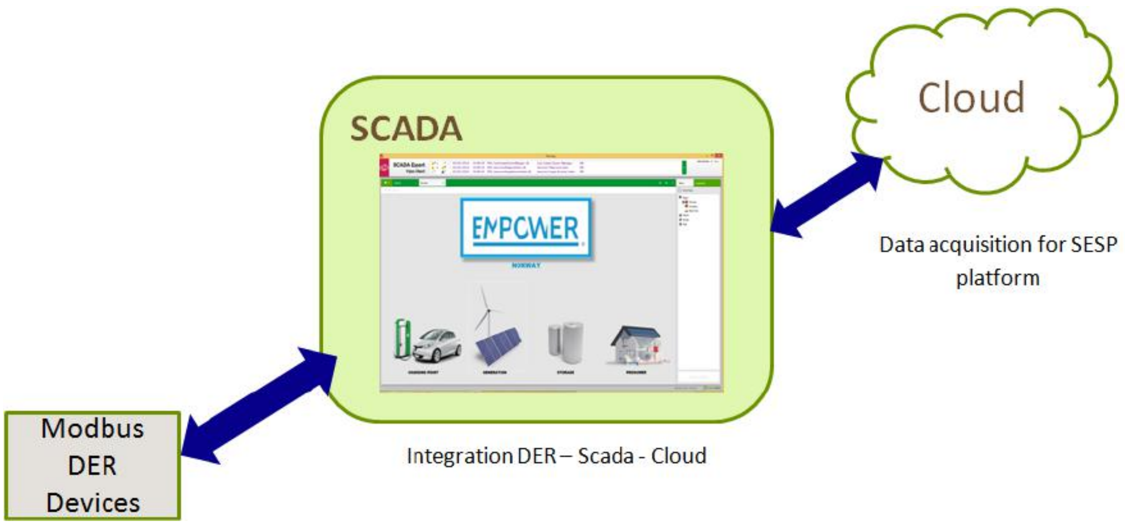


Figure 2: Integration DER – SCADA – SESP platform

For achieving communication between DER devices and the SCADA, it is necessary to know the memory map of these devices, to develop the correct connector using Modbus protocol.

There is no API defined for connecting DER devices and SCADA because this system is adapted to recognize this protocol. However a Modbus mapping is required for every device.

In the following sections, some examples of these Modbus memory mappings are described (Ev and Generation). These register mappings may change depending on the technology and features of the equipment used. Mappings from storage and other devices will be developed during the project execution, and this will be dependent on the WP7 and Pilots deployment.

2.1.2 EV charging station Modbus mapping

The charging station and the external system of remote authentication will communicate with Modbus TCP through a shared Modbus table interface describing the authentication information.

Addr	Register	Type	Size	Access
1520	Remote Authent UID	word	8	RW
1528	Remote user type	word	1	RW
934	Remote authent Manager lifebit	word	1	RW
935	Remote authent Manager status	Word	1	R
150	Remote command	word	1	Read/Write
20	Remote Command Status	word	1	Read
23	Event Status MSB	word	1	Read
24	Event Status LSB	word	1	Read

Table 1: Modbus interface for EV charging stations

The protocol of communication between the external system and the charging station is as following:

The external system pushes to the master board an authentication composed of an UID and an optional user type (user, VIP, admin) and then notifies the master board of this new sending of authentication information by sending a Remote Authentication command.

On reception of this command, the master board processes the authentication information and authenticates the user. The external system checks that the authentication command have been successfully executed by reading the Remote

Command status. After reading, it acknowledges the remote command status to the master board.

On reception of the acknowledgement, EVSE will reset the authentication information and is ready to receive new remote authentication

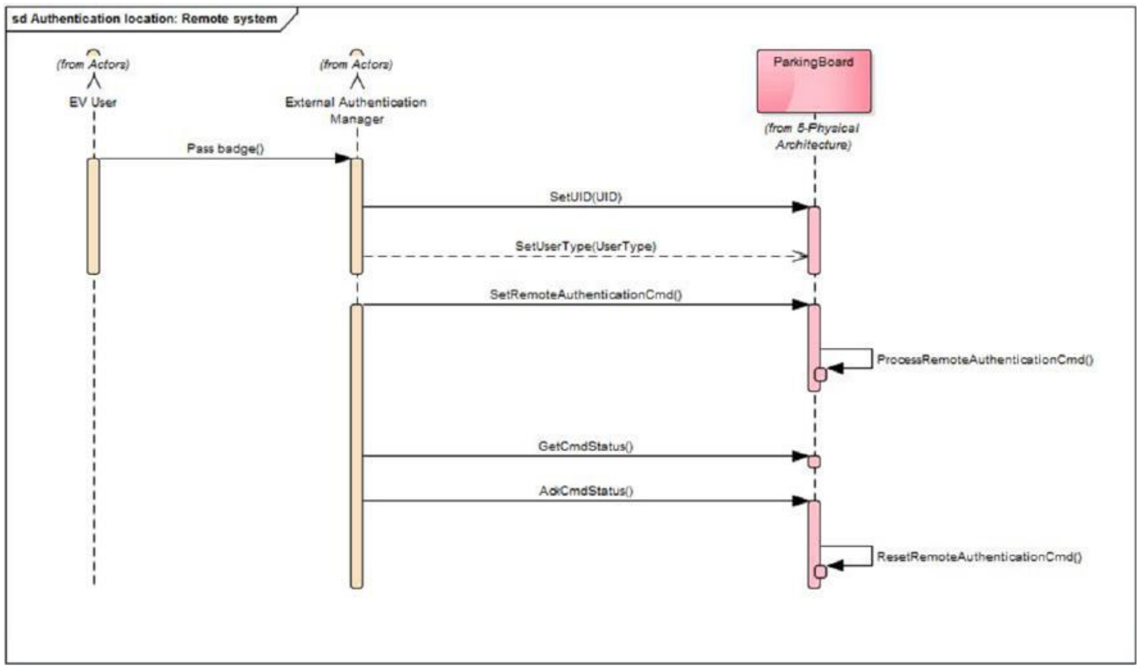

sd Authentication location: Remote system
(from Actors)
EV User
(from Actors)
External Authentication Manager
ParkingBoard
(from 5-Physical Architecture)
Pass badge()
SetUID(UID)
SetUserType(UserType)
SetRemoteAuthenticationCmd()
ProcessRemoteAuthenticationCmd()
GetCmdStatus()
AckCmdStatus()
ResetRemoteAuthenticationCmd()

Figure 3: Protocol Diagram

2.1.2.1 Historic Event Management for EV Charging Stations

The last four events status words are saved in 4 registers. The register contains Start date, End date, Plug and event code.

Addr	Register	Type	Size	Access
1600	PreviousEVent0 start time (s)	word	1	Read
1602	PreviousEvent0 end time (s)	word	1	Read
1604	PreviousEvent0 code	word	1	Read
1605	Reserved	word	1	Read
1615	PreviousEvent1 start time (s)	word	1	Read
1617	PreviousEvent1 end time (s)	word	1	Read
1619	PreviousEvent1 code	word	1	Read
1620	Reserved	word	1	Read
1630	PreviousEvent2 start time (s)	word	1	Read
1632	PreviousEvent2 end time (s)	word	1	Read
1634	PreviousEvent2 code	word	1	Read
1635	Reserved	word	1	Read
1645	PreviousEvent3 start time (s)	word	1	Read
1647	PreviousEvent3 end time (s)	word	1	Read
1649	PreviousEvent3 code	word	1	Read
1650	Reserved	word	1	Read
1660	PreviousEvent4 start time (s)	word	1	Read
1662	PreviousEvent4 end time (s)	word	1	Read
1664	PreviousEvent4 code	word	1	Read

Table 2: Modbus interface for EV charging stations

2.1.3 Inverter for generation Modbus mapping

This section describes the registers mapping for the inverter monitoring data (read-only MODBUS protocol data).

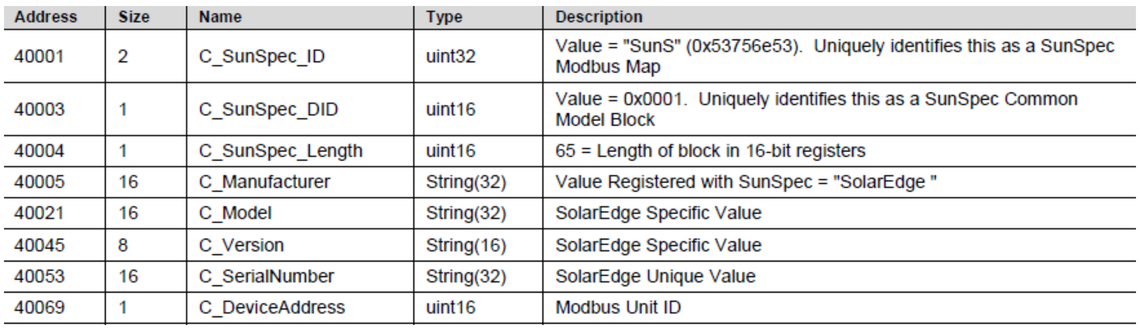
The inverter mapping for monitoring data is based on the open protocol managed by SunSpec: SunSpec Alliance Interoperability Specification – Inverter Models v1.0 (Refer to the *SunSpec Alliance Interoperability Specification – Common Models (Elements)* document for a detailed description of the protocol). The register mapping can be downloaded from the SunSpec Alliance web page: http://www.sunspec.org/.

2.1.3.1 Common Model MODBUS Register Mappings

The base Register Common Block is set to 40001 (Modbus PLC address [base 1]), or 40000 (Modbus Protocol Address [base 0]).

- **C_Manufacturer is set to SolarEdge.**

- **C_Model** is set to the appropriate inverter model, e.g. SE5000.
- **C_Version** contains the CPU software version with leading zeroes, e.g. 0002.0611.
- **C_SerialNumber** contains the inverter serial number.
- **C_DeviceAddress** is the device Modbus ID (default: 1).

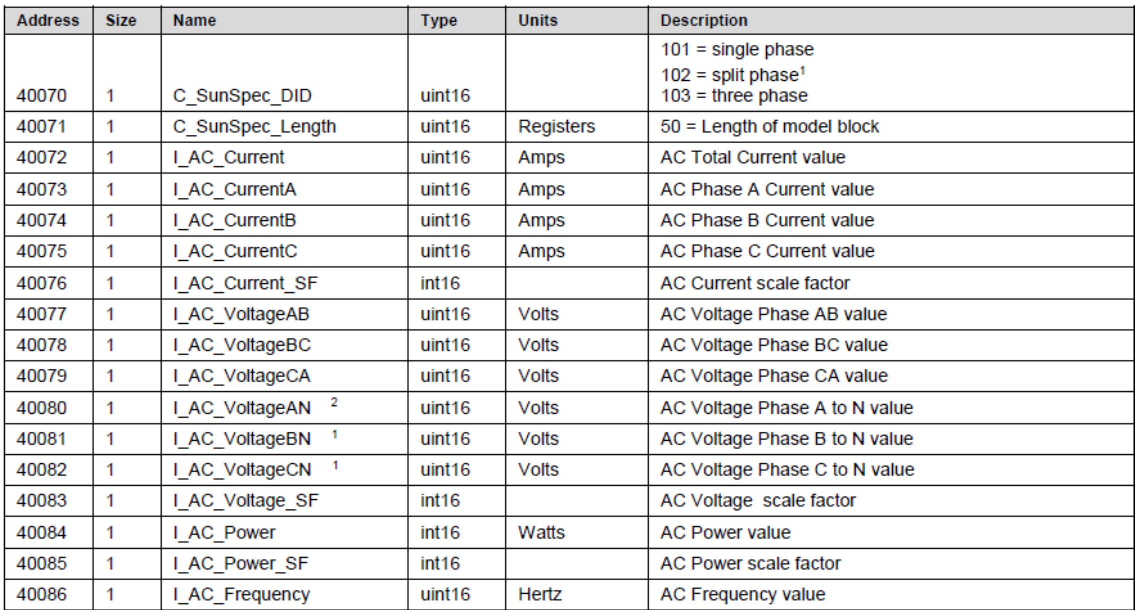


Address	Size	Name	Type	Description
40001	2	C_SunSpec_ID	uint32	Value = "SunS" (0x53756e53). Uniquely identifies this as a SunSpec Modbus Map
40003	1	C_SunSpec_DID	uint16	Value = 0x0001. Uniquely identifies this as a SunSpec Common Model Block
40004	1	C_SunSpec_Length	uint16	65 = Length of block in 16-bit registers
40005	16	C_Manufacturer	String(32)	Value Registered with SunSpec = "SolarEdge "
40021	16	C_Model	String(32)	SolarEdge Specific Value
40045	8	C_Version	String(16)	SolarEdge Specific Value
40053	16	C_SerialNumber	String(32)	SolarEdge Unique Value
40069	1	C_DeviceAddress	uint16	Modbus Unit ID

Table 3: Common Modbus Registers

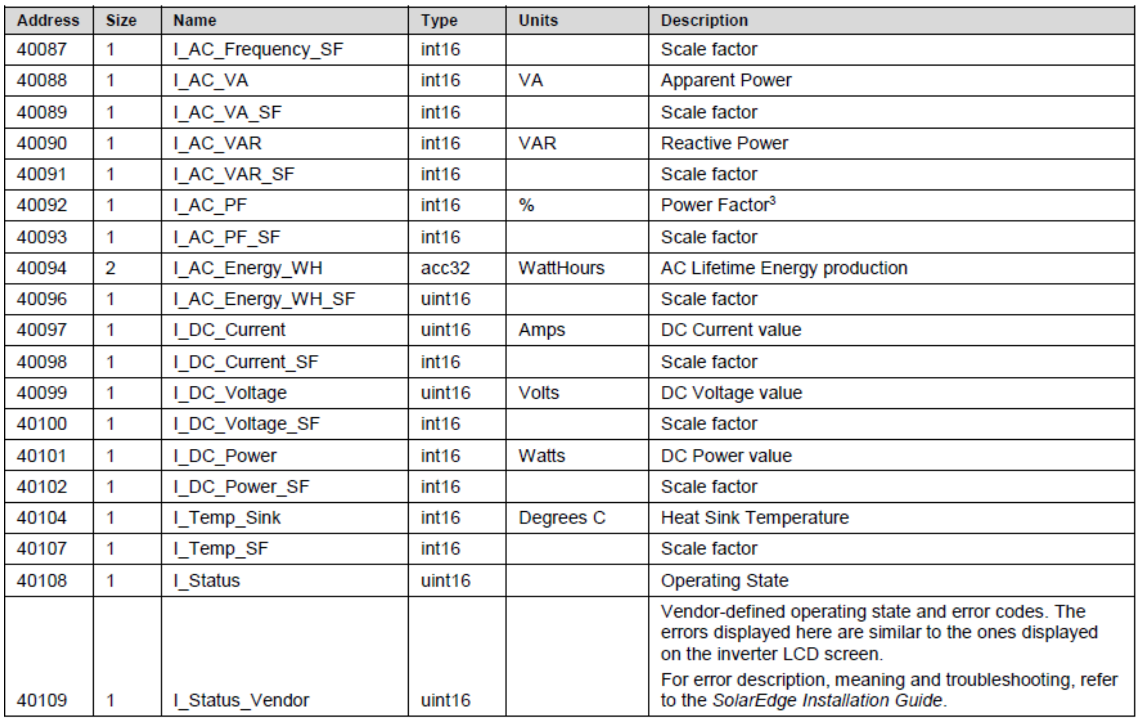
2.1.3.2 Inverter Model MODBUS Register Mappings

The following tables lists the supported MODBUS register values. Unsupported values are indicated by the NOT_IMPLEMENTED value. The base register of the Device Specific block is set to 40070 (Modbus PLC address [base 1]), or 40069 (Modbus Protocol Address [base 0]).



Address	Size	Name	Type	Units	Description
40070	1	C_SunSpec_DID	uint16		101 = single phase 102 = split phase[1] 103 = three phase
40071	1	C_SunSpec_Length	uint16	Registers	50 = Length of model block
40072	1	I_AC_Current	uint16	Amps	AC Total Current value
40073	1	I_AC_CurrentA	uint16	Amps	AC Phase A Current value
40074	1	I_AC_CurrentB	uint16	Amps	AC Phase B Current value
40075	1	I_AC_CurrentC	uint16	Amps	AC Phase C Current value
40076	1	I_AC_Current_SF	int16		AC Current scale factor
40077	1	I_AC_VoltageAB	uint16	Volts	AC Voltage Phase AB value
40078	1	I_AC_VoltageBC	uint16	Volts	AC Voltage Phase BC value
40079	1	I_AC_VoltageCA	uint16	Volts	AC Voltage Phase CA value
40080	1	I_AC_VoltageAN [2]	uint16	Volts	AC Voltage Phase A to N value
40081	1	I_AC_VoltageBN [1]	uint16	Volts	AC Voltage Phase B to N value
40082	1	I_AC_VoltageCN [1]	uint16	Volts	AC Voltage Phase C to N value
40083	1	I_AC_Voltage_SF	int16		AC Voltage scale factor
40084	1	I_AC_Power	int16	Watts	AC Power value
40085	1	I_AC_Power_SF	int16		AC Power scale factor
40086	1	I_AC_Frequency	uint16	Hertz	AC Frequency value

Table 4: Solar inverter register Modbus



Address	Size	Name	Type	Units	Description
40087	1	I_AC_Frequency_SF	int16		Scale factor
40088	1	I_AC_VA	int16	VA	Apparent Power
40089	1	I_AC_VA_SF	int16		Scale factor
40090	1	I_AC_VAR	int16	VAR	Reactive Power
40091	1	I_AC_VAR_SF	int16		Scale factor
40092	1	I_AC_PF	int16	%	Power Factor[3]
40093	1	I_AC_PF_SF	int16		Scale factor
40094	2	I_AC_Energy_WH	acc32	WattHours	AC Lifetime Energy production
40096	1	I_AC_Energy_WH_SF	uint16		Scale factor
40097	1	I_DC_Current	uint16	Amps	DC Current value
40098	1	I_DC_Current_SF	int16		Scale factor
40099	1	I_DC_Voltage	uint16	Volts	DC Voltage value
40100	1	I_DC_Voltage_SF	int16		Scale factor
40101	1	I_DC_Power	int16	Watts	DC Power value
40102	1	I_DC_Power_SF	int16		Scale factor
40104	1	I_Temp_Sink	int16	Degrees C	Heat Sink Temperature
40107	1	I_Temp_SF	int16		Scale factor
40108	1	I_Status	uint16		Operating State
40109	1	I_Status_Vendor	uint16		Vendor-defined operating state and error codes. The errors displayed here are similar to the ones displayed on the inverter LCD screen. For error description, meaning and troubleshooting, refer to the *SolarEdge Installation Guide*.

Table 5: Solar Inverter Register Modbus

2.2 SCADA – SESP interface (Azure Event Hub Send Event)

As mentioned above, a dedicated Azure Event Hub will be used for all communications from SCADA to the SESP system.

2.2.1 The API description of the Azure event hub REST API is as follows:Request

Method	**Request URI**	**HTTP version**
Post	https://{serviceNamespace}.servicebus.windows.net/{eventHubPath}/publishers/{gatewayid}/messages	HTTP/1.1

The *serviceNamespace, eventHubPath* are parameters that are unique for each SESP installation. The SESP platform environment created for the EMPOWER project is configured with the following parameters:

- {serviceNamespace} = “empower-cg-prod”
- {eventHubPath} = "iotexternaleh"

The parameter {gatewayId} is a (globally) unique identifier of the sending device that is sending the data. A dedicated *gatewayId* will be created in SESP for each sending device. If there exist more than one sending device there will be one gatewayId for each device. For each gateway there is a corresponding SAS token, which is used for authentication.

2.2.1.1 Request Headers

Request Header	**Description**
Authorization	A SAS token.
Content-Type	Set to **application/atom+xml;type=entry;charset=utf-8.**
BrokerProperties	JSON-encoded list of BrokeredProperties

2.2.1.2 Request body

The body of the event message or JSON-encoded string that contains multiple messages.

2.2.2 Response

The response includes an HTTP status code and a set of response headers.

2.2.2.1 Response Codes

Code	**Description**
201	Success
401	Authorization failure.
500	Internal error.

2.2.2.2 Response headers

None

2.2.2.3 Response Body

If the request is successful, the body is empty. If the request is not successful, the body contains an error code and error message.

2.3 Prosumer – SESP interface (UPC & eSmart)

A local controller (gateway) will be installed at all prosumers, and all communication between the SESP and the prosumer will be handled by the local controller. Azure

Event Hub will be used for all telemetry messages (data collection) from the local controller to the SESP system (lease see paragraph 2.2 for more details regarding the Azure Event Hub Send Event), and Azure IoT Hub will be used for all communication (request for data, control signals, etc.) from the SESP to the various devices connected to the local controller.

Azure IoT Hub is a fully managed service that enables reliable and secure bidirectional communications between millions of IoT devices and a solution back end. Azure IoT Hub:

- Provides reliable device-to-cloud and cloud-to-device messaging at scale.
- Enables secure communications using per-device security credentials and access control.
- Provides extensive monitoring for device connectivity and device identity management events.
- Includes device libraries for the most popular languages and platforms.

Azure IoT Hub offer reliable cloud-to-device messaging (or commands). The solution back end can use IoT Hub to send messages with an at-least-once delivery guarantee to individual devices. Each message has an individual time-to-live setting, and the back end can request both delivery and expiration receipts. This ensures full visibility into the life cycle of a cloud-to-device message.

3 Use Cases API´s

3.1 Electric Vehicle Charging Station (EV_CS)

3.1.1 Messages from EV_CS to SESP

All technical information from the charging stations will be sent to the SESP in a periodically manner. In addition, the SESP can request for such information at any time if needed.

SCADA will be the gateway between all charging stations and SESP, and all communication to and from SESP will take place via SCADA.

An asset register in SESP is populated with detailed information about each charging station. The SESP asset register will be initialized and maintained in a manual manner.

The technical information will be sent to the SESP by posting messages to the Azure Event Hub (described in 1.1.1.1). All technical information from the charging station will be stored as time series in SESP.

The table below describes all types of technical information that can be sent from the charging station to SESP.

Table 3-1 Electrical vehicle variables

Variable	**Unit**	**Resolution**	**Type**
Active power (charging power)	Watt	1 minute	Time serie
Reactive power	VA	1 minute	Time serie
Voltage	V	1 minute	Time serie
Current	A	1 minute	Time serie
Charging point state*	0, 1, 2	1 minute	Time serie
Other (other properties may be added if needed)			

*Charging point state – possible values: 0-suspended, 1-available, 2-charging

The Azure event hub is designed so that it works better when receiving smaller data packages frequently rather than receiving large data packets less frequently. Consequently there will be created one message for each variable when sending values from the charging station to SESP. The message format will be JSON, and the message is described in a more detailed manner below.

3.1.1.1 Electrical vehicle charging station message format

The following describes a JSON message format for sending messages to the SESP event hub, which is the ingestion point of the SESP IoT service.

Property name	**Type**	**Optional**	**Description**
deviceid	string	N	This id is a unique identifier for the device.
property	string	N	This uniquely identify the property of the value sent. Valid properties for EV_CS to SESP messages are: - “Active Power”

Property name	Type	Optional	Description
			- "Reactive Power" - "Voltage" - "Current" - "Charging Point State"
format	string	Y	
timezone	string	Y	Time zone of the start and end times in the Value object (see below). This is optional, and preferred practice is to include time zone in the time fields.
unit	string	Y	Unit of value. This is optional. E.g. "Watt", "MW", "C"
values	array	N	This is an array of value objects. The value object is described below. The values array is required and must have at least one value object.

Table 3-2 Electrical vehicle charging station message format

3.1.1.2 Value object format for electrical vehicle charging station message format

Property name	Type	Description
starttime	string datetime	Start time of the value. This is normally required, but may be omitted for a device continuously sending instantaneous values. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. "2015-04-30T14:32:52Z" If no time zone is given UTC is assumed.
endtime	string datetime	End time of a value that applies for a duration of time (e.g. metered consumption for an hour, maximum temperature for a day). This is optional and would normally not be included for instantaneous values. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. 2015-04-30T14:32:52Z If no time zone is given UTC is assumed.

Property name	Type	Description
value	number	The value. Note that "." is always used as decimal separator for JSON messages. value is an optional field, but either the value or the message field must be present in a Value object.
message	string	This may either be a status modifier for the value or a notification message from the device. message is an optional field, but either the value or the message field must be present in a Value object.

Table 3-3 Value object format for electrical vehicle charging station message format

3.1.1.3 Message schema

The schema for the message is presented below:

```
{

    "$schema": "http://json-schema.org/draft-04/schema#",
    "name": "Sample",
    "type": "object",
    "properties": {
        "deviceid": {
            "type": "string",
            "description": "Device identifier"
        },
        "property": {
            "type": "string",
            "description": "Property identifier"
        },
        "format": {
            "type": "string",
            "description": "Message format including version number"
        },
        "timezone": {
            "type": "string",
            "description": "Time zone for starttime and endttime in values
array"
        },
        "unit": {
            "type": "string",
            "description": "values unit"
        },
        "values": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "starttime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "start time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "endtime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "end time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "value": {
                        "type": "number",
                        "description": "value"
                    },
                    "message": {
                        "type": "string",
                        "description": "status modifier or notification
message"
                    }
                }
            }
        }
    },
    "required": ["deviceid",
    "values"]
}
```

3.1.2 Messages from SESP to EV_CS

The SESP shall have the possibility to send messages to an electric vehicle charging station. Types of messages are either a request for variables (time series for a given property at a given DER), or control signals for demand response.

All SESP messages will be sent to SCADA who will be responsible for re-routing the message in an appropriate format according to what supported by the EV_CS.

Any control signals from SESP to EV_CS can contain both *starttime* and *endtime* for the control signal, and in that case SCADA will be responsible for the execution of both start and stop of the demand response plan. If the control signal only includes a *starttime*, SESP will be responsible for sending a new message that will reverse the control command when the demand response plan is finished.

3.1.2.1 Message format

The following describes a JSON message format for sending messages from SESP to EV_CS (SCADA). The SESP will use SCADA as the endpoint for all messages initiated by the SESP, and it will be SCADAs responsibility to dispatch the message to the EV_CS.

Property name	**Type**	**Optional**	**Description**
messageid	string	N	This id is a unique message identifier, and this message identifier must be included in an ACKnowledge message from SCADA to SESP.
deviceid	string	N	This id is a unique identifier for the device.
action	String	N	Define the action to execute – valid values are: - “get”

Property name	Type	Optional	Description
			- “set”
property	string	N	This uniquely identify the property of the action sent. Valid properties for SESP to EV_CS messages are: - “Charge” (“set”) - “Active Power” (“get”) - “Reactive Power” (“get”) - “Voltage” (“get”) - “Current” (“get”) - “Charging Point State” (“get”)
format	string	Y	
timezone	string	Y	Time zone of the start and end times in the Value object (see below). This is optional, and preferred practice is to include time zone in the time fields.
unit	string	Y	Unit of value. This is optional. E.g. “Watt”, “MW”, “C”
values	array	N	This is an array of value objects. The value object is described below. The values array is required and must have at least one value object.

Table 3-4 Message format

3.1.2.2 Value object format

Property name	Type	Description
starttime	string datetime	Start time for when to perform the action. If omitted, the action will be executed immediately. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. “2015-04-30T14:32:52Z” If no time zone is given UTC is assumed.
endtime	string datetime	The end time for when to end the interval for a control signal. The time is given as an ISO 8601 date time formatted string,

Property name	Type	Description
		including time zone: E.g. 2015-04-30T14:32:52Z If no time zone is given UTC is assumed.
value	number	The value. Note that "." is always used as decimal separator for JSON messages. value is an optional field, but either the value or the message field must be present in a Value object. Valid values for action "set" together with property "Charge" are: - 1 (Charge on) - 0 (Charge off) No values are necessary for action "get", and in this case the message element will be populated with "N/A".
message	string	This may either be a status modifier for the value or a notification message from the device. message is an optional field, but either the value or the message field must be present in a Value object.

Table 3-5 Value object format

3.1.2.3 Message schema

The schema for the message is presented below:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "name": "Sample",
    "type": "object",
    "properties": {
        "messageid": {
            "type": "string",
            "description": "Message identifier"
        },
        "deviceid": {
            "type": "string",
            "description": "Device identifier"
        },
        "action": {
            "type": "string",
            "description": "Action to execute"
        },
        "property": {
            "type": "string",
            "description": "Property identifier"
        },
        "format": {
            "type": "string",
            "description": "Message format including version number"
        },
        "timezone": {
            "type": "string",
            "description": "Time zone for starttime and endttime in values
array"
```

```
        },
        "unit": {
            "type": "string",
            "description": "values unit"
        },
        "values": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "starttime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "start time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "endtime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "end time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "value": {
                        "type": "number",
                        "description": "value"
                    },
                    "message": {
                        "type": "string",
                        "description": "status modifier or notification
message"
                    }
                }
            }
        }
    },
    "required": ["deviceid",
    "action",
    "property",
    "values"]
}
```

3.2 Generation Resources

3.2.1 Messages from generation to SESP

All technical information from generation resources like solar panels and wind turbines will be sent to the SESP in a periodically manner. In addition, the SESP can request for such information at any time if needed.

SCADA will be the gateway between all generation resources and SESP, and all communication to and from SESP will thus take place via SCADA.

An asset register in SESP is populated with detailed information about each generation resource. The SESP asset register will be initialized and maintained in a manual manner.

The technical information will be sent to the SESP by posting messages to the Azure Event Hub (described in 1.1.1.1). All technical information from the generation resources will be stored as time series in SESP.

The table below describes all types of technical information that can be sent from the generation resources to SESP.

Variable	**Unit**	**Resolution**	**Type**
Active power	Watt	1 minute	Time serie
Reactive power	VA	1 minute	Time serie
Other (other properties may be added if needed)			

Table 3-6 Generation variables

3.2.1.1 Generation message format

The following describes a JSON message format for sending messages the generators to the SESP event hub, which is the ingestion point of the SESP IoT service.

Property name	**Type**	**Optional**	**Description**
deviceid	string	N	This id is a unique identifier for the device.
property	string	N	This uniquely identify the property of the value sent. Valid properties for generators to SESP messages are: - “Active Power” - “Reactive Power”
format	string	Y	
timezone	string	Y	Time zone of the start and end times in the Value object (see below). This is optional, and preferred practice is to include time zone in the time fields.
unit	string	Y	Unit of value. This is optional. E.g. “KW”, “MW”, “C”
values	array	N	This is an array of value objects. The value object is described below. The values array is required and

Property name	Type	Optional	Description
			must have at least one value object.

Table 3-7 Generation message format

3.2.1.2 Value object format for generation message format

Property name	Type	Description
starttime	string datetime	Start time of the value. This is normally required, but may be omitted for a device continuously sending instantaneous values. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. “2015-04-30T14:32:52Z” If no time zone is given UTC is assumed.
endtime	string datetime	End time of a value that applies for a duration of time (e.g. metered consumption for an hour, maximum temperature for a day). This is optional and would normally not be included for instantaneous values. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. 2015-04-30T14:32:52Z If no time zone is given UTC is assumed.
value	number	The value. Note that “.” is always used as decimal separator for JSON messages. value is an optional field, but either the value or the message field must be present in a Value object.
message	string	This may either be a status modifier for the value or a notification message from the device. message is an optional field, but either the value or the message field must be present in a Value object.

Table 3-8 Value object format for generation message format

3.2.1.3 Generation to SESP message schema

The schema for the message is presented below:

```
{

    "$schema": "http://json-schema.org/draft-04/schema#",
    "name": "Sample",
    "type": "object",
    "properties": {
        "deviceid": {
            "type": "string",
            "description": "Device identifier"
        },
        "property": {
            "type": "string",
            "description": "Property identifier"
        },
        "format": {
            "type": "string",
            "description": "Message format including version number"
        },
        "timezone": {
            "type": "string",
            "description": "Time zone for starttime and endttime in values
array"
        },
        "unit": {
            "type": "string",
            "description": "values unit"
        },
        "values": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "starttime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "start time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "endtime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "end time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "value": {
                        "type": "number",
                        "description": "value"
                    },
                    "message": {
                        "type": "string",
                        "description": "status modifier or notification
message"
                    }
                }
            }
        }
    },
    "required": ["deviceid",
    "values"]
}
```

3.2.2 Messages from SESP to Generation

The SESP shall have the possibility to send messages to a PV or wind turbine generator. Types of messages are either a request for variables (time series for a given property at a given DER), or control signals for demand response (start or stop of production).

All SESP messages will be sent to SCADA who will be responsible for re-routing the message in an appropriate format according to what supported by the generation resource.

Any control signals from SESP to generation can contain both *starttime* and *endtime* for the control signal, and in that case SCADA will be responsible for the execution of both start and stop of the demand response plan. If the control signal only includes a *starttime*, SESP will be responsible for sending a new message that will reverse the control command when the demand response plan is finished.

3.2.2.1 Message format

The following describes a JSON message format for sending messages from SESP to a generator (SCADA). The SESP will use SCADA as the endpoint for all messages initiated by the SESP, and it will be SCADAs responsibility to dispatch the message to the generator.

Property name	**Type**	**Optional**	**Description**
messageid	string	N	This id is a unique message identifier, and this message identifier must be included in an ACKnowledge message from SCADA to SESP.
deviceid	string	N	This id is a unique identifier for the device.
action	String	N	Define the action to execute – valid values are: - “get” - “set”
property	string	N	This uniquely identify the property of the action sent. Valid properties for SESP to generation messages are: - “Generation” (“set”) - “Active Power” (“get”) - “Reactive Power” (“get”)
format	string	Y	
timezone	string	Y	Time zone of the start and end times in the Value object (see below). This is optional, and preferred practice is to include time zone in the time fields.
unit	string	Y	Unit of value. This is optional. E.g. “watt”, “MW”, “C”
values	array	N	This is an array of value objects. The value object is described below. The values array is required and

Property name	Type	Optional	Description
			must have at least one value object.

Table 3-9 Message format from SESP to Generator

3.2.2.2 Value object format

Property name	Type	Description
starttime	string datetime	Start time for when to perform the action. If omitted, the action will be executed immediately. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. “2015-04-30T14:32:52Z” If no time zone is given UTC is assumed.
endtime	string datetime	The end time for when to end the interval for a control signal. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. 2015-04-30T14:32:52Z If no time zone is given UTC is assumed.
value	number	The value. Note that “.” is always used as decimal separator for JSON messages. value is an optional field, but either the value or the message field must be present in a Value object. Valid values for action “set” together with property “Generation” are: - 1 (Generation on) - 0 (Generation off) No values are necessary for action “get”, and in this case the message element will be populated with “N/A”.
message	string	This may either be a status modifier for the value or a notification message from the device. message is an optional field, but either the value or the message field must be present in a Value object.

Table 3-10 Value object format from SESP to Generator

3.2.2.3 Message schema

The schema for the message is presented below:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "name": "Sample",
    "type": "object",
    "properties": {
        "messageid": {
            "type": "string",
            "description": "Message identifier"
        },
        "deviceid": {
            "type": "string",
            "description": "Device identifier"
        },
        "action": {
            "type": "string",
            "description": "Action to execute"
        },
        "property": {
            "type": "string",
            "description": "Property identifier"
        },
        "format": {
            "type": "string",
            "description": "Message format including version number"
        },
        "timezone": {
            "type": "string",
            "description": "Time zone for starttime and endttime in values
array"
        },
        "unit": {
            "type": "string",
            "description": "values unit"
        },
        "values": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "starttime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "start time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "endtime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "end time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "value": {
                        "type": "number",
                        "description": "value"
                    },
                    "message": {
                        "type": "string",
                        "description": "status modifier or notification
message"
                    }
                }
            }
        }
    },
    "required": ["deviceid",
    "action",
    "property",
    "values"]
}
```

3.3 Storage

3.3.1 Messages from Storage to SESP

All technical information from the storage will be sent to the SESP in a periodically manner. In addition, the SESP can request for such information at any time if needed.

SCADA will be the gateway between the storage and SESP, and all communication between to and from SESP will thus take place via SCADA.

An asset register in SESP must be populated with detailed information about each storage. The SESP asset register will be initialized and maintained in a manual manner.

The technical information will be sent to the SESP by posting messages to the Azure Event Hub (described in 1.1.1.1). All technical information from the storage will be stored as time series in SESP.

The table below describes all types of technical information that can be sent from the storage to SESP.

Variable	**Unit**	**Resolution**	**Type**
Charging current	A	1 minute	Time serie
Discharging current	A	1 minute	Time serie
State of charge	Ah	1 minute	Time serie
Other (other properties may be added if needed)			

Table 3-11 Storage variables

The Azure event hub is designed so that it works better when receiving smaller data packages frequently rather than receiving large data packets less frequently. As a consequence, there will be created one message for each variable when sending values from the charging station to SESP. The message format shall be JSON, and the message is described in a more detailed manner below.

3.3.1.1 Storage message format

The following describes a JSON message format for sending messages from the storage to the SESP event hub, which is the ingestion point of the SESP IoT service.

Property name	Type	Optional	Description
deviceid	string	N	This id is a unique identifier for the device.
property	string	N	This uniquely identify the property of the value sent. Valid properties for storage to SESP messages are: - “Charging current” - “Discharging current” - “State of charge”
format	string	Y	
timezone	string	Y	Time zone of the start and end times in the Value object (see below). This is optional, and preferred practice is to include time zone in the time fields.
unit	string	Y	Unit of value. This is optional. E.g. “A”, “Ah”
values	array	N	This is an array of value objects. The value object is described below. The values array is required and must have at least one value object.

Table 3-12 Storage message format

3.3.1.2 Value object format for storage message format

Property name	Type	Description
starttime	string datetime	Start time of the value. This is normally required, but may be omitted for a device continuously sending instantaneous values. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. “2015-04-30T14:32:52Z” If no time zone is given UTC is assumed.
endtime	string datetime	End time of a value that applies for a duration of time (e.g. metered consumption for an hour, maximum temperature for a day). This is optional and would normally not be included for instantaneous values. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. 2015-04-30T14:32:52Z If no time zone is given UTC is assumed.

Property name	Type	Description
value	number	The value. Note that "." is always used as decimal separator for JSON messages. value is an optional field, but either the value or the message field must be present in a Value object.
message	string	This may either be a status modifier for the value or a notification message from the device. message is an optional field, but either the value or the message field must be present in a Value object.

Table 3-13 Value object format for storage message format

3.3.1.3 Storage to SESP message schema

The schema for the message is presented below:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "name": "Sample",
    "type": "object",
    "properties": {
        "deviceid": {
            "type": "string",
            "description": "Device identifier"
        },
        "property": {
            "type": "string",
            "description": "Property identifier"
        },
        "format": {
            "type": "string",
            "description": "Message format including version number"
        },
        "timezone": {
            "type": "string",
            "description": "Time zone for starttime and endttime in values
array"
        },
        "unit": {
            "type": "string",
            "description": "values unit"
        },
        "values": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "starttime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "start time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "endtime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "end time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "value": {
                        "type": "number",
                        "description": "value"
                    },
                    "message": {
                        "type": "string",
                        "description": "status modifier or notification
message"
                    }
                }
            }
        }
    },
    "required": ["deviceid",
    "values"]
}
```

3.3.2 Messages from SESP to Storage

The SESP shall have the possibility to send messages to a storage. Types of messages are either a request for variables (time series for a given property at a given DER), or control signals for demand response (start or stop of charging).

All SESP messages will be sent to SCADA who will be responsible for re-routing the message in an appropriate format according to what supported by the storage.

Any control signals from SESP to storage can contain both *starttime* and *endtime* for the control signal, and in that case SCADA will be responsible for the execution of both start and stop of the demand response plan. If the control signal only includes a *starttime*, SESP will be responsible for sending a new message that will reverse the control command when the demand response plan is finished.

3.3.2.1 Message format

The following describes a JSON message format for sending messages from SESP to storage(SCADA). The SESP will use SCADA as the endpoint for all messages initiated by the SESP, and it will be SCADAs responsibility to dispatch the message to the storage.

Property name	**Type**	**Optional**	**Description**
messageid	string	N	This id is a unique message identifier, and this message identifier must be included in an ACKnowledge message from SCADA to SESP.
deviceid	string	N	This id is a unique identifier for the device.
action	String	N	Define the action to execute – valid values are:

Property name	Type	Optional	Description
			- "get" - "set"
property	string	N	This uniquely identify the property of the action sent. Valid properties for SESP to Storage messages are: - "Charge" ("set") - "Charging current" (get) - "Discharging current" (get) - "State of charge" (get)
format	string	Y	
timezone	string	Y	Time zone of the start and end times in the Value object (see below). This is optional, and preferred practice is to include time zone in the time fields.
unit	string	Y	Unit of value. This is optional. E.g. "Watt", "MW", "C"
values	array	N	This is an array of value objects. The value object is described below. The values array is required and must have at least one value object.

Table 3-14 Message format from SESP to Storage

3.3.2.2 Value object format

Property name	Type	Description
starttime	string datetime	Start time for when to perform the action. If omitted, the action will be executed immediately. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. "2015-04-30T14:32:52Z" If no time zone is given UTC is assumed.
endtime	string datetime	The end time for when to end the interval for a control signal. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. 2015-04-30T14:32:52Z

Property name	Type	Description
		If no time zone is given UTC is assumed.
value	number	The value. Note that "." is always used as decimal separator for JSON messages. value is an optional field, but either the value or the message field must be present in a Value object. Valid values for action "set" together with property "Charge" are: - 1 (Charge on) - 0 (Charge off) No values are necessary for action "get", and in this case the message element will be populated with "N/A".
message	string	This may either be a status modifier for the value or a notification message from the device. message is an optional field, but either the value or the message field must be present in a Value object.

Table 3-15 Value object format from SESP to Storage

3.3.2.3 Message schema

The schema for the message is presented below:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "name": "Sample",
    "type": "object",
    "properties": {
        "messageid": {
            "type": "string",
            "description": "Message identifier"
        },
        "deviceid": {
            "type": "string",
            "description": "Device identifier"
        },
        "action": {
            "type": "string",
            "description": "Action to execute"
        },
        "property": {
            "type": "string",
            "description": "Property identifier"
        },
        "format": {
            "type": "string",
            "description": "Message format including version number"
        },
        "timezone": {
            "type": "string",
            "description": "Time zone for starttime and endttime in values
array"
        },
        "unit": {
            "type": "string",
            "description": "values unit"
        },
        "values": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "starttime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "start time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "endtime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "end time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "value": {
                        "type": "number",
                        "description": "value"
                    },
                    "message": {
                        "type": "string",
                        "description": "status modifier or notification
message"
                    }
                }
            }
        }
    },
    "required": ["deviceid",
    "action",
    "property",
    "values"]
}
```

3.4 Load Resources

3.4.1 Messages from load resources to SESP

All technical information from the controllable load resources will be sent to the SESP in a periodically manner. In addition, the SESP can request for such information at any time if needed.

SCADA will be the gateway between the load resources and SESP, and all communication between to and from SESP will thus take place via SCADA.

An asset register in SESP is populated with detailed information about each load resource. The SESP asset register will be initialized and maintained in a manual manner.

The technical information will be sent to the SESP by posting messages to the Azure Event Hub (described in 1.1.1.1). All technical information from the load resources will be stored as time series in SESP.

The table below describes all types of technical information that can be sent from the load resources to SESP.

Variable	**Unit**	**Resolution**	**Type**
Active power	Watt	1 minute	Time serie
Reactive power	VA	1 minute	Time serie
Other (other properties may be added if needed)			

Table 3-16 Controllable load variables

The Azure event hub is designed so that it works better when receiving smaller data packages frequently rather than receiving large data packets less frequently. As a consequence, there will be created one message for each variable when sending values from the load resources to SESP. The message format shall be JSON, and the message is described in a more detailed manner below.

3.4.1.1 Load resources message format

The following describes a JSON message format for sending messages from the load resources to the SESP event hub, which is the ingestion point of the SESP IoT service.

Property name	**Type**	**Optional**	**Description**
deviceid	string	N	This id is a unique identifier for the device.
property	string	N	This uniquely identify the property of the value sent. Valid properties for load resources to SESP messages are: - “Active Power” - “Reactive Power”
format	string	Y	
timezone	string	Y	Time zone of the start and end times in the Value object (see below). This is optional, and preferred practice is to include time zone in the time fields.
unit	string	Y	Unit of value. This is optional. E.g. “KW”, “MW”, “C”
values	array	N	This is an array of value objects. The value object is

Property name	Type	Optional	Description
			described below. The values array is required and must have at least one value object.

Table 3-17 Load resources message format

3.4.1.2 Value object format for load resources message format

Property name	Type	Description
starttime	string datetime	Start time of the value. This is normally required, but may be omitted for a device continuously sending instantaneous values. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. "2015-04-30T14:32:52Z" If no time zone is given UTC is assumed.
endtime	string datetime	End time of a value that applies for a duration of time (e.g. metered consumption for an hour, maximum temperature for a day). This is optional and would normally not be included for instantaneous values. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. 2015-04-30T14:32:52Z If no time zone is given UTC is assumed.
value	number	The value. Note that "." is always used as decimal separator for JSON messages. value is an optional field, but either the value or the message field must be present in a Value object.
message	string	This may either be a status modifier for the value or a notification message from the device. message is an optional field, but either the value or the message field must be present in a Value object.

Table 3-18 Value object format for load resources

3.4.1.3 Load resources to SESP message schema

The schema for the message is presented below:

```
{

    "$schema": "http://json-schema.org/draft-04/schema#",
    "name": "Sample",
    "type": "object",
    "properties": {
        "deviceid": {
            "type": "string",
            "description": "Device identifier"
        },
        "property": {
            "type": "string",
            "description": "Property identifier"
        },
        "format": {
            "type": "string",
            "description": "Message format including version number"
        },
        "timezone": {
            "type": "string",
            "description": "Time zone for starttime and endttime in values
array"
        },
        "unit": {
            "type": "string",
            "description": "values unit"
        },
        "values": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "starttime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "start time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "endtime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "end time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "value": {
                        "type": "number",
                        "description": "value"
                    },
                    "message": {
                        "type": "string",
                        "description": "status modifier or notification
message"
                    }
                }
            }
        }
    },
    "required": ["deviceid",
    "values"]
}
```

3.4.2 Messages from SESP to Load resources

The SESP shall have the possibility to send messages to a load resource. Types of messages are either a request for variables, or control signals for demand response (power off / power on).

All SESP messages will be sent to SCADA who will be responsible for re-routing the message in an appropriate format according to what supported by the load resource.

Any control signals from SESP to a load resource can contain both *starttime* and *endtime* for the control signal, and in that case SCADA will be responsible for the execution of both start and stop of the demand response plan. If the control signal only includes a *starttime*, SESP will be responsible for sending a new message that will reverse the control command when the demand response plan is finished.

3.4.2.1 Message format

The following describes a JSON message format for sending messages from SESP to a load resource (SCADA). The SESP will use SCADA as the endpoint for all messages initiated by the SESP, and it will be SCADAs responsibility to dispatch the message to the load resource.

Property name	**Type**	**Optional**	**Description**
messageid	string	N	This id is a unique message identifier, and this message identifier must be included in an ACKnowledge message from SCADA to SESP.
deviceid	string	N	This id is a unique identifier for the device.
action	String	N	Define the action to execute – valid values are: - “get” - “set”
property	string	N	This uniquely identify the property of the action sent. Valid properties for SESP to load resource messages are: - “Power” (“set”) - “Active Power” (“get”) - “Reactive Power” (“get”)
format	string	Y	
timezone	string	Y	Time zone of the start and end times in the Value object (see below). This is optional, and preferred practice is to include time zone in the time fields.
unit	string	Y	Unit of value. This is optional. E.g. “Watt”, “MW”, “C”
values	array	N	This is an array of value objects. The value object is described below. The values array is required and

Property name	Type	Optional	Description
			must have at least one value object.

Table 3-19 Message format from SESP to load resources

3.4.2.2 Value object format

Property name	Type	Description
starttime	string datetime	Start time for when to perform the action. If omitted, the action will be executed immediately. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. “2015-04-30T14:32:52Z” If no time zone is given UTC is assumed.
endtime	string datetime	The end time for when to end the interval for a control signal. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. 2015-04-30T14:32:52Z If no time zone is given UTC is assumed.
value	number	The value. Note that “.” is always used as decimal separator for JSON messages. value is an optional field, but either the value or the message field must be present in a Value object. Valid values for action “set” together with property “Load resource” are: - 1 (Power on) - 0 (Power off) No values are necessary for action “get”, and in this case the message element will be populated with “N/A”.
message	string	This may either be a status modifier for the value or a notification message from the device. message is an optional field, but either the value or the message field must be present in a Value object.

Table 3-20 Value object format from SESP to load resources

3.4.2.3 Message schema

The schema for the message is presented below:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "name": "Sample",
    "type": "object",
    "properties": {
        "messageid": {
            "type": "string",
            "description": "Message identifier"
        },
        "deviceid": {
            "type": "string",
            "description": "Device identifier"
        },
        "action": {
            "type": "string",
            "description": "Action to execute"
        },
        "property": {
            "type": "string",
            "description": "Property identifier"
        },
        "format": {
            "type": "string",
            "description": "Message format including version number"
        },
        "timezone": {
            "type": "string",
            "description": "Time zone for starttime and endttime in values
array"
        },
        "unit": {
            "type": "string",
            "description": "values unit"
        },
        "values": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "starttime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "start time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "endtime": {
                        "type": "string",
                        "format": "date-time",
                        "description": "end time of value in time zone of
timezone (or UTC if missing)"
                    },
                    "value": {
                        "type": "number",
                        "description": "value"
                    },
                    "message": {
                        "type": "string",
                        "description": "status modifier or notification
message"
                    }
                }
            }
        }
    },
    "required": ["deviceid",
    "action",
    "property",
    "values"]
}
```

3.5 Prosumers

3.5.1 Messages from local controller to SESP

A local controller (gateway) from Develco Products will be installed at prosumers (private households with production generators). The local controller has support for several types of devices that will be used for consumption- and production monitoring, powering on and off connected devices, etc.

All metering information (telemetry) from devices that are connected to the local controller, will be sent to the SESP in a periodically manner.

The local controller from Develco Products will be the gateway between the various devices and SESP, and all communication between to and from SESP will take place via the gateway.

An asset register in SESP is populated with detailed information about each connected device. The SESP asset register will be initialized and maintained in a manual manner.

The technical information will be sent to the SESP by posting messages to the Azure Event Hub (described in 1.1.1.1). All telemetry data from the various devices will be stored as time series in SESP.

The Azure event hub is designed so that it works better when receiving smaller data packages frequently rather than receiving large data packets less frequently. As a consequence, there will be created one message for each variable when sending values from the DER to SESP. The message format shall be JSON, and the message is described in a more detailed manner below.

Property name	**Type**	**Optional**	**Description**
deviceid	string	N	This id is a unique identifier for the device.
property	string	N	This uniquely identify the property of the value sent. E.g. “metered energy consumption”, “metered power”, “temperature’, etc. The property may be a descriptive string or a numeric id which is known to the device and the eSmart event hub. This field is optional when there is only one property for the device.
format	string	Y	Identifies the format – including version – of this message. This field is optional.

Property name	Type	Optional	Description
timezone	string	Y	Time zone of the start and end times in the Value object (see below). This is optional, and preferred practice is to include time zone in the time fields.
unit	string	Y	Unit of value. This is optional. E.g. “A”, “Ah”
values	array	N	This is an array of value objects. The value object is described below. The values array is required and must have at least one value object.

Table 3- Local controller message format

3.5.1.1 Value object format for local controller message format

Property name	Type	Description
starttime	string datetime	Start time of the value. This is normally required, but may be omitted for a device continuously sending instantaneous values. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. “2015-04-30T14:32:52Z” If no time zone is given UTC is assumed.
endtime	string datetime	End time of a value that applies for a duration of time (e.g. metered consumption for an hour, maximum temperature for a day). This is optional and would normally not be included for instantaneous values. The time is given as an ISO 8601 date time formatted string, including time zone: E.g. 2015-04-30T14:32:52Z If no time zone is given UTC is assumed.
value	number	The value. Note that “.” is always used as decimal separator for JSON messages. value is an optional field, but either the value or the message field must be present in a Value object.
message	string	This may either be a status modifier for the value or a notification message from the device. message is an optional field, but either the value or the message field must be present in a Value object.

Table 3-21 Value object format for controller

3.5.2 Messages from SESP to local controller

The SESP shall have the possibility to send messages to the devices that are connected to the local controller. Types of messages are either a request for variables, or control signals for demand response (power off / power on).

All SESP messages will be sent to the local controller who will be responsible for re-routing the message in an appropriate format according to what supported by the load resource.

Any control signals from SESP to a controllable device can contain both *starttime* and *endtime* for the control signal, and in that case the local controller will be responsible for the execution of both start and stop of the demand response plan. If the control signal only includes a *starttime*, SESP will be responsible for sending a new message that will reverse the control command when the demand response plan is finished.

3.6 Conclusions

In this document, the APIs functions used to exchange messages inside the EMPOWER system are described. The communication interfaces were identified in D3.3, and they include the interfaces between the SCADA and the DER devices, the interface between the SESP and the SCADA, and the interface between the SESP and the prosumer domestic households. For each of these interfaces, the message format is specified.

In addition, the message content, format and schema are detailed for each of the use cases defined in D4.1. These use cases include the EV charging station, the generation resources, the storage, the load resources and the prosumers.

The outcome of this deliverable is a direct input to the SESP development, which in its second phase will focus on adding support for local controller functions, and will result in SESP v2.0 and deliverable D5.2.

In addition, as communications with DER devices are very depending on technology, there is no API defined for connecting each DER device to the SCADA. However, SCADA interfaces will be adapted to recognize these distributed devices using Modbus protocol, as every manufacturer has its own memory maps, addresses, variables and commands. These register mappings may change depending on the technology and features of the equipment used, so they will be developed during the project execution which is very related to the WP7 and pilot's deployment.

3.7 References

- Azure Event Hub (azure.microsoft.com)
- Azure IoT Hub (azure.microsoft.com)
- Protocol Specification for Modbus exchanges between external system and charging Station – www.Schneider-electric.com
- Technical Note – unSpec Logging in SolarEdge Inverters (Susnpec Alliance)- www.sunspec.org/