The eBADGE Data Model Report – First Version

Deliverable report

Document Information

Programme	**FP7 – Cooperation / ICT**
Project acronym	eBADGE
Project Full Title	Development of Novel ICT tools for integrated Balancing Market Enabling Aggregated Demand Response and Distributed Generation Capacity
Grant agreement number	318050
Number of the Deliverable	D3.1.1
WP/Task related	WP3 / T3.1
Type (distribution level)[1]	PU
Due date of deliverable	30.09.2013 (Month 12)
Date of delivery	14.1.2014
Status and Version	Final, v2.2
Number of pages	45 pages
Document Responsible	Marjan Šterk - XLAB
Author(s)	Marjan Šterk, Staš Strozak - XLAB, Radovan Sernec - TS, Peter Nemček - CG, Gianluigi Migliavacca, Alessandro Zani - RSE, Darko Kramar - ELES, Andraž Šavli - Borzen, Boris Turha - EL, Daniel Burnier de Castro - AIT, Ingo Sauer - Sauper
Reviewers	Radovan Sernec - TS

1

PU	Public
RP	Restricted to other programme participants (including the Commission Services)
RE	Restricted to a group specified by the consortium (including the Commission Services)
CO	Confidential, only for members of the consortium (including the Commission Services)

Issue record

Version	Date	Author(s)	Notes	Status
2.2	08.01.2014	Radovan Sernec, TS	Re-written section on IEC 61850	Final v2.2.
2.1	07.01.2014	Marjan Šterk, XLAB	Re-written section on OPC UA, added glossary, updated table of acronyms	Draft
2.0	13.12.2013	Renata Blatnik, XLAB	Project deliverable template applied	Final v2.0.
1.6	12.12.2013	Radovan Sernec, TS	Reviewed, multiple improvements throughout	Draft
1.5	10.12.2013	Marjan Šterk, XLAB	Re-written general sections according to recommendations from project review; title changed	Draft
1.1	16.10.2013	Marjan Šterk, XLAB	Format partially unified with other deliverables	Final v1.1.
1.0	30.09.2013	Marjan Šterk, XLAB	Text polished before submission	Draft
0.7	27.09.2013	Radovan Sernec, TS	Reviewed, multiple improvements throughout	Draft
0.6	26.09.2013	Marjan Šterk, XLAB	Fixed some inconsistencies in data model	Draft
0.5	24.09.2013	Marjan Šterk, XLAB (with input from all partners)	Improved text throughout	Draft
0.4	23.09.2013	Marjan Šterk, XLAB	Added section on existing standards	Draft
0.3	20.09.2013	Marjan Šterk, XLAB (with input from all partners)	Data model inserted into the report	Draft
0.2	19.09.2013	Marjan Šterk, XLAB	Written up general sections	Draft
0.1	05.09.2013	Marjan Šterk, XLAB	Document outline created	Draft

NOTICE
The research leading to the results presented in the document has received funding from the European Community's Seventh Framework Programme under grant agreement n. 318050.

Table of Contents

Table of Figures

Table of Acronyms

Acronym	Meaning
ACER	Agency for the Cooperation of Energy Regulators
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BSP	Balancing Service Provider
CIM	Common Information Model
CSV	Comma-separated values, a text-based tabular data format
DG	Distributed Generation
DR	Demand response
DSO	Distribution System Operator
ECAR	Electric car
ENTSO-E	European Network of Transmission System Operators for Electricity
GME	Gestore Mercati Energetici, the Italian market operator
GUI	Graphical user interface
HEH	Home Energy Hub
HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Technologies
IEC	International Electrotechnical Commission
IP	Internet Protocol
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
NTP	Network Time Protocol
NYSE	New York Stock Exchange
NYSE MKT	NYSE MKT LLC, formerly known as the American Stock Exchange (AMEX)
OLE	Object Linking and Embedding
OPC	Open Platform Communications (initially OLE for Process Control)
OPC UA	OPC Unified Architecture
OpenADR	Open Automated Demand-Response
OSGP	Open Smart Grid Protocol
PV	Photovoltaics
RES	Renewable Energy Source
SCL	Substation Configuration Language
SOAP	Simple Object Access Protocol
TSO	Transmission System Operator
UTF-8	UCS Transformation Format – 8-bit, a variable-width encoding of characters and strings
UTP	Unshielded Twisted Pair, the most common type of computer network cable
UUID	Universally Unique Identifier
VEN	Virtual End Node
VPP	Virtual Power Plant
VTN	Virtual Top Node
WP	Work Package
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

Glossary

Term	Meaning
.NET	a software framework and run-time environment for multiple programming languages (e.g. C#, Visual Basic.NET), developed by Microsoft
C	a low-level programming language
C++	an object-oriented programming language based on C
data model	a specification of data types (objects) and the fields they contain; in this document refers to the eBADGE data model [eBADGE D3.1.1, 2013]
gateway	synonymous to proxy
Protocol Buffers	a method for serializing structured data, developed by Google
gzip	a file compression/decompression tool
Java	a programming language and corresponding run-time environment developed by Oracle (originally by Sun Microsystems)
JavaScript	an interpreted programming language, mostly used in web applications
market model	a mathematical model of balancing energy market, being developed in eBADGE WP2
market simulator	a software simulator of the market model, being developed in eBADGE WP2
message acknowledgement	a return message that acknowledges that the previous message was received
message bus	a generic name for a messaging proxy, the required server and client software, typically also capable of multicast (one-to-many) communication
message header	the part of a message that contains the message metadata and typically precedes the payload
message payload	the part of a message that contains the actual data and typically follows the header
messaging proxy	an intermediary entity (typically, a server) through which one entity (end-point) can communicate with another without requiring a direct connection
multi-cast	a communication pattern with multiple recipients in a single transmission from the source; one-to-many
namespace	a container (software concept) for a set of identifiers (for example, field names), typically used to prevent naming conflicts
payload encoding	method of serializing structured data into the message payload
proxy	an intermediary entity through which one entity (end-point) can reach another entity in the absence of direct access
Python	a high-level programming language
semantic model	a formal description of a certain domain's semantics
semantics	the meaning of something (messages, commands, fields etc.)
serialization/deserialization	the process of transforming a software object to/from a stream of bytes
super-TSO	the central operator of the international balancing market that maintains the common merit order list

eBADGE Project

The 3rd Energy Package clearly boosts the development of an Integrated European balancing mechanism. In this context, ACER has in 2011 started the development of the Framework Guidelines on Electricity Balancing. It is expected from the ACER statements that Demand Response will play significant role in the future integrated balancing market allowing Virtual Power Plants, comprising Demand Response and Distributed Generation resources to compete on equal ground.

The overall objective of the eBadge project is to propose an optimal pan-European Intelligent Balancing mechanism also able to integrate Virtual Power Plant Systems by means of an integrated communication infrastructure that can assist in the management of the electricity Transmission and Distribution grids in an optimized, controlled and secure manner.

In order to achieve the above overall objective the eBadge project will have four objectives focusing on:

1. Developing the components: simulation and modelling tool; message bus; VPP data analysis, optimisation and control strategies; home energy cloud; and business models between Energy, ICT and Residential Consumers sector;
2. Integrating the above components into a single system;
3. Validating these in lab and field trials;
4. Evaluating its impact.

Project Partners

Telekom Slovenije d.d. – Slovenia
cyberGRID GmbH – Austria
Ricerca sul sistema energetico – RSE Spa - Italy
XLAB Razvoj programske opreme in svetovanje d.o.o. – Slovenia
ELES d.o.o. – Slovenia
Austrian Power Grid – Austria
Borzen, Organizator trga z električno energijo, d.o.o. – Slovenia
Elektro Ljubljana, Podjetje za distribucijo električne energije d.d. – Slovenia
Technische Universitaet Wien – Austria
Austrian Institute of Technology GmbH – Austria
Sauper Umweltdatentechnik GmbH – Austria
SAP AG – Germany
Vaasaett Ltd Ab Oy – Finland

Project webpage

http://www.ebadge-fp7.eu/

Executive Summary

This document defines the first version of the eBADGE data model and encoding used for the communication between all the stakeholders in the pilot project, such as a home energy hub sending usage data to its VPP operator and receiving curtailment commands, or a TSO collecting balancing energy bids and sending activation commands when needed.

We first list some general requirements and then evaluate the suitability of the existing standards in the field (e.g. OpenADR) and further propose the implementation of "green field" solution focused on achieving eBADGE objectives.

Some general conventions are described next, such as avoidance of binary encodings for reasons of human-readability, ease of development, extensibility, and maintainability, the use of JSON, the encoding of standard data types, and extensibility conventions.

A full, formal definition of the data model and encoding is given, containing the list of all the message types required to cover the currently foreseen eBADGE use cases, with complete details and JSON examples. This is accompanied by sequence diagrams for the more complex communication patterns. Also of note is that the related eBADGE public deliverable D3.2.1, "The eBADGE Message Bus", contains a reference Python implementation of the data model.

Finally, we list some of the possible changes in the future versions of this data model.
Note: this is just the first prototype and should not be considered a stable specification.

1. Introduction

The aim of this deliverable and the corresponding project task is to analyze the existing ICT interfaces and standards used in TSOs (ENTSO-E), VPPs, DSOs, Resources (loads, DG, RES), Electricity and Balancing Markets. As each of these stakeholders and its ICT components has its own set of interfaces, the data attributes of all of them have to be harvested and a common data model and communication protocol defined with all the relevant attributes for information exchange between the entities.

The original title of this document is “The eBADGE Data Standard Report – First Version”. However, the choice of that title was not appropriate because formal acceptance by a standardization body was not planned. Rather, the purpose of this data model is solely to enable and unify communication between the entities in the eBADGE pilot; that is, the data model itself is a pilot intended to show one possible approach to this communication.

This deliverable presents the first version of the proposed open data model. The model defines the types of messages, their contents, encoding, meaning, and the typical sequences of messages in certain non-trivial scenarios.

1.1 Approach

This first version of the eBADGE data model was derived by first defining the typical demand-response and balancing energy market scenarios and analyzing the communication patterns. Next, existing standards were reviewed to find out whether eBADGE can benefit from using them, either as-is, or by defining our extensions. Unfortunately, as is argued in this report, this turned out not to be the case; nevertheless, we looked for the features and solutions implemented in these standards that can be re-used when defining our own data model.

All the required message types and contents were extracted from the defined scenarios. During this process, inconsistencies, corner cases and additional possibilities arose, which required refinement of the scenarios. This iterative process resulted in the full message types specification as given in this document.

In parallel, the non-functional requirements such as message volume, reliability, security, and expandability/scalability/extensibility were defined. Although these are mostly relevant for the message bus implementation [eBADGE D3.2.1, 2013], they were followed when defining some general approaches in the data model, such as how the message contents are encoded and how the message acknowledgements are formatted.

1.2 Relation to the Rest of the Project

This deliverable, together with the eBADGE message bus [eBADGE D3.2.1] developed in the same work package, provides the first prototype of the data exchange software for the eBADGE pilot. The requirements for the data model presented here are influenced by and result from the market model and simulator developed in WP2 (data exchange in international balancing energy market), the business use cases developed in WP4 (required data exchange with demand response resources and users) and the home energy hub and cloud prototype specifications, also from WP4 (e.g., performance and other limitations, network security model).

1.3 Outline of This Report

The next section briefly summarizes the requirements and the corresponding division of the model into two communication levels. Section 3 analyzes existing standards related to eBADGE communication. Section 4 describes the chosen approach in general, common to all message types. Sections 5 and 6 contain detailed specifications of all the message types and their contents. Additionally, section 6 illustrates some of the typical market scenarios using sequence diagrams, which help understanding the communication patterns.

2. Requirements

The required communication in the eBADGE pilot can be divided into multiple levels where different entities communicate:

1) energy resources, energy storage, smart meters, home energy hubs (gateways), VPPs, microgrids, DSOs, energy providers exchange messages with home energy usage profiles, VPP activation commands and similar,
2) VPPs and other BSPs (balancing service providers), e.g. traditional generators, send bids to their TSO, who send activation commands back,
3) in an international balancing market based on the TSO-to-TSO model, TSOs forward bids to a common merit order list and coordinate balancing energy allocation from this list.

No messages pass the level border (for example, the home energy hub never communicates directly with a TSO). Each higher level contains less message volume but must be more resistant to attacks and failures. In this document we will refer to the first level as the *home energy hub (HEH) level* and to the remaining two levels as the *market level*.

Ideally, all levels can be covered using the same technology. However, the actual communication channels must be isolated so that a potential attack on a home energy hub does not pose any threat on the market level.

In the eBADGE pilot the maintenance of the common merit order list and the coordination of balancing energy allocations from this list will be simulated through a central market operator. The role of the latter also includes the process of selecting the optimal bids in any given moment not only with regard to its price but also its dispatchability to the target area given the capacities and flows in the network. To encompass this complex role, we refer to this central market operator as a *super-TSO*.

The main purpose of the market level and the simulator is to demonstrate the regulatory and economic feasibility of an integrated balancing market and the increased social welfare that such a market would provide. The implementation challenges beyond this pilot are mostly regulatory and information technology is secondary. On the other hand, the HEH level of the pilot also strives to demonstrate the technical feasibility of using a large number of households or small businesses as demand response resources. It is therefore conceivable that commercial projects following this pilot will strongly consider re-using the HEH-level technology developed here, while for the market level this is far less likely. Consequently, when one technical choice is optimal for the HEH level and another for the market level, we will most often choose the former over the latter.

The full list of required message types and contents that resulted from the requirements collection process was used directly to define this data model. For example, a scenario was envisaged where the VPP subscribes to periodic load reports from the home energy hub, thus the message types *get_periodic_load_report* and *load_report* were defined. As the resulting model is provided in this document, the full requirements would just duplicate the information and are thus not listed here; rather, an overview is given in this section.

2.1 Home Energy Hub Level Requirements

The basic business idea is a VPP aggregating the demand-response potential of many end-users (through HEHs) and selling it as balancing energy on the market. More elaborate scenarios are being developed in WP4 where, for example, telecommunication companies may offer data acquisition from HEHs and data storage, and DSOs and energy providers also take part in end-user-level demand-response. However, for this version of the data model, we shall restrict ourselves to the basic case. Thus, this level of the data model defines how the metering data is communicated from the HEH to the VPP or other relevant entity, how the activation commands are communicated back, and various status inquiries/reports.

We estimate that the demand-response potential of a single home is in the order of 1 kW, so thousands of them must be aggregated to provide megawatts to the balancing energy market. Thus, the data model must be scalable to at least thousands, preferably tens of thousands HEHs per VPP. Since two VPPs can (and most often should)

use completely isolated channels to communicate with their HEHs, the total number of VPPs is not the limiting factor for the HEH-level communications.

The HEH has to be able to send the following reports:

- power usage and generation profile for the whole building,
- power usage and generation profile for each device (load, generator or storage) that is measured individually,
- anomalous events, such as power outages, overvoltages,
- detailed, high-resolution profile (U, I, P, Q, S, harmonics; or a subset of those if not all are being measured),
- predicted load and generation profiles for the smart devices that are able to predict their usage (for example, after a user has set a washing machine cycle, the usage for the whole cycle could be predicted easily),
- HEH status,
- capabilities of the HEH and of individual devices.

These reports are sent either on request or periodically, where the period and other parameters must be remotely configurable.

The VPP must be able to control the HEH and the appliances through the following:

- activation commands for individual devices (i.e. increase/decrease load or generation on the specified device),
- activation commands for the whole household/business, where it is up to the HEH and/or the user how to achieve it,
- tariff (pricing) updates through which the VPP can motivate/boost/incentivize demand response.

The data model must allow the HEH to reject or modify activations. However, depending on the contract between e.g. the home user and the VPP, this may only be allowed in certain limited cases.

2.2 Market Level Requirements

The market-level of the data model will be used to collect bids from BSPs at the TSOs and to send activation commands to the BSPs. On the international market, the model must also cover sharing bids of each TSO with the super-TSO and coordinating the cross-border balancing mechanism. Scalability to tens of TSOs and thousands of BSPs must be ensured.

This part of the model is not intended to replace the existing protocols used to allocate cross-border capacities, control the transmission grid, inform about failures in the energy grid etc.; rather, it should contain just the messages necessary for the implementation of the eBADGE pilot. In particular, the market simulator developed in WP2 will be the pilot entity representing the central market operator (so called super-TSO) and the pilot VPPs will act as the BSPs communicating with the simulator.

The BSPs must be able to, regardless of whether the balancing energy market is national or cross-border:

- send bids to their TSO,
- receive activation commands from their TSO and acknowledge or reject them.

In an international market operated by a super-TSO, the latter must also be able to:

- receive forwarded bids from all TSOs in order to build a common merit order list,
- receive a balancing request from a TSO that detects an imbalance in its area, i.e. a requests to activate bid(s) from the common merit order list,
- send a bid activation order to the TSO that the bid came from, who then forwards it to the BSP,
- inform the imbalanced TSO whether their request was granted or not,

- it must be possible to express imbalance netting.

How the super-TSO is informed about the network state (cross-border capacities, flows etc.) is beyond the scope of this data model. In the eBADGE pilot the network state will be simulated by the market simulator while in a production environment existing communication protocols would most probably be used to exchange network state data.

For an illustration please refer to the sequence diagrams in Sections 6.2 and 6.3.

3. Analysis of Existing Standards

Currently, the data standards related to the HEH level communications and those related to the market level are completely different. The home level only started evolving with the Smart Grid initiatives, thus the standards are relatively recent and in line with modern software patterns and technologies. On the other hand, the market level is currently a mix of various legacy and more modern protocols, many of which are defined by individual software/system vendors rather than a standardization consortium.

This section analyzes the applicability of the most important existing standards to eBADGE. Unfortunately, as it turns out, no existing standard fits the requirements perfectly, thus any of them would have to be extended to the point of losing direct compatibility. Additionally, many of the existing standards are quite complex to implement, and it makes little sense to spend development time on implementation of large standards without the compatibility benefits. Finally, many smart grid approaches, data standards and middlewares are competing and it is far from certain which ones will prevail [DR comparison, 2011; Martínez et al, 2013], thus it is not advisable for this pilot project to invest a significant effort into adoption of standards whose future is uncertain unless there are other benefits.

3.1 Smart Grid Data Standards

3.1.1. iGorenje

iGorenje [iGorenje v15, 2012] is a simple HTTP/SOAP API for demand response intended primarily for smart home appliances. As a HTTP-based protocol, it requires either inbound connectivity into the clients (HEHs in our case) or periodic polling by the clients (asking the server "Do you have a command for me?"). Inbound connectivity is to be avoided for reasons of security, IP address scarcity etc. Periodic polling introduces performance problems for all but very small deployments, as experience from Sauper shows.

Content-wise, iGorenje fits the eBADGE requirements very well and could easily be extended by adding new methods to the API. Existing iGorenje semantics is thus a good starting point for definition of HEH-level messages. However, even if we re-implemented the semantics exactly but using another communication channel rather than HTTP, the software and devices based on our data model would not be compatible with iGorenje any more.

3.1.2. OpenADR, OpenADR 2.0

The original OpenADR (Open Automated Demand-Response) focuses on price-based demand-response, which is not the focus of eBADGE. In addition it covers utility-to-aggregator communication and bidding, which are not relevant to the HEH level of eBADGE [OpenADR, 2009; DR comparison, 2011].

OpenADR 2.0 is a better but not a perfect fit. An eBADGE HEH is approximately analogous to an OpenADR VEN (virtual end node) and a VPP could be a VTN (virtual top node). In contrast to iGorenje, OpenADR 2.0 supports communication over XMPP in addition to HTTP, thus solving the above-mentioned problems of HTTP [OpenADR 2.0, 2013].

We have analyzed it in detail and defined OpenADR 2.0 (profile B) XML equivalents of the messages required in eBADGE: an activation command (eiEventSignal LOAD_DISPATCH in OpenADR terminology), a rejection of the activation (eventResponse optOut) and an electricity pricing update (eiEventSignal ELECTRICITY_PRICE). These equivalents contain many obligatory fields (XML tags) that are not needed for our purposes, thus the messages are quite long. For example, our example OpenADR activation command takes 2.5 kB (847 bytes if compressed with gzip) while with our data model it only uses 178 bytes (not further compressible with gzip). The ratio is similar for the other mentioned abovementioned message types. When a VPP needs to send thousands of such messages at once to curtail loads by several megawatts, the size difference is significant. If, on the other hand, the fields that we do not use were simply left out, the HEH and the software developed in the eBADGE pilot would not be interoperable with third-party OpenADR 2.0 equipment.

Some of the messages required in eBADGE cannot (at least as of the time of writing of this report) be expressed in OpenADR 2.0, such as HEH's predictions of load, generation, and curtailment capacity; they could be mapped either to EiAvail, which is only planned for future releases, or to the Report type Forecast, which was planned in the

draft but later removed. These could be implemented as extensions, which OpenADR allows. There are also many OpenADR message types that eBADGE does not need and would thus be ignored by our HEHs. Again, interoperability would not be achieved.

Nevertheless, certain good practices from OpenADR are used in the eBADGE data model. For example, each activation order contains a *modification_count* field so that an issued order can be changed without the need for atomic "cancel-this-activation-and-activate-this-new-one" transactions.

3.1.3. Open Smart Grid Protocol

OSGP is a large, complex binary standard for smart meters [OSGP, 2012]. For example, to get the end device's status, the request (once decrypted, as OSGP uses encrypted commands) is
30 00 03 8E 56 E4 21 D6 2F 43 91 0C 48 A3 CC,
where:

- 30 is the OSGP command "Full Read",
- 00 03 is the ID of the table "BT03: End Device Mode Status",
- 8E 56 E4 21 is the OSGP sequence number of the request.

The response is an 8-byte sequence that encodes one integer and more than 40 boolean fields with a pre-defined meaning. As such, OSGP is difficult to implement and impossible to extend without breaking compatibility, which is in stark contrast to the eBADGE requirerments.

3.1.4. OPC

The original OPC is a communication standard used in many fields of industrial automatics, including demand-response. The acronym originally stood for OLE (Object Linking and Embedding) for Process Control but was later re-defined as Open Platform Communications.

Being a communication standard, it only defines ways to access attributes (registers) and issue commands. The semantics of these attributes and commands are application/equipment-specific and each (type of) device requires additional configuration to define which registers can be accessed, what the data read from them means and what the commands that can be issued mean. It also requires Microsoft Windows. Although these properties make OPC unsuitable for eBADGE, certain existing OPC devices can be supported through separate protocol gateway services that we plan to develop as a part of the eBADGE message bus.

3.1.5. OPC UA

OPC UA (OPC Unified Architecture) is a completely new standard, although it also includes provisions for compatibility with the original OPC that it intends to replace. Unlike the original OPC, OPC UA is cross-platform [OPC UA web page]. It defines both a binary communication protocol as well as an HTTP/SOAP-based one. Multiple server implementations and client bindings for multiple programming languages are available from different software vendors.

OPC UA also contains the ability to model the semantics of the data objects. It allows data, alarms and events, and their history to be integrated into a single OPC UA Server. For example, OPC UA servers are able to represent a temperature transmitter as an object that is composed of a temperature value, a set of alarm parameters, and a corresponding set of alarm limits. Such models may be standardized for certain fields, which is then termed an *OPC UA companion standard* [OPC UA Part I, 2009].

There are multiple OPC UA server implementations, including royalty-free, open-source ones from the OPC foundation and proprietary servers from various vendors. Similarly, client libraries for C/C++, Java, .NET, and some other programming languages are available.

OPC UA has been considered as a transport layer for OpenADR [OpenADR Transport Layer] but was then rejected. The evaluation of OPC UA from the aspect of OpenADR would be very interesting for eBADGE as well; however, we did not find such an evaluation in the publicly available literature.

In our case, adopting OPC UA would imply both defining the eBADGE data model as an OPC UA semantic model and using OPC UA for all (or most of) the communication between the entities, that is, as a message bus equivalent. It has to be noted that we do not plan to use any existing OPC UA-based products within the pilot. Thus, rather than investing into the adoption of OPC UA, we have decided for a more straight-forward approach that will allow a rapid implementation of our requirements and then let the consortium concentrate on the development of innovative VPP functionalities and business practices, the demonstration of which is a major goal of the project. Nevertheless, certain strengths of OPC UA are mimicked in our data model, such as automatic discovery of the home energy hub's capabilities through the *get_capabilities* request.

3.2 Energy Market Data Standards

The energy markets mostly use data exchange formats defined by the software used in the individual markets, without following any recognized standards. On the other hand, for the control of the transmission grids, older, well-known, stable standards are mostly used.

3.2.1. IEC 60870-5-101, 60870-5-104

These two IEC standards define how to exchange numbered lists of simple data points over various physical layers while optimizing the use of bandwidth and hardware [IEC 6*, 2008].

3.2.2. IEC 61850

IEC 61850 [IEC61850, 2013] strives to not only replace 60870-5-104 but also to significantly expand the scope. It is a well-established IEC standard with first ratified documents from 2002 and covering aspects of communication networks and systems for substation, power utility and feeder equipment automation. Thus it supports integration and networked energy (power) systems built from multiple vendors to perform protection, monitoring, automation, metering, and control. It defines the transport, session, presentation, and application level, as shown in Figure 1.

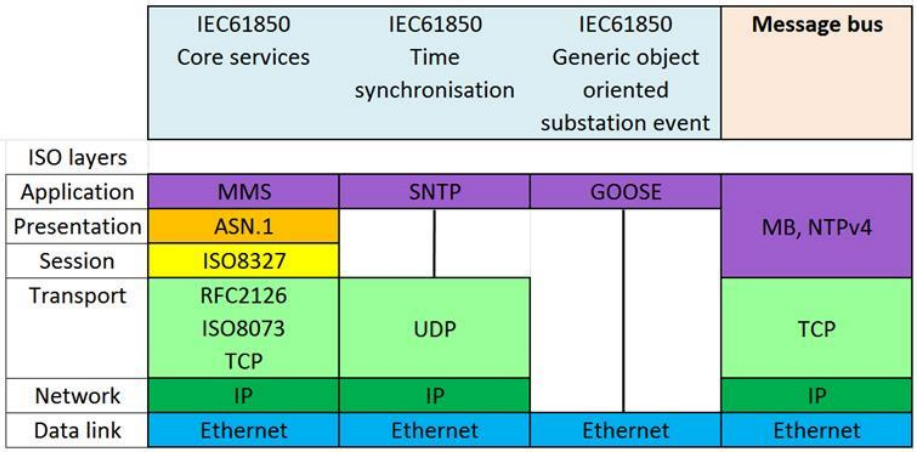

Figure 1: IEC 61850 vs eBADGE message bus layered on the ISO protocol stack.

IEC 61850 targets physical infrastructure elements for power distribution, whereas eBADGE is concerned with communication between business entities. It is clear that on lower layers completely different set of constraints and goals exist for each, thus different implementation options emerge. As such these layers of IEC 61850 are not relevant for the discussion of the eBADGE data model.

The scope of IEC 61850 was originally focused only on the “inside” the substation, later widened to inter substations and only recently there is activity to promote it also for distributed energy generators [Sučić, 2012; IEC SmartGrid]. Notable is the object-oriented, system-of-systems approach for substation automation where standardized device models are using names (and not object/register numbers), defined by Substation Configuration Language (SCL). From this aspect it is similar to OPC UA. On the other hand it is very focused on substation communication supporting even multi-cast messaging for inter-relays (used for protection). It is also possible to use Common Information Model (CIM) descriptions but mapping from SCL is required.

The data semantics provided by IEC 61850 are closely related to the utility devices such as substations, wind power plants, etc. [Sučić, 2012]. The top parent class is a Server, i.e. a *device controller*, consisting of one or more *logical devices*, which combine several *logical nodes*. A logical node represents a device functionality, such as a wind turbine (labelled as *WTUR* in IEC 61850), a temperature measurement point (labelled as *SMTP*), etc.

Given the elaborate, top-down approach of IEC 61850, adopting it for the purpose of communicating the limited number of message types envisaged in eBADGE would be advisable if integration of existing artefacts based on it were planned in the pilot. In our case, on the other hand, a more straightforward implementation of our focused requirements suffices and will allow us to better concentrate on the project content. Furthermore, although open-source implementations exist, the standard itself is subject to licensing fees.

3.2.3. Existing Market Applications

The trading applications mostly provide an interface to enter bids manually and to import them from a simple format. For example, the Slovenian intra-day market uses the ComTrader application [BSP web page]. The bids can be entered through a web GUI, imported from XML and CSV formats, or submitted through an API that also uses the XML format. Each bid contains required fields such as whether it is a buy or sell bid, quantity, price, and product ID. Additionally, a bid may include a date/time when it will automatically expire, a text/comment field for bidder's notes, sender and receiver IDs. Similarly, bids for the GME market can be entered manually or imported from XML.

3.2.4. NYSE UTPDirect

As an example of a high-volume trading standard, we also analyzed the API for NYSE and NYSE MKT Cash Equities Markets [NYSE, 2012]. It is a high-performance, low-level binary format operating, as the name suggest, directly above the physical layer. For example, the first byte marks the type, the next two the length, the rest is message body. Prices are represented as fractions with integer numerators and power-of-ten denominators. As an extremely high volume, low level, binary standard, it is not at all suitable for eBADGE.

4. General Approach

This section defines the encoding of message contents and other general technical choices and conventions that the whole data model follows.

4.1 Specificity versus Generality

The required data can be encoded either in specific messages with fully defined semantics (meaning and expected action on the receiver's side), general message types complemented by some kind of "model of the system" that defines the semantics, or a compromise between the two. In this version we chose the first option so that the data model stands for itself and implementation of the communicating software can straightforwardly follow this specification.

In the future versions we may generalise the approach to a certain extent. For example, non-electricity data such as temperature measurements may very well be relevant for load forecasting and for the calculation of demand response potential of a home. Instead of adding new message types for such data we may generalise the data model to sending any kind of "signal" or "measurement", and extend the *device_capabilities* message with the information of supported signal types.

4.2 Message Payload Encoding

The message contents can be encoded in one of the many binary and text formats.

4.2.1. Binary Encoding

Binary formats are compact and, depending on whether a native encoding of the used programming language is used, serialization and de-serialization may be extremely efficient. On the other hand, lack of human readability lengthens the learning process for developers and, more importantly, makes debugging and diagnosing problems significantly more difficult. Human readability may seem undesired from the security aspect; however, since the model has to be public anyway, non-human-readable messages will not deter a determined attacker and may even make it harder to detect forged or malicious messages.

If our model defined the exact binary representation, a major effort would be required to ensure consistency and to develop the encoding and decoding software. On the other hand, if it relied on a standard serialization method such as Java object serialization or Google Protocol Buffers, it would be difficult to ensure interoperability even with the current versions of various programming languages, while maintaining it for the foreseen lifetime of the prototypes and artefacts based on this model would be even harder.

Finally, binary encodings are less extensible because adding a single data field may break the format of the whole message. Extensibility points have to be clearly defined in advance, which further complicates the model and lessens the compactness advantage of binary formats.

4.2.2. Text-Based Encoding: XML vs. JSON

The two most widely used text-based encodings are XML and JSON. XML is traditionally used in business-level software because it was the first widely accepted text-based structured encoding and because the validity of any XML document can be verified against a suitable XML schema, which describes the allowed format of the document. The schema can also be used to generate document-to-class mapping for a number of widely used programming languages. XML namespaces also solve the problem of distributed extensibility.

JSON, at first mostly used in small web-based applications due to easy conversion to/from JavaScript objects, is becoming increasingly popular in other fields. Although both XML and JSON can be made quite compact by choosing short field/tag names, JSON is still more compact. For example, a small message containing two string and six integer fields might be 158 bytes as a (namespace-less) XML, 128 bytes as JSON, 40 bytes serialized with Google Protocol Buffers and 28 as raw binary.

Finally, JSON's good integration with programming languages like Python and JavaScript eases agile software development. Thus we have based this first version of the data model on JSON.

4.2.3. Field names

The choice of field names in JSON (or, likewise, XML) is always a compromise between readability and compactness. In this draft we stressed the former over the latter. If later the testing reveals that shorter messages would be beneficial and that using compression on top of them is not recommended, we may consider either defining shorter synonyms for the most often used fields or shortening those field names altogether.

All message types and field names shall contain only the alphanumerical characters plus the underscore [A-Za-z0-9_] and shall begin with [A-Za-z] so that they can be mapped to object/structures in standard programming languages.

All field names are case-sensitive, except when explicitly defined otherwise. However, in order to allow easy mappings to object/structures in case-insensitive programming languages, no two field names shall differ only in the case.

4.2.4. Encoding of Standard Data Types

As per JSON standard, all strings in eBADGE messages shall be UTF-8 encoded.

All JSON numbers are floating-point; nevertheless, the eBADGE data model may define certain fields to be integer, or positive, or contain at most two decimals etc. The JSON standard does not support the IEEE 754[2] special values NaN, Inf, or -Inf. Although some implementations add support for these values, our data model does not use them in order to remain fully JSON-compliant.

All numerical fields in eBADGE messages have defined units. Use of other units is forbidden, so that the receivers of messages need not support any conversion. In this version, all prices and money amounts are in € and other currencies are not supported.

JSON does not define date or time formats, thus all times are encoded as strings according to the ISO 8601[3]. Although the latter allows „local time" without a specified time-zone, this is forbidden in eBADGE messages to avoid confusion; all times must always specify the time-zone explicitly. Any time zone may be used since most programming languages offer easy-to-use conversion between them.

4.2.5. Extensibility

To ensure distributed extensibility, any party may introduce additional message types or additional fields in existing messages. This version of the data model does not define how the receiver should react to unknown extensions, so the sender must not include them unless it knows for certain that the receiver supports them.

The reserved prefix for extensions is "ext_" – no message type nor field name in the core data model will start with "ext_". The extension name should always include a unique identifier of the party defining the extension, such as their internet domain without special characters. For example, a new message type for water usage profile could be named "ext_at_sauper_water_profile" and a message field containing extended VPP description could be named "ext_com_cybergrid_vpp_extended_desc".

There are currently multiple efforts related to JSON extensibility: JSON-LD[4], JSON Schema[5], and the JSON namespace proposal[6]. If by the time the eBADGE data model is finalized one of these or another solution become widely accepted, we may adopt it and retire this naming convention.

[2] A floating-point arithmetic standard.
[3] A standard defining encoding of times, dates, and periods in a text-based format.
[4] http://json-ld.org/
[5] http://json-schema.org/
[6] http://www.watersprings.org/pub/id/draft-saintandre-json-namespaces-00.txt

4.3 Sender and Receiver Identification

The messages will always travel over some kind of communication channel; in the eBADGE project, this channel will be the message bus developed in task T3.2. Most standard communication channels either connect a single sender to a single receiver or identify the sender and the receiver of each message in the message headers, thus it is not necessary to specify them explicitly in the message.

4.4 Asynchronous and Synchronous Communication

This data model is agnostic to whether a synchronous communication protocol (i.e. immediate response built into the protocol, such as HTTP) or an asynchronous one (such as AMQP) is used. Whenever an asynchronous protocol is used and a message requires a response, the latter can be provided by a generalized message type *response* (see Section 6.1.2 for the exact specification).

4.5 Advice to Implementers

Please note that this is the first, draft version of the data model that has never been used in a production environment. As such, it will certainly change in the next two years. Implementations based on this version will almost certainly require updates to be compatible with later versions, and the updates may have to be major.

On a positive side, a reference Python implementation is available in the eBADGE project public deliverable D3.2.1, entitled "The eBADGE Message Bus, First Intermediate Version".

5. Specification of Messages at the Home Energy Hub Level

The flow of messages at the HEH level is quite simple. Only the following scenarios are possible:

- the VPP or other controlling entity sends requests for reports or activation orders, to which the HEH responds,
- the HEH autonomously sends certain data measurements and control reports, if configured to do so,
- the VPP sends pricing data.

The rest of this section specifies in detail all the message types at this level. For each message type its full contents are given and fields described based on an example.

5.1 Energy consumption/generation

5.1.1. Message type: get_load_report

Meaning: request for an individual load report (sent by VPP).

Field	Description/comment	Example value
from	start of period for which the report is to be returned	"2013-07-21T10:00:00.000Z"
to	end of period for which the report is to be returned	"2013-07-21T10:30:00.000Z"
resolution	in seconds	120
device	the ID of the device to report for; use null for "total"	"WaterHeater01"

Raw JSON example: {"msg":"get_load_report", "from":"2013-07-21T10:00:00.000Z", "to":"2013-07-21T10:30:00.000Z", "resolution":120, "device":"WaterHeater01"}

5.1.2. Message type: get_periodic_load_report

Meaning: request for periodic load reports; overrides previous setting, default is no reports (sent by VPP).

Field	Description/comment	Example value
interval	in seconds, -1 to disable periodic reports	900
first_from	start of period that the first report should cover; if null, first report period may start at any time between 'now' and 'now+interval'	"2013-07-21T10:00:00.000Z"
resolution	in seconds	120
device		null

Raw JSON example: {"msg":"get_periodic_load_report", "interval":900, "first_from":"2013-07-21T10:00:00.000Z", "resolution":120, "device":null}

5.1.3. Message type: load_report

Meaning: load report, sent either as response to individual request or periodically (sent by HEH).

Field	Description/comment	Example value
from	start of period covered by this report	"2013-07-21T10:00:00.000Z"

to	end of period covered by this report	"2013-07-21T10:30:00.000Z"
resolution	in seconds	120
device	the ID of the device this report is for; null for "total"	"WaterHeater01"
load	each value is average load in kW for last "resolution" seconds	[0.12,0.17,0.33,0]

Raw JSON example: {"msg":"load_report", "from":"2013-07-21T10:00:00.000Z", "to":"2013-07-21T10:30:00.000Z", "resolution":120, "device":"WaterHeater01", "load":[0.12,0.17,0.33,0]}

If the period requested for the report (get_load_report.(to-from) or get_periodic_load_report.interval) is not a multiple of resolution then the actual reported period's shall be longer, i.e. the 'to' time will be later than requested.

5.1.4. Message type: get_generation_report

Meaning: request for an individual generation report for a single generator or for the total (sent by VPP).

Field	Description/comment	Example value
from	start of period for which the report is to be returned	"2013-07-23T16:10:00.000Z"
to	end of period for which the report is to be returned	"2013-07-23T17:20:00.000Z"
resolution	in seconds	120
device	the ID of the generator to report for; null for "total"	"PV02"

Raw JSON example: {"msg":"get_generation_report", "from":"2013-07-23T16:10:00.000Z", "to":"2013-07-23T17:20:00.000Z", "resolution":120, "device":"PV02"}

5.1.5. Message type: get_periodic_generation_report

Meaning: request for periodic generation reports; overrides previous setting for this generator or total, default is no reports for any generators nor totals (sent by VPP).

Field	Description/comment	Example value
interval	in seconds, set to -1 to disable periodic reports	300
first_from	start of period that the first report should cover; if null, first report period may start at any time between 'now' and 'now+interval'	"2013-07-23T16:10:00.000Z"
resolution	in seconds	30
device	the ID of the generator to set the period for; null for "total"	"ECAR01"

Raw JSON example: {"msg":"get_periodic_generation_report", "interval":300, "first_from":"2013-07-23T16:10:00.000Z", "resolution":30, "device":"ECAR01"}

5.1.6. Message type: generation_report

Meaning: generation report, sent either as response to individual request or periodically (sent by HEH).

Field	Description/comment	Example value
from	start of period covered by this report	"2013-07-23T16:10:00.000Z"
to	end of period covered by this report	"2013-07-23T17:20:00.000Z"
resolution	in seconds	30

generation	each value is average generation power in kW for last "resolution" seconds	[1.22,0.98,0.78,1.03]
device	the device being reported for, null for "total"	"PV02"

Raw JSON example: {"msg":"generation_report", "from":"2013-07-23T16:10:00.000Z", "to":"2013-07-23T17:20:00.000Z", "resolution":30, "generation":[1.22,0.98,0.78,1.03], "device":"PV02"}

5.1.7. Message type: get_energy_events

Meaning: request for report on energy-related anomalous events (sent by VPP).

Field	Description/comment	Example value
from	start of period for which the report is to be returned	"2013-07-23T16:10:00.000Z"
to	end of period for which the report is to be returned	"2013-07-23T17:20:00.000Z"
severity	minimum severity to include in report	1

Raw JSON example: {"msg":"get_energy_events", "from":"2013-07-23T16:10:00.000Z", "to":"2013-07-23T17:20:00.000Z", "severity":1}

5.1.8. Message type: get_energy_events_realtime

Meaning: request to report events immediately when they happen (sent by VPP).

Field	Description/comment	Example value
severity	minimum severity to report immediately; set very high to disable real-time reporting	2

Raw JSON example: {"msg":"get_energy_events_realtime", "severity":2}

5.1.9. Message type: energy_events

Meaning: report about energy-related events, sent automatically on severe events or on request (sent by HEH).

Field	Description/comment	Example value
events	list of events	[{"severity":2, "type":"voltage_low", "start_time":"2013-07-23T16:33:56.345Z", "end_time":"2013-07-23T16:33:58.645Z"}]

Raw JSON example: {"msg":"energy_events", "events":[{"severity":2, "type":"voltage_low", "start_time":"2013-07-23T16:33:56.345Z", "end_time":"2013-07-23T16:33:58.645Z"}]}

Details for field: events (array)

Field	Description/comment	Example value
severity	higher is more severe	2
type	possible values: voltage_low, voltage_high, frequency_low, frequency_high; other values may be defined in next version of data model	"voltage_low"
start_time	start of event (when e.g. voltage dipped below threshold)	"2013-07-23T16:33:56.345Z"

end_time	end time (when voltage climbed over threshold and then stayed there for some time)	"2013-07-23T16:33:58.645Z"

5.1.10. Message type: get_electricity_profile

Meaning: request for a detailed profile of (a subset of) quantities that the HEH measures (sent by VPP).

Field	Description/comment	Example value
from	start of period for which the profile is to be returned	"2013-07-23T16:33:00.000Z"
to	end of period for which the profile is to be returned	"2013-07-23T16:35:00.000Z"
resolution	in seconds; must be a multiple of measurement board resolution	2
device	the device to report for, null for "total"	null
fields	list of fields to report; possible values include U [V], I [A], P [kW], Q [kVAr], S [kVA], harmonics from H1 to H50 [%]; field names are case-insensitive	["I", "P"]

Raw JSON example: {"msg":"get_electricity_profile", "from":"2013-07-23T16:33:00.000Z", "to":"2013-07-23T16:35:00.000Z", "resolution":0.5, "device":null,"fields":["I", "P"]}

Such a profile will typically be requested for periods when anomalous events occurred. The HEH may only store the profile for a few hours, so requests for older data may be ignored.

5.1.11. Message type: electricity_profile

Meaning: detailed HEH measurement for requested device/total (sent by HEH).

Field	Description/comment	Example value
from	start of period that this profile covers	"2013-07-23T16:33:00.000Z"
to	end of period that this profile covers	"2013-07-23T16:35:00.000Z"
resolution	in seconds	0.5
device	the device this report is for or null for "total"	null
profile	array of measurements; each element contains fields requested in the get_electricity_profile request	[{"i":[0.003,8.12,0], "p":[0.001,1.892,0]}, {"i":[0.002,7.93,0], "p":[0,1.887,0]}]

Raw JSON example: {"msg":"electricity_profile", "from":"2013-07-23T16:33:00.000Z", "to":"2013-07-23T16:35:00.000Z", "resolution":0.5, "device":null,"profile":[{"i":[0.003,8.12,0], "p":[0.001,1.892,0]}, {"i":[0.002,7.93,0], "p":[0,1.887,0]}]}

Details for field: profile (array)

Field	Description/comment	Example value
u	voltage [V] on each phase	[238.12,234.33,241.09]
i	current [A]	[0.003,8.12,0]
p	real power [kW]	[0.001,1.892,0]
q	reactive power [kVAr]	[0,1.14,0]
s	apparent power [kVA]	[0.003,8.2,0]
h1	first harmonic [%]	[0.08,0.073,0.102]

h2	second harmonic [%]	[0.078,0.076,0.069]
hX	...	[42,181,1087]

All the fields are optional. All the profile entries in the same electricity_profile should contain the same set of elements, that is, the set requested in get_electricity_profile.

For single-phase devices, only one of the three values will be non-null. The voltages for each device will (presumably) be equal to the voltage of the building.

5.2 Predicted profiles

5.2.1. Message type: get_predicted_load_profile

Meaning: request for predicted load profile (sent by VPP).

Field	Description/comment	Example value
from	start of period to send profile for	"2013-07-23T16:10:00.000Z"
to	end of period to send profile for	"2013-07-23T17:20:00.000Z"
device	the ID of the device; use null for "total"	"ECAR01"

Raw JSON example: {"msg":"get_predicted_load_profile", "from":"2013-07-23T16:10:00.000Z", "to":"2013-07-23T17:20:00.000Z", "device":"ECAR01"}

5.2.2. Message type: predicted_load_profile

Meaning: predicted load profile (sent by HEH).

Field	Description/comment	Example value
to	the end of last section in the "profile" array	"2013-07-24T04:00:00.000Z"
device	the ID of the device or "null" for total	"ECAR01"
profile	list of profile sections; profile is constant within each section	[{"from":"2013-07-23T16:00:00.000Z", "load":5.5, "potential":[0,5.5]}, {"from":"2013-07-23T20:10:00.000Z", "load":3.4, "potential":[0,5.5]}]

Raw JSON example: {"msg":"predicted_load_profile", "to":"2013-07-24T04:00:00.000Z", "device":"ECAR01", "profile":[{"from":"2013-07-23T16:00:00.000Z", "load":5.5, "potential":[0,5.5]}, {"from":"2013-07-23T20:10:00.000Z", "load":3.4, "potential":[0,5.5]}]}

There is no "from" field; the profile starts at the "from" of the first "profile" entry.

Details for field: profile (array)

Field	Description/comment	Example value
from	start of first section must equal the value of "from" in request	"2013-07-23T16:00:00.000Z"
load	predicted consumption in kW from "from" until next "from" or until "to"	5.5
potential	potential; in this case, consumption can be decreased to 0 but cannot be increased	[0,5.5]

5.2.3. Message type: get_predicted_generation_profile

Meaning: request for predicted generation profile (sent by VPP).

Field	Description/comment	Example value
from	start of period to send profile for	"2013-07-23T16:10:00.000Z"
to	end of period to send profile for	"2013-07-23T17:20:00.000Z"
device	the ID of the device; use null for "total"	"ECAR01"

Raw JSON example: {"msg":"get_predicted_generation_profile", "from":"2013-07-23T16:10:00.000Z", "to":"2013-07-23T17:20:00.000Z", "device":"ECAR01"}

5.2.4. Message type: predicted_generation_profile

Meaning: predicted generation profile (sent by HEH).

Field	Description/comment	Example value
to	the end of last section in the "profile" array	"2013-07-24T04:00:00.000Z"
device	the ID of the device or "null" for total	"ECAR01"
profile	list of profile sections; profile is constant within each section	[{"from":"2013-07-23T16:00:00.000Z", "generation":0, "potential":[0,0]}, {"from":"2013-07-23T20:10:00.000Z", "generation":0, "potential":[0,1.7]}]

Raw JSON example: {"msg":"predicted_generation_profile", "to":"2013-07-24T04:00:00.000Z", "device":"ECAR01", "profile":[{"from":"2013-07-23T16:00:00.000Z", "generation":0, "potential":[0,0]}, {"from":"2013-07-23T20:10:00.000Z", "generation":0, "potential":[0,1.7]}]}

Details for field: profile (array)

Field	Description/comment	Example value
from	start of first section must equal the value of "from" in request	"2013-07-23T16:00:00.000Z"
generation	by default, ECAR does not inject energy into the grid	0
potential	during first phase, the ECAR can only charge	[0,0]

5.3 Activations and tariff updates

5.3.1. Message type: activate

Meaning: activation order (sent by VPP).

Field	Description/comment	Example value
id	a unique ID of this activation order, assigned by the VPP	"938f2b97-314c-49e8-9860-f441df2284a1"
modification_count	increased by VPP each time a modified activation is sent; the	0

	pair uniquely identifies the activation order; if an activation with the same ID but lower modification count was previously accepted, this message implies modification of the accepted activation	
from		"2013-07-24T11:10:20.000Z"
to		"2013-07-24T11:14:55.000Z"
quantity	in kW, positive to increase generation/decrease load, negative for vice versa	3.4
device	device ID; use null for "any/total"	"ECAR01"

Raw JSON example: {"msg":"activate", "id":"938f2b97-314c-49e8-9860-f441df2284a1", "modification_count":0, "from":"2013-07-24T11:10:20.000Z", "to":"2013-07-24T11:14:55.000Z", "quantity":3.4, "device":"ECAR01"}

5.3.2. Message type: accept_activation

Meaning: activation accepted (sent by HEH).

Field	Description/comment	Example value
id	the ID of the activation being accepted	"938f2b97-314c-49e8-9860-f441df2284a1"
modification_count	the modification count of the activation being accepted	0

Raw JSON example: {"msg":"accept_activation", "id":"938f2b97-314c-49e8-9860-f441df2284a1", "modification_count":0}

5.3.3. Message type: reject_activation

Meaning: activation rejected (sent by HEH).

Field	Description/comment	Example value
id	the ID of the activation being rejected	"938f2b97-314c-49e8-9860-f441df2284a1"
modification_count	the modification count of the activation being rejected; if an activation with the same ID but lower modification count was previously accepted, this implies rejection of the modification and preservation of the previously accepted version of the activation	0

Raw JSON example: {"msg":"reject_activation", "id":"938f2b97-314c-49e8-9860-f441df2284a1", "modification_count":0}

5.3.4. Message type: modify_activation

Meaning: suggestion for modification of the activation (sent by HEH); implies rejection of original parameters; if VPP agrees to modification, must send a new activation order with suggested modifications.

Field	Description/comment	Example value
id	the ID of the activation being rejected	"938f2b97-314c-49e8-9860-

		f441df2284a1"
modification_count	the modification count of the activation being rejected	0
from	all fields must be present, including the ones with unchanged values	"2013-07-24T11:11:25.000Z"
to		"2013-07-24T11:14:55.000Z"
quantity		"3.6"
device		"ECAR01"

Raw JSON example: {"msg":"modify_activation", "id":"938f2b97-314c-49e8-9860-f441df2284a1", "modification_count":0, "from":"2013-07-24T11:11:25.000Z", "to":"2013-07-24T11:14:55.000Z", "quantity":"3.6", "device":"ECAR01"}

5.3.5. Message type: contingency_activate

Meaning: contingency activation order; activate by as much as possible up to the specified quantity (sent by VPP).

Field	Description/comment	Example value
id	unique ID	"3640aa93-28a7-420e-aebf-f4a7fc3a08d2"
from	if in past, activate ASAP	"2013-07-24T10:55:13.000Z"
to		"2013-07-24T10:56:18.000Z"
max_quantity	maximum quantity to activate if possible, in kW; null means no upper limit	120

Raw JSON example: {"msg":"contingency_activate", "id":"3640aa93-28a7-420e-aebf-f4a7fc3a08d2", "from":"2013-07-24T10:55:13.000Z", "to":"2013-07-24T10:56:18.000Z", "max_quantity":120}

Details still need to be defined. Does the HEH need to acknowledge a contingency activation? If yes then what's the difference between a normal and a contingency activation?

5.3.6. Message type: contingency_end

Meaning: signals when the contingency activation should end (sent by VPP).

Field	Description/comment	Example value
id	ID of the contingency activation being ended	"3640aa93-28a7-420e-aebf-f4a7fc3a08d2"
end	when to end; use a past date for immediate end	"2000-01-31T23:00:00.000Z"

Raw JSON example: {"msg":"contingency_end", "id":"3640aa93-28a7-420e-aebf-f4a7fc3a08d2", "end":"2000-01-31T23:00:00.000Z"}

5.3.7. Message type: load_price

Meaning: informs HEH about changed price at which the end-user BUYS energy (sent by VPP).

Field	Description/comment	Example value
from		"2013-07-27T23:10:00.000Z"
to		"2013-07-28T00:30:00.000Z"
price	new price in €/kWh	0.138

Raw JSON example: {"msg":"load_price", "from":"2013-07-27T23:10:00.000Z", "to":"2013-07-28T00:30:00.000Z", "price":0.138}

5.3.8. Message type: generation_price

Meaning: informs HEH about changed price at which the end-user SELLS energy (sent by VPP).

Field	Description/comment	Example value
from		"2013-07-27T23:10:00.000Z"
to		"2013-07-28T00:30:00.000Z"
price	new price in €/kWh	0.117
device	the ID of the generator, as the generation price depends on (at least) the type of the generator	"PV01"

Raw JSON example: {"msg":"generation_price", "from":"2013-07-27T23:10:00.000Z", "to":"2013-07-28T00:30:00.000Z", "price":0.117, "device":"PV01"}

5.3.9. Message type: get_all_prices

Meaning: a request for all available pricing info; sent by HEH on first boot and if for some reason it lost the information (sent by HEH).

This message has no content other than the message type field.

Raw JSON example: {"msg":"get_all_prices"}

5.4 Other

5.4.1. Message type: get_status_report

Meaning: a request for current status and all events that happened in the given interval (sent by VPP).

Field	Description/comment	Example value
from		"2013-07-24T10:55:13.000Z"
to		"2013-07-24T10:56:18.000Z"
severity_threshold	events with this and higher severity will be reported	2

Raw JSON example: {"msg":"get_status_report", "from":"2013-07-24T10:55:13.000Z", "to":"2013-07-24T10:56:18.000Z", "severity_threshold":2}

5.4.2. Message type: status_report

Meaning: status report - response to status_request or automatically sent on the most severe events (sent by HEH).

Field	Description/comment	Example value
status	list of possible values will be	"OK"

	defined later	
clock	current time on device, to check if it is in sync	"2013-07-24T10:33:59.873Z"
events		[{"time":"2013-07-23T03:23:12.342Z","severity":4, "type":"network_down"}, {"time":"2013-07-23T03:24:45.932Z", "severity":2, "type":"network_up"}]

Raw JSON example: {"msg":"status_report", "status":"OK", "clock":"2013-07-24T10:33:59.873Z", "events":[{"time":"2013-07-23T03:23:12.342Z", "severity":4, "type":"network_down"}, {"time":"2013-07-23T03:24:45.932Z", "severity":2, "type":"network_up"}]}

Details for field: events (array)

Field	Description/comment	Example value
time		"2013-07-23T03:23:12.342Z"
severity		4
type	possible values: network up/down, power_up/down (HEH power), ???	"network_down"

5.4.3. Message type: set_clock

Meaning: set clock - issued if local clock in status_report is found to be wrong (sent by VPP).

Field	Description/comment	Example value
offset	offset in seconds to add to the HEH's local clock; if null, the HEH should re-sync its clock over default means (ie NTP)	-3600

Raw JSON example: {"msg":"set_clock", "offset":-3600}

5.4.4. Message type: set_smart_mode

Meaning: sets smart/non-smart mode of HEH (sent by VPP, e.g. in an emergency).

Field	Description/comment	Example value
mode	normal, passive (all activations cancelled, all messages except set_smart_mode ignored, local logging still on), off (physical shutdown, requires manual intervention to return to normal)	"passive"
reset	whether HEH should also forget all activations, pricing etc. and reboot itself; ignored if mode=off	false

Raw JSON example: {"msg":"set_smart_mode", "mode":"passive", "reset":false}

5.4.5. Message type: get_capabilities

Meaning: request to report the capabilities of HEH and/or individual devices (sent by VPP).

Field	Description/comment	Example value

device	device ID to report for; null for HEH itself/total over all devices	null

Raw JSON example: {"msg":"get_capabilities", "device":null}

5.4.6. Message type: total_capabilities

Meaning: report about HEH/total capabilities (sent by HEH).

Field	Description/comment	Example value
device_name		"eBADGE Home Energy Hub"
device_version		"1.0"
load_capability	the load will always be inside this interval	[0,13.2]
generation_capability	the generation will always be inside this interval	[0,4.5]
devices	list of devices the HEH knows about	["ECAR01", "PV01", "PV02", "WASHDR01", "device01", "device02"]
can_predict_profile	this HEH cannot predict the total profile for whole home	false
can_predict_curtailment_capacity	this HEH cannot predict the total curtailment capacity of all the devices	false
ext_com_cybergrid_vpp_extended_description	a third-party extension field of the message, identified by ext_company_domain_ prefix	"some non-standard stuff"

Raw JSON example: {"msg":"total_capabilities", "device_name":"eBADGE Home Energy Hub", "device_version":"1.0", "load_capability":[0,13.2], "generation_capability":[0,4.5], "devices":["ECAR01", "PV01", "PV02", "WASHDR01", "device01", "device02"], "can_predict_profile":false, "can_predict_curtailment_capacity":false, "ext_com_cybergrid_vpp_extended_description":"some non-standard stuff"}

5.4.7. Message type: device_capabilities

Meaning: report about device capabilities (sent by HEH).

Field	Description/comment	Example value
device	signals that this is about the device ECAR01	"ECAR01"
classes	subset of: consumer, generator, storage	["consumer", "storage"]
type	type of device; hierarchical description with standardized possible values	"car/electric"
device_name	a descriptive name, e.g. commercial name of a car make and model	"Electric Cars Model E1"
version		"0.9 beta"
ext_com_electriccars_ecar_battery_capacity	extension example, in this case car battery capacity in	14.2

	kWh, extensions defined by the company Electric Cars	
load_capability		[0,6.6]
generation_capability		[0,6.6]
can_predict_profile	this device can predict its profile (how it communicates the profile to the HEH is out of the scope of this document)	true
can_predict_curtailment_capacity	this device can predict its curtailment capacity	true

Raw JSON example: {"msg":"device_capabilities", "device":"ECAR01", "classes":["consumer", "storage"], "type":"car/electric", "device_name":"Electric Cars Model E1", "version":"0.9 beta", " ext_com_electriccars_ecar_battery_capacity":14.2, "load_capability":[0,6.6], "generation_capability":[0,6.6], "can_predict_profile":true, "can_predict_curtailment_capacity":true}

6. Specification of Messages at the Market Level

The message flow at this level is more complicated, partly because of stricter consistency requirements and partly because the market model is also not trivial. We can divide it into the following scenarios:

1. On the national balancing reserve market, the TSO collects balancing reserve bids from the BSPs until gate closure. It then selects the appropriate subset of bids to accept, informs the respective TSOs about it, and informs all the BSPs that the market has been cleared.

2. On the national balancing energy market, the TSO collects balancing energy bids from the BSPs until gate closure and builds the (national) merit order list. When imbalance is detected, the BSP of the first bid from the list is instructed to activate the balancing energy that is the subject of that bid.

3. On the international balancing energy market, all the TSOs collect balancing energy bids from their respective BSPs and forward (some or all of) them to the super-TSO (the international market operator), who builds a common merit order list. When a TSO detects imbalance, it requests balancing from the super-TSO. The super-TSO may decide that it can be netted with another TSO's imbalance and/or that some bids need to be activated, taking into account both the merit order and the transmission capacities. It then informs the unbalanced TSO(s) that the balance is/will be (partly or fully) restored and the TSO(s) where the bid(s) come from that the bid(s) need to be activated.

The following subsections detail the messages for these three scenarios and also illustrate them with sequence diagrams.

6.1 Balancing reserve market

Balancing reserve market will only be modelled later in the project. However, some messages are the same as in the balancing energy market so they are listed here.

6.1.1. Message type: balancing_reserve_bid

Meaning: balancing reserve bid (sent by BSP to its TSO).

Field	Description/comment	Example value
msg_id	UUID of message assigned (created) by BSP; all market-level messages that require a response have msg_id, others do not	"938f2b97-314c-49e8-9860-f441df2284a1"
bid_id	UUID of bid assigned by BSP, used to refer to it later (e.g. to accept or cancel it)	"3640aa93-28a7-420e-aebf-f4a7fc3a08d2"
product	product description - described below	{"positive":true, "start":"2013-07-21T10:00:00.000Z", "end":"2013-07-21T10:15:00.000Z"}
quantity	in MW	45.3
divisible	false for all-or-nothing; optional, default is true	true
reserve_price	reserve price in €/MWh	1.4
energy_price	energy price in €/MWh	8.32

Raw JSON example: {"msg":"balancing_reserve_bid", "msg_id":"938f2b97-314c-49e8-9860-f441df2284a1", "bid_id":"3640aa93-28a7-420e-aebf-f4a7fc3a08d2", "product":{"positive":true, "start":"2013-07-21T10:00:00.000Z", "end":"2013-07-21T10:15:00.000Z"}, "quantity":45.3, "divisible":true, "reserve_price":1.4, "energy_price":8.32}

The price is floating-point because JSON has no fixed-point numbers. No other currencies are supported in this version of the data model.

Details for field: product

Field	Description/comment	Example value
positive	direction - positive for generation/injection/upward reserve, negative for consumption/withdrawal/downward reserve; part of product description so that 'one product, one merit list'	true
start	start of period covered by this product	"2013-07-21T10:00:00.000Z"
end	end of period covered by this product	"2013-07-21T10:15:00.000Z"

6.1.2. Message type: response

Meaning: common response format for the messages that need acknowledgment or rejection (sent by receiver of the original message back to the sender).

Field	**Description/comment**	**Example value**
msg_id	ID of message we are responding to	"938f2b97-314c-49e8-9860-f441df2284a1"
response_code	error code; like OpenADR, we can use a subset of HTTP response codes, 200 being OK	200
response_subcode	optional message-specific additional code that further specifies error condition	200
response_desc	OK (code 200) or explanation of error (all other codes)	"OK"

Raw JSON example: {"msg":"response", "msg_id":"938f2b97-314c-49e8-9860-f441df2284a1", "response_code":200, "response_subcode":200, "response_desc":"OK"}

All messages that contain msg_id require a 'response' message as acknowledgement/rejection (except, of course, for the 'response' message itself).

Until the sender receives the response, it MUST NOT rely on the message being delivered or acted upon.

There is no cancel_bid message; bids can be changed by re-bidding the same bid ID, and cancelled by re-bidding with zero quantity.

There is no gate closure notification message - closure time is specified in the market rules and thus known in advance to all players.

6.1.3. Message type: bid_accepted

Meaning: bid accepted, BSP must confirm and then reserve the capacity (sent by TSO to BSP).

Field	**Description/comment**	**Example value**
msg_id		"fd127905-8005-44cc-8870-00ac8b6d27e3"
bid_id	the bid that was accepted	"3640aa93-28a7-420e-aebf-f4a7fc3a08d2"
quantity	the accepted quantity	33.1

Raw JSON example: {"msg":"bid_accepted", "msg_id":"fd127905-8005-44cc-8870-00ac8b6d27e3", "bid_id":"3640aa93-28a7-420e-aebf-f4a7fc3a08d2", "quantity":33.1}

6.1.4. Message type: market_cleared

Meaning: a notification that the market is cleared, thus all bids not yet accepted will never be accepted (sent by TSO to all BSPs that have bid).

Field	Description/comment	Example value
product	the product whose market is cleared	{"positive":true, "start":"2013-07-21T10:00:00.000Z", "end":"2013-07-21T10:15:00.000Z"}

Raw JSON example: {"msg":"market_cleared", "product":{"positive":true, "start":"2013-07-21T10:00:00.000Z", "end":"2013-07-21T10:15:00.000Z"}}

6.2 National balancing energy market

In the national balancing energy market, bids are submitted with the balancing_energy_bid message, which is almost the same as reserve_energy_bid. The acknowledgement message 'response' is exactly the same. There are no bid_accepted and market_cleared because bids are only inserted into the merit order list.

In addition, an 'activate' message signals both that the energy bid was accepted AND that it has to be activated in the specified period/quantity.

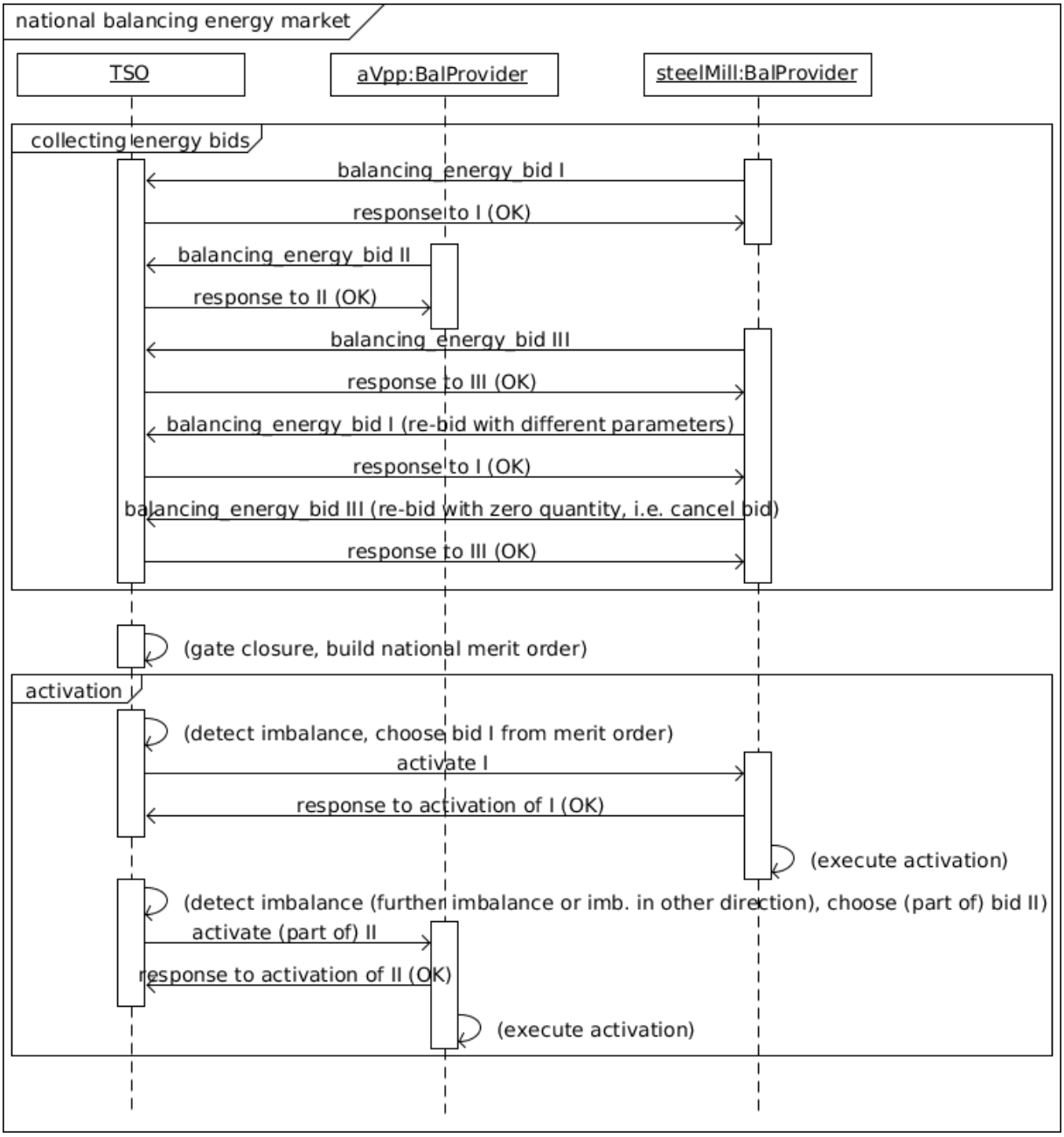

Figure 2: A typical message sequence in a national balancing energy market

6.2.1. Message type: balancing_energy_bid

Meaning: balancing energy bid (sent by BSP to its TSO).

Field	Description/comment	Example value
msg_id		"6c4de4fa-c950-4054-8a74-258088c29947"
bid_id	UUID of bid, used to refer to it later (e.g. to accept or cancel it)	"9b42269d-5b80-4029-8cce-79462eaf986e"
product	product object as shown above	{"positive":true, "start":"2013-07-21T10:00:00.000Z", "end":"2013-07-21T10:15:00.000Z"}

quantity	in MWh	15
divisible	false for all-or-nothing; default is true	false
energy_price	energy price in €/MWh	14.11

Raw JSON example: {"msg":"balancing_energy_bid", "msg_id":"6c4de4fa-c950-4054-8a74-258088c29947", "bid_id":"9b42269d-5b80-4029-8cce-79462eaf986e", "product":{"positive":true, "start":"2013-07-21T10:00:00.000Z", "end":"2013-07-21T10:15:00.000Z"}, "quantity":15, "divisible":false, "energy_price":14.11}

6.2.2. Message type: activate_bid

Meaning: activation, which BSP must confirm and then activate the capacity (sent by TSO to BSP).

Field	Description/comment	Example value
msg_id		"b10c501d-a10d-47d6-a0bf-068951eb6ae7"
bid_id	bid to activate	"9b42269d-5b80-4029-8cce-79462eaf986e"
quantity	quantity to activate	15
from	activation start; use null for 'ASAP'	"2013-07-24T11:10:20.000Z"
to	activation end; use null for 'till end of product'	"2013-07-24T11:14:55.000Z"

Raw JSON example: {"msg":"activate_bid", "msg_id":"b10c501d-a10d-47d6-a0bf-068951eb6ae7", "bid_id":"9b42269d-5b80-4029-8cce-79462eaf986e", "quantity":15, "from":"2013-07-24T11:10:20.000Z", "to":"2013-07-24T11:14:55.000Z"}

Once activated, a bid cannot be deactivated; it remains active until scheduled 'to' time or until end of product.

Should the TSO be able to modify an activation (e.g. increase quantity, postpone activation end time)? Since there are no deactivations we might need this.

6.3 International balancing energy market

On the international market, the TSOs forward (some or all of) the bids to the super-TSO with the same balancing_energy_bid message. For each forwarded bid, the super-TSO must also store the originating (forwarding) TSO.

The possibility of unshared bids does not affect the data model. The unshared bids are simply not forwarded.

Since the TSOs are not the market organizers any more (the latter role being trusted to the super-TSO), additional message types are required for the TSOs to request balancing energy, and for the super-TSO to grant and/or deny these requests.

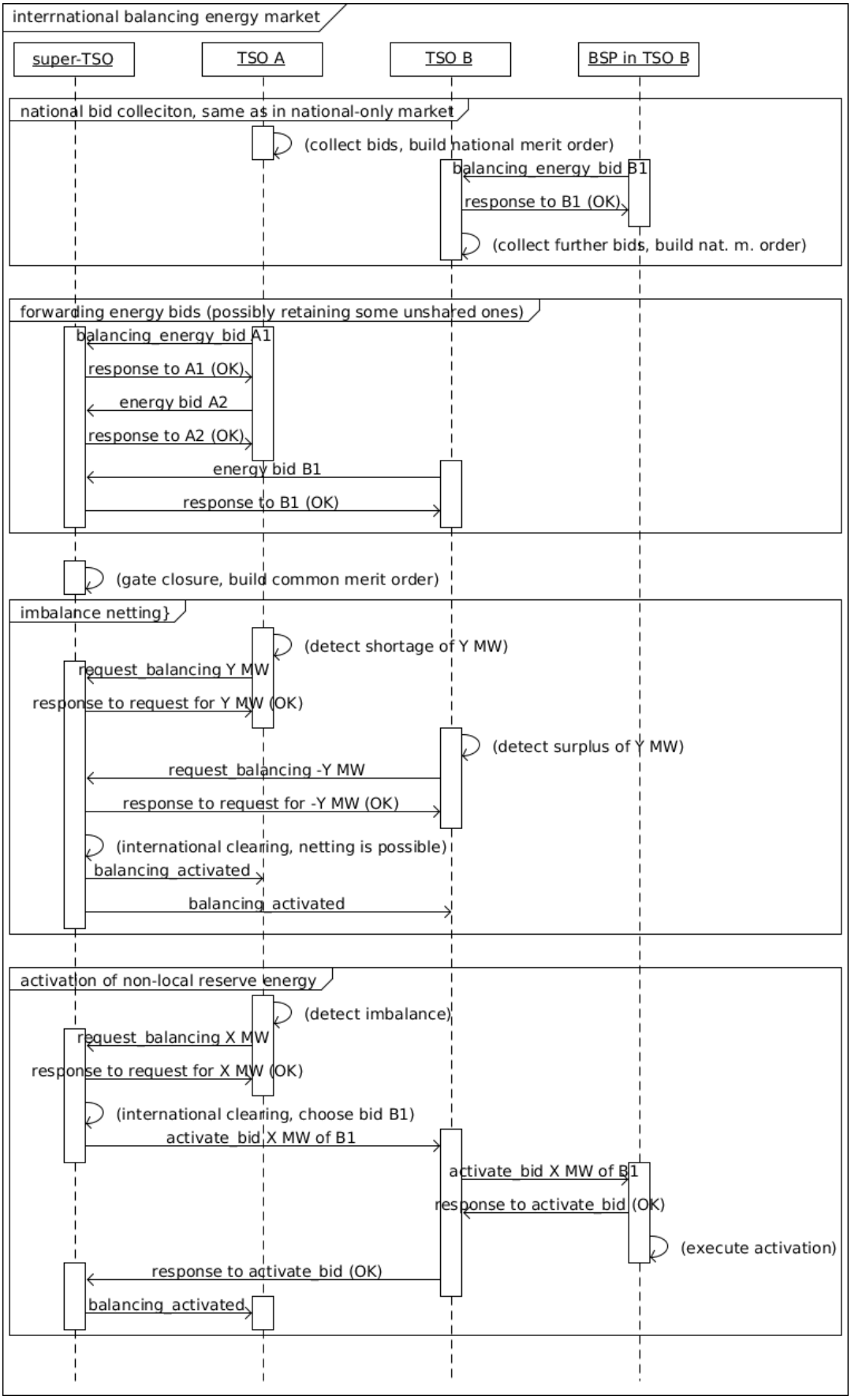

Figure 3: A typical message sequence in an international balancing energy market

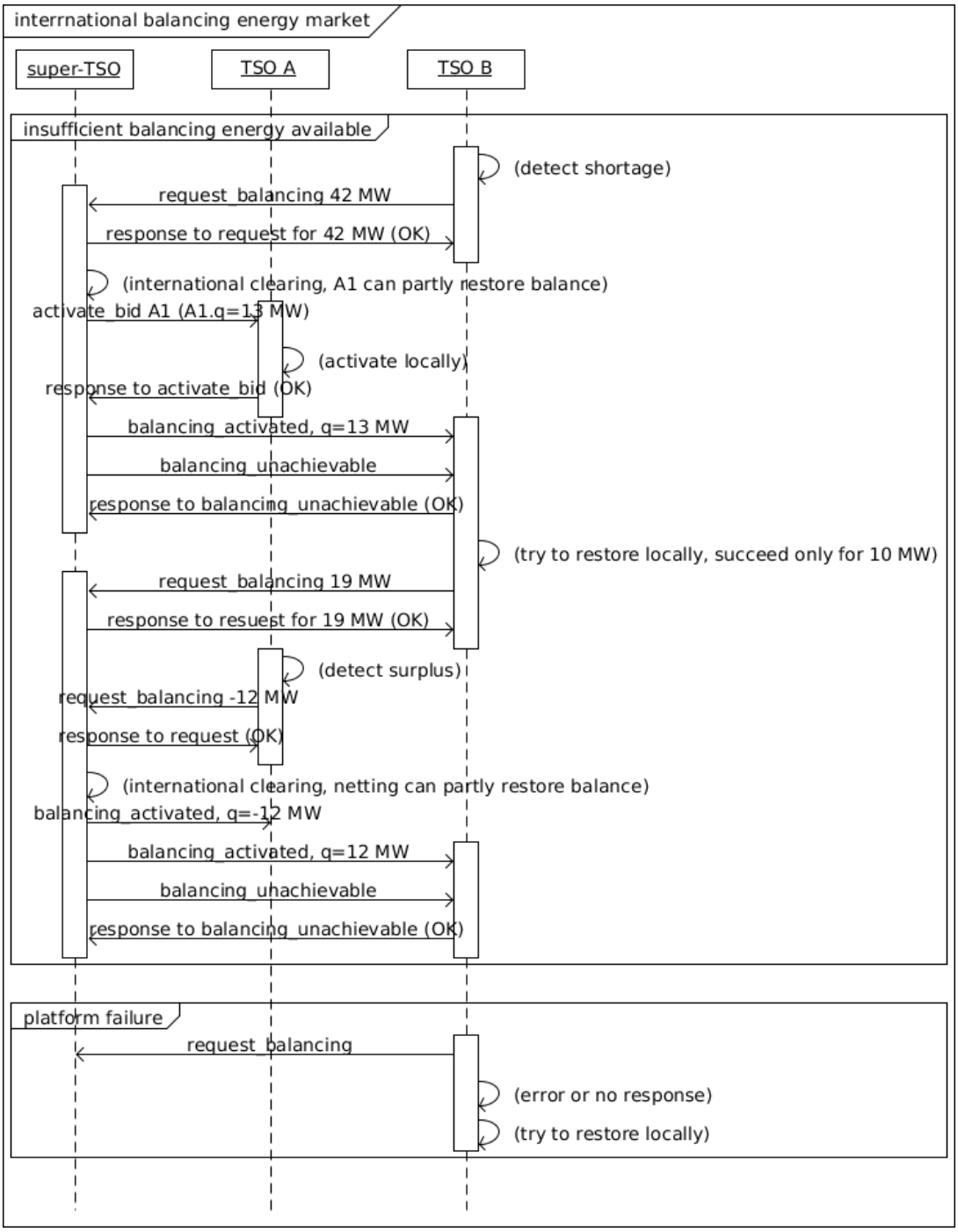

Figure 4: Two examples of unsuccessful balancing request in an international market

6.3.1. Message type: request_balancing

Meaning: notification about imbalance/request to activate balancing energy (sent by TSO to super-TSO).

Field	Description/comment	Example value
msg_id		"e8c99f81-97c2-43a5-90ad-98b7279a4a28"
quantity	quantity to activate; positive for 'shortage/consumption-higher-than-	-4.5

	generation/injection-required', negative for 'surplus/consumption-lower-than-generation/withdrawal-required'	
from	time when balancing is expected to be needed; use null for NOW	"2013-07-24T11:10:20.000Z"
to	time when not needed anymore; use null for UNTIL FURTHER NOTICE	"2013-07-24T11:14:55.000Z"

Raw JSON example: {"msg":"request_balancing", "msg_id":"e8c99f81-97c2-43a5-90ad-98b7279a4a28", "quantity":-4.5, "from":"2013-07-24T11:10:20.000Z", "to":"2013-07-24T11:14:55.000Z"}

When this message is acknowledged by the super-TSO, it is ADDED TO the imbalance (position) of the sending TSO (shortage or surplus) as known by the super-TSO. A request may be cancelled by sending another request_balancing message with the same quantity but opposite sign.

An activation is the same as in the national market, except that they are first sent by the super-TSO to the originating TSO (the TSO that the balancing energy bid came from), which forwards it to the BSP. The acknowledgements are, of course, sent and forwarded in the opposite direction.

A typical message sequence would be:

1. TSO A sends request_balancing.
2. The super-TSO acknowledges.
3. The super-TSO selects (one or more) bid(s) from the common merit order list and activates it/them by sending the appropriate messages.
4. The super-TSO informs the requesting TSO that the request was (partly or fully) granted with (one or more) balancing_activated message(s).

Note that activation of a bid originating from TSO A does NOT imply that the balance is restored in A; it may have been activated to restore balance anywhere else. If the TSO requesting balancing is the same as the one the selected bid comes from, it will receive both 'activate' and 'balancing_activated' messages.

6.3.2. Message type: balancing_activated

Meaning: informs the destination TSO that its balance is/will be (partly) restored by energy from originating TSO (sent by super-TSO to destination TSO).

Field	Description/comment	Example value
quantity	quantity that was activated	-4.5
from	start of activation	"2013-07-24T11:10:20.000Z"
to	end of activation	"2013-07-24T11:14:55.000Z"
originating_tso	originating TSO	"TSO-A"
bid_id	bid ID; null if the imbalance of two TSOs was netted	"9b42269d-5b80-4029-8cce-79462eaf986e"

Raw JSON example: {"msg":"balancing_activated", "quantity":-4.5, "from":"2013-07-24T11:10:20.000Z", "to":"2013-07-24T11:14:55.000Z", "originating_tso":"TSO-A", "bid_id":"9b42269d-5b80-4029-8cce-79462eaf986e"}

The balancing_activated message implies that the TSO's imbalance/position for this period changes by the specified quantity, i.e. the quantity is added to the TSO's position.

The balancing_activated message has no acknowledgement. The activation will happen anyway, even if e.g. destination TSO's computer is down, thus an acknowledgement would not serve any purpose.

The destination TSO does not know the contents of the bid, but it can save the ID in case of later disputes.

6.3.3. Message type: balancing_unachievable

Meaning: informs a TSO that the referenced balancing request could not be (fully, or at all) fulfilled (sent by super-TSO).

Field	Description/comment	Example value
msg_id		"79b25a8f-67b1-4303-a980-fc43f7cc6f63"
request_id	the msg_id of request_balancing message that this refers to	"e8c99f81-97c2-43a5-90ad-98b7279a4a28"

Raw JSON example: {"msg":"balancing_unachievable", "msg_id":"79b25a8f-67b1-4303-a980-fc43f7cc6f63", "request_id":"e8c99f81-97c2-43a5-90ad-98b7279a4a28"}

This message may or may not be preceded by balancing_activated messages that partly fulfil the request, but it implies that no later balancing_activated messages will be sent that would balance that request.

After receiving an acknowledgement of this, the super-TSO will assume that the TSO will to restore the remaining imbalance by other means (e.g. with unshared bids). The super-TSO shall not try to restore the remaining imbalance, not even if it could be netted with the imbalance in the other direction requested later by another TSO.

If the TSO that remains imbalanced wants the super-TSO to re-try, it must send a new request_balancing message.

7. Conclusions

The requirements analysis has shown that the communication and data model can be divided into two levels:

- HEH-level, where the metering data is collected, individual demand-response commands are sent etc.,
- market level, where the BSPs send balancing energy bids and the TSOs activate selected ones as needed.

The analysis of existing data and messaging standards has shown that on the HEH-level, the eBADGE requirements and the OpenADR 2.0 standard are not aligned closely enough to warrant implementation of this standard. All other standards from the Smart Grid domain are even less appropriate for reasons of inextensibility, lack of fitness for our objectives, or dependency on inbound connectivity into the home energy hubs.

On the market level, no suitable standards exist either. The existing markets, such as the Slovenian intra-day market, use the data models and exchange formats defined by the respective vendors of the market software.

After the definition of a new data model, JSON encoding was chosen as a suitable compromise between compactness, human-readability, interoperability and extensibility.

The detailed descriptions and examples of all message types for the first version of the eBADGE data model are given in this document, accompanied by sequence diagrams for the more complex communication scenarios. The data model covers the specific requirements of the eBADGE pilot but the implementation strives to be applicable beyond this project and allows third-party extensions.

7.1 Planned Further Developments in Later Versions

This data model will be updated and refined in deliverables D3.1.2 and D3.1.3, due 12 and 24 months after this deliverable, respectively. Firstly, certain features that have already been foreseen but are not essential for the development of first prototypes will be added, such as the ability to report non-electricity metering data (e.g. water usage, heat generation) or communication with additional types of entities (e.g. DSOs, energy providers).

Secondly, we will keep following the development of new and existing relevant standards. If a new standard emerges that covers our use cases very well or, more likely, an existing standard is updated in such manner, we will re-consider adopting it.

Finally, as the project progresses and the pilot is implemented, the data model will almost certainly need to be changed due to requirements that we were not able to capture in the first year, both from the perspective of content (e.g., a previously unforeseen market mechanism may require additional message types) as well as from the technical perspective (e.g., potential decision for another type of message bus may require adding explicit sender and/or receiver fields to each message).

References

[BSP web page]: BSP Regionalna Energetska Borza d. o. o., http://www.bsp-southpool.com/trgovanje-znotraj-dneva.html

[DR comparison, 2011]: Comparison of Demand Response Communication Protocols, Scott Coe, UISOL, http://canmetenergy.nrcan.gc.ca/renewables/smart-grid/publications/3045

[eBADGE D3.2.1, 2013]: The eBADGE Message Bus – First Intermediate version, eBADGE project deliverable D3.2.1, September 2013.

[IEC 6*, 2008]: Comparison of IEC 60870-5-101/-103/-104, DNP3, and IEC 60870-6-ASE.2 with IEC 61850, Karlheinz Schwarz, 2008, http://www.nettedautomation.com/news/n_51.html

[IEC 61850, 2013]: News on IEC 61850 and related Standards. Compilation of the IEC 61850 blog until August 4th, 2013, http://www.nettedautomation.com/download/Newsletter/IEC61850-Blog_until_2013-08-05_o.pdf

[IEC SmartGrid]: Core IEC Smart Grid Standards, http://www.iec.ch/smartgrid/standards/

[iGorenje v15, 2012]: iGorenje specification (classified).

[Martínez et al, 2013]: José-Fernán Martínez, Jesús Rodríguez-Molina, Pedro Castillejo, and Rubén de Diego, Middleware Architectures for the Smart Grid: Survey and Challenges in the Foreseeable Future. Energies 2013, 6, 3593-3621. http://www.mdpi.com/1996-1073/6/7/3593

[NYSE, 2012]: NYSE UTPDirect (CCG Binary) API Specification – NYSE and NYSE MKT Cash Equities Markets, 2012, http://usequities.nyx.com/connecting/ccg-binary-api

[OPC UA Part I, 2009]: OPC Unified Architecture Specification, Part I: Overview and Concepts, Release 1.01. OPC Foundation, February 5, 2009.

[OPC UA web page]: OPC Unified Architecture, http://www.opcfoundation.org/Default.aspx/01_about/UA.asp?MID=AboutOPC

[OpenADR, 2009]: Open Automated Demand Response Communication Specification (Version 1.0), Lawrence Berkeley National Laboratory, http://openadr.lbl.gov/pdf/cec-500-2009-063.pdf

[OpenADR 2.0, 2013]: OpenADR 2.0 Profile Specification – B Profile, OpenADR Alliance, 2013, http://www.openadr.org/specification

[OpenADR Transport Layer]: OpenADR Version 2 – Potential Transport Mechanisms, http://cio.nist.gov/esd/emaildir/lists/h2g_interop/pdf00007.pdf

[OSGP, 2012]: Open Smart Grid Protocol (OSGP), http://www.etsi.org/deliver/etsi_gs/OSG/001_099/001/01.01.01_60/gs_OSG001v010101p.pdf

[Sučić, 2012]: Stjepan Sučić, Ante Martinić, Ana Kekelj, Utilizing standards-based semantic services for modeling novel Smart Grid supervision and remote control frameworks, IEEE International Conference on Industrial Technology, Athens, 2012.