The eBADGE Data Standard Report – First Version

Deliverable: **D3.1.1**

Of project: **eBADGE**

Grant Agreement no. **318050**

Project Full Title:

Development of Novel ICT tools for integrated Balancing Market Enabling Aggregated Demand Response and Distributed Generation Capacity

Project Duration: **3 years**

Project start date: **October 1st, 2012**

SP1 – Cooperation

Collaborative project

Small or medium-scale focused research project

Due date of deliverable:	Month 12 (30. 9. 2013)
Workpackage:	WP3
WP Coordinator:	XLAB
Authors:	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)

Dissemination level		
PU	Public	**X**
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)	

Table of Contents

Executive Summary

This document defines the eBADGE data standard used for the communication between all the stakeholders in the pilot project. We first list some general requirements and then evaluate the appropriateness of the existing standards, such as OpenADR, OPC, IEC 61850, and others. Of those, only OpenADR 2.0 seems to provide benefits to the project; however, the fit is far from perfect and implementing fully OpenADR 2.0-compatible software is beyond the scope of the project. Thus, we have decided to implement a "green field" solution.

We decided against binary encodings for reasons of readability, ease of development, extensibility, and maintainability. Instead, JSON avoids these shortcomings and provides a much more compact encoding than XML as well as better support for lightweight and scripting programming languages.

The document explains the general approach to encoding dates and times, selecting field names, implementing synchronous operation where necessary, and adding third-party extensions. This is followed by the formal definition of the standard, i.e. the list of all the message types required to cover the currently foreseen eBADGE use cases, with complete details and JSON examples. The standard definition 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 standard.

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 data standard developed with all the relevant attributes for information exchange between the entities.

This deliverable presents the first version of the proposed open data standard that will be used for end-to-end communication between the various entities in the eBADGE pilot. The standard defines the types of messages, their contents, meaning, and the typical sequences of messages in various scenarios.

1.1 Approach

This first version of the eBADGE data standard 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 they can be used for eBADGE, 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.

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, extensibility etc 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 standard, such as how the message contents are encoded and how the message acknowledgements are formatted.

1.2 Outline of This Report

The next section briefly summarizes the requirements and the corresponding division of the standard into two communication levels. Section 3 analyzes existing standards related to eBADGE communication. Section 4 describes the general approach applicable 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 balancing providers (e.g. traditional generators) send bids to their TSO, who send activation commands back,
3) TSOs forward bids to a common/central energy market operator and request balancing energy allocation from it.

No messages pass the level border (for example, the home 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.

The full list of required message types and contents that resulted from the requirements collection process was used directly to define this data standard. As the resulting standard is provided in this document, the full requirements would just duplicate the information and are thus not listed here.

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 telco 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 standard, we shall restrict ourselves to the basic model. Thus, this level of the data standard 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.

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 etc,
- 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 those smart devices that are able to predict their usage,
- 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.

Additionally, the HEH must be able to receive tariff (pricing) updates and activation commands (by how much to reduce or increase the load or generation, the time period, and possibly the device that should be used to achieve this). The data standard has to allow the HEH to reject an activation and to suggest its modification, although, depending on the contract between e.g. the home user and the VPP, this may only be allowed in limited cases.

2.2 Market Level Requirements

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

This part of the standard 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.

3 Analysis of Existing Standards

Currently, the data standards at the HEH level and 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 will analyze the applicability of the most important existing standards to eBADGE. Unfortunately, as it turns out, no existing standard fits the requirements well enough, 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.

3.1 Smart Grid Data Standards

3.1.1 iGorenje

iGorenje [iGorenje v15, 2012] is a simple HTTP/SOAP API for demand response. As a HTTP-based protocol, it requires either inbound connectivity into the clients (HEHs) 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. Furthermore, the overhead of the SOAP protocol is significant.

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 the semantics would be re-implemented exactly but using another communication channel rather than HTTP, one cannot talk about the same standard any more.

3.1.2 OpenADR, OpenADR 2.0

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

OpenADR 2.0 is semantically a better, but not a perfect fit. In our case, the HEH would act as an OpenADR VEN (virtual end node) and a VPP could be a VTN (virtual top node). OpenADR 2.0 can use HTTP or XMPP communication; the latter solves the above-mentioned problems of HTTP [OpenADR 2.0, 2013].

We have analyzed it in detail and defined OpenADR 2.0 equivalents of some of the messages required in eBADGE: an activation command (eiEventSignal LOAD_DISPATCH in OpenADR terminology), rejection of the activation (eventResponse optOut) and a pricing update (eiEventSignal ELECTRICITY_PRICE). In addition to OpenADR's quite verbose XML encoding, the imperfect semantic fit results in large parts of messages that are required according to the standard but of no use for eBADGE. This results in the three OpenADR 2.0 test messages being around twelve times as big as a lighter-weight JSON encoding of the same information (four times when compressed).

Although OpenADR 2.0 could be partly implemented, a full implementation is much beyond the scope of this project. Furthermore, certain requirements are not supported, thus we would have to define extensions (which are allowed in the standard). An eBADGE HEH would thus not be compatible with third-party OpenADR 2.0 equipment, which would defeat the purpose.

Nevertheless, certain aspects of OpenADR are mimicked in the eBADGE data standard. For example, each activation order contains a modification_count 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.

3.1.4 OPC, OPC UA

The older OPC (originally standing for OLE for Process Control, re-defined as Open Platform Communications) and newer OPC UA (OPC Unified Architecture) are communication standards used in industrial automatics. They are used in many fields, including demand-response. However, OPC only defines ways to access attributes and issue commands. The semantics of these attributes and commands

are application/equipment-specific or can be defined by an upper-level standard, so it is not really applicable as an "eBADGE data standard".

Additionally, the original OPC requires Microsoft Windows. OPC UA is cross-platform and defines two communication protocols: HTTP/SOAP as well as a binary protocol.

Although neither OPC nor OPC UA will be part of the eBADGE data standard, existing OPC/OPC UA devices can be supported through separate protocol gateway services.

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 (International Electrotechnical Commission) standards define how to exchange numbered lists of simple data points while optimizing the use of bandwidth and hardware. While 60870-5-101 works over serial lines, 60870-5-104 works over TCP [IEC 6*, 2008]. None of them really covers our requirements, e.g. to send bids for balancing energy.

3.2.2 IEC 61850

IEC 61850 is an electrical substation automation standard [IEC61850, 2013]. It can be seen as a modern version of 60870-5-104, being based on XML to provide an extensible, orthogonal message and data format. Similarly to e.g. OPC, it defines a way to set/get attribute values rather than prescribe names, types, or meaning of the attributes.

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, 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

Following the decision to propose a new standard, optimal choices could be made on all technical properties of the standard.

4.1 Message Payload Encoding

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

4.1.1 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 standard 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.

Should the standards define the exact binary representation, a lot of work is needed to ensure consistency and to develop the encoding and decoding software. On the other hand, if it relies on a standard serialization method such as Java object serialization or Google Protocol Buffers, it is difficult to ensure interoperability even with the current versions of various programming languages, while maintaining it for the foreseen lifetime of the standard is even harder.

Finally, binary standards 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 standard and lessens the compactness advantage of binary formats.

4.1.2 Text-Based Encoding

The two most widely used text-based encodings are XML and JSON. XML is traditionally used in business-level software because the validity of any XML document can be verified against a suitable XML schema, which describes the allowed format of the document. Also, the use of namespaces solves 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 more popular in other fields. Although both XML and JSON can be made quite compact by choosing short field/tag names, JSON is the clear winner as far as message size is concerned. This advantage, combined with ease of use in most programming languages, makes JSON the encoding of choice for eBADGE data standard.

4.1.3 Field names

The choice of field names in JSON is always a compromise between readability and compactness. In this draft standard we stressed the former over the latter. If 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.

4.1.4 Encoding of Standard Data Types

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

Although all JSON numbers are floating-point, the eBADGE standard may define certain fields to be integer, or positive, or contain at most two decimals etc. The JSON standard does not support IEEE754 special values Inf, -Inf, and NaN. Although some implementations add support for these values, the eBADGE standard 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 €.

JSON does not define date or time formats, thus all times are encoded as strings according to ISO 8601. Although this standard 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.

4.1.5 Extensibility

To ensure distributed extensibility, any party may introduce additional message types or additional fields in existing messages. This version of the standard does not define how the receiver should react to unknown extensions, so the sender must assume they may be ignored 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 standard 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 cirrently multiple efforts related to JSON extensibility: JSON-LD[1], JSON Schema[2], and the JSON namespace proposal[3]. By the time the eBADGE data standard is finalized, one of these or another solution may become standardized, allowing us to retire the above convention.

4.2 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 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.

4.3 Asynchronous and Synchronous Communication

The data standard is agnostic to whether a synchronous (such as HTTP) or asynchronous (such as AMQP) communication protocol is used. Whenever an asynchronous protocol is used and a message requires a response, a special message type is needed.

4.3.1 Message type: response

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

The sender of any message that requires response must assume that the message may have been lost until it receives this response.

Field	Description/comment	Example value
msg_id	A unique identifier of the message that this is the response to. Each message that requires a response must contain such a	"d1c7ff9d-23f9-4257-9b28-41ab54931aa4"

[1] http://json-ld.org/

[2] http://json-schema.org/

[3] http://www.watersprings.org/pub/id/draft-saintandre-json-namespaces-00.txt

	unique ID.	
response_code	Error code; like OpenADR, we can use a subset of HTTP response codes, 200 signalling "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":"d1c7ff9d-23f9-4257-9b28-41ab54931aa4", "response_code":200, "response_subcode":200, "response_desc":"OK"}

4.4 Advice to Implementors

Please note that this is the first, draft version of the data standard that has not yet 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 as "The eBADGE Message Bus, First Intermediate Version", eBADGE project public deliverable D3.2.1.

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.

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; ANY MORE?	"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 measurment 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	"ECAR01"

	total	
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 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	0
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}

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).

Field	Description/comment	Example value

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	not sure what we need here...	"OK"
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 capabilites 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		"IMV e-

		katrca"
version		"0.9 beta"
ext_si_imv_ecar_battery_capacity	extension example, in this case battery capacity in kWh	14.2
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":"IMV e-katrca", "version":"0.9 beta", "ext_si_imv_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 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,	"938f2b97-314c-49e8-9860-f441df2284a1"

	others do not	
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 standard.

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).

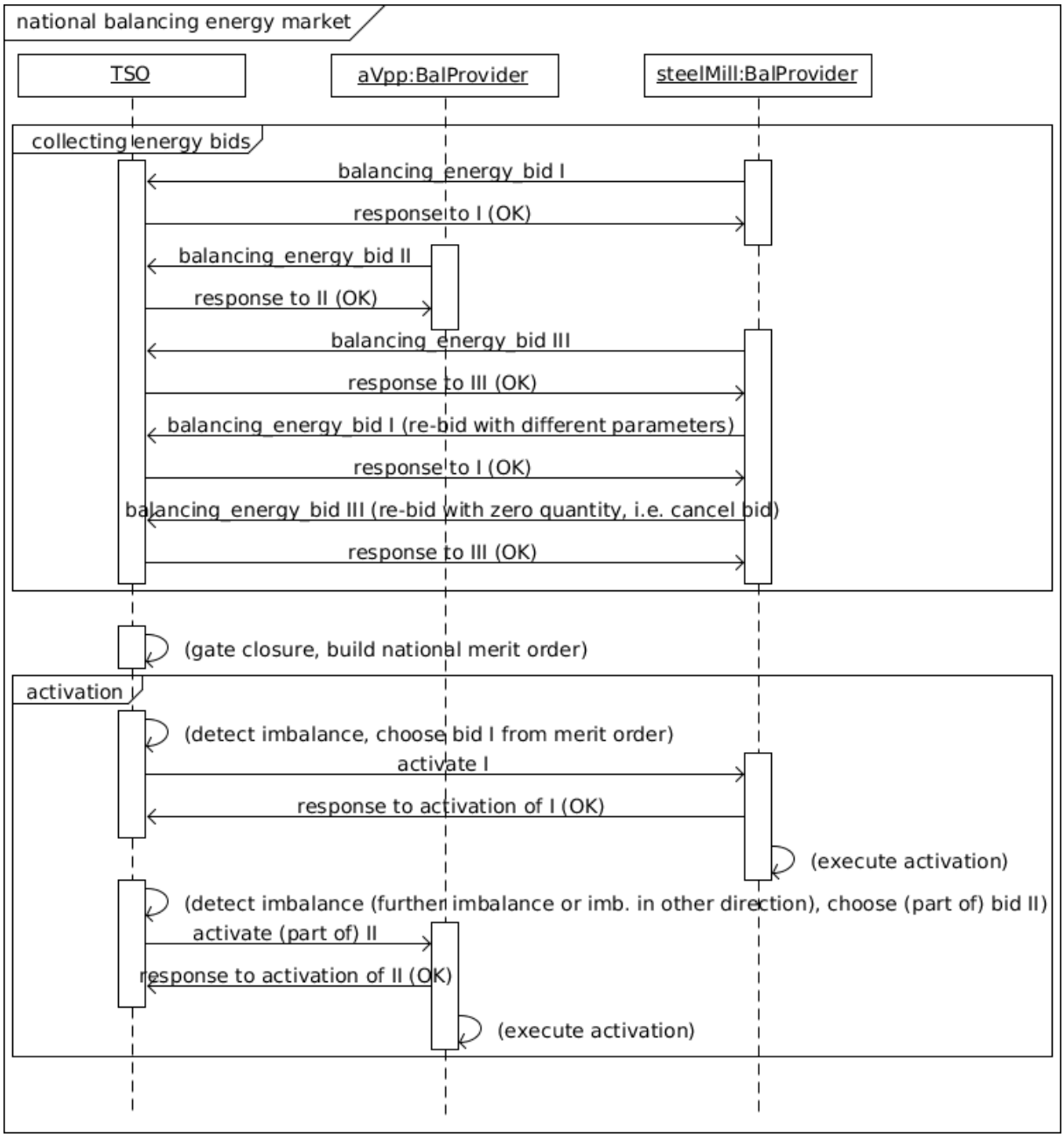
Field	Description/comment	Example value
product	the product whose market is cleared	{"positive":true, "start":"2013-07-21T10:00:00.000Z", "end":"2013-07-21T10:15:00.000Z"}

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.



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.

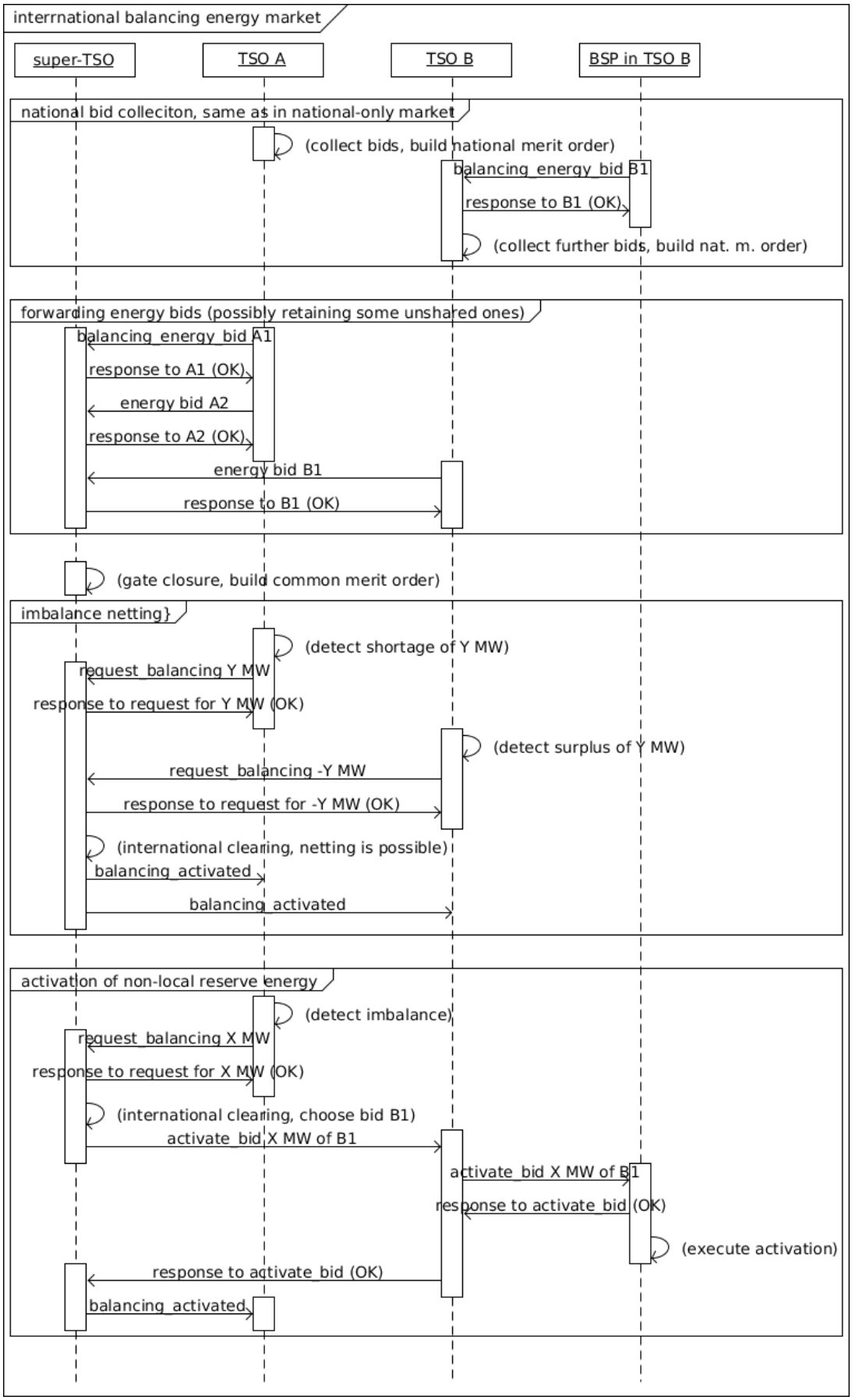
Should the TSO be able to modify an activation (e.g. increase quantity, postpone activation end time)? Since there are no deactivations we might need this.

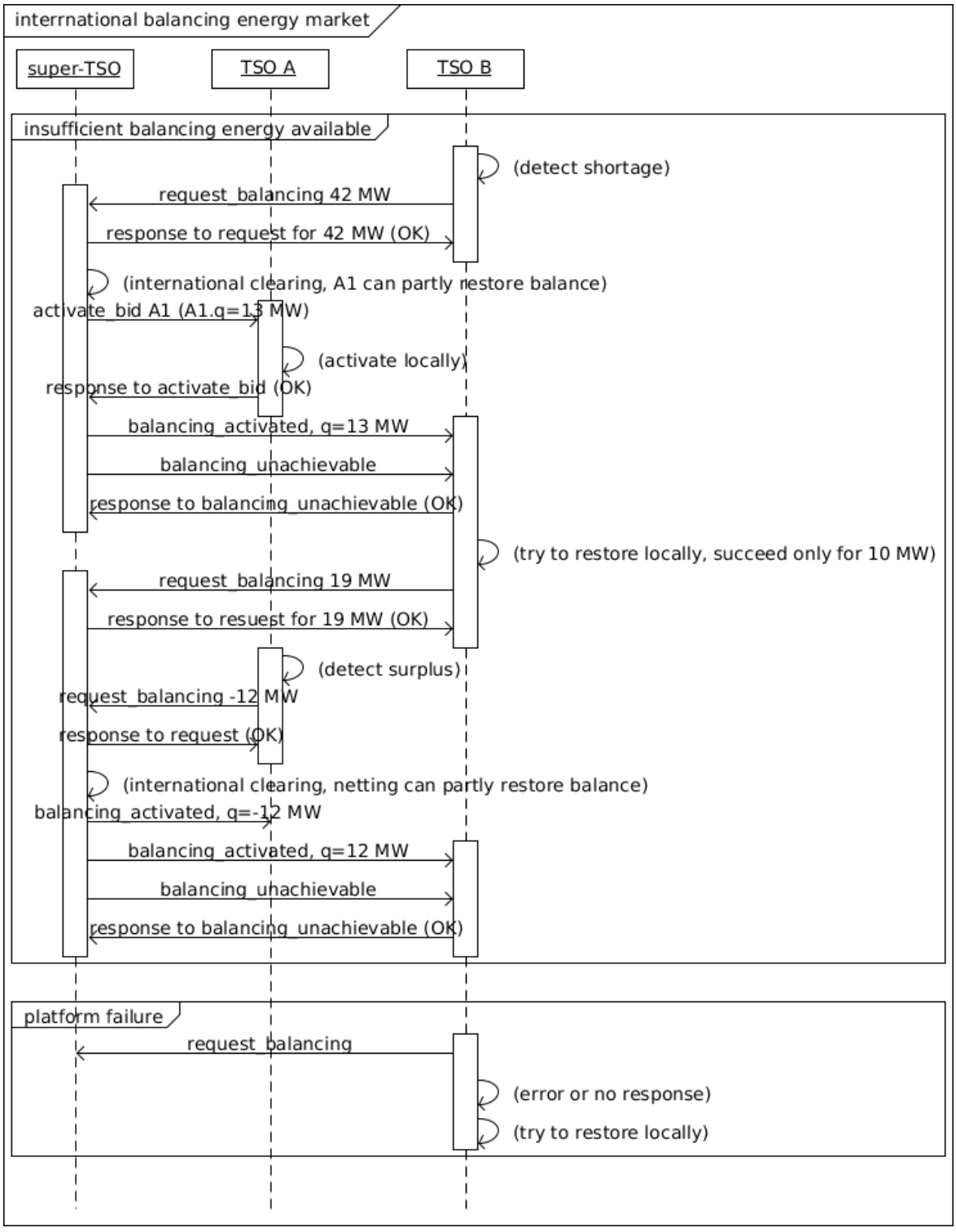
6.3 International balancing energy market

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 possiblity of unshared bids does not affect the messaging standard. 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.


interrnational balancing energy market
super-TSO
TSO A
TSO B
BSP in TSO B
national bid colleciton, same as in national-only market
(collect bids, build national merit order)
balancing_energy_bid B1
response to B1 (OK)
(collect further bids, build nat. m. order)
forwarding energy bids (possibly retaining some unshared ones)
balancing_energy_bid A1
response to A1 (OK)
energy bid A2
response to A2 (OK)
energy bid B1
response to B1 (OK)
(gate closure, build common merit order)
imbalance netting}
(detect shortage of Y MW)
request_balancing Y MW
response to request for Y MW (OK)
(detect surplus of Y MW)
request_balancing -Y MW
response to request for -Y MW (OK)
(international clearing, netting is possible)
balancing_activated
balancing_activated
activation of non-local reserve energy
(detect imbalance)
request_balancing X MW
response to request for X MW (OK)
(international clearing, choose bid B1)
activate_bid X MW of B1
activate_bid X MW of B1
response to activate_bid (OK)
(execute activation)
response to activate_bid (OK)
balancing_activated



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-generation/injection-required', negative for 'surplus/consumption-lower-than-generation/withdrawal-required'	-4.5
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 positione.

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 first shown that the communication and data model can be divided into the HEH-level, where the metering data is collected, individual demand-response commands are sent etc, and the market level, where the BSPs send balancing energy bids and the TSOs activate selected ones as needed.

The analysis of existing data and messaging standard 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 are even less appropriate for reasons of inextensibility, lack of fitness for our purpose, dependency on inbound connectivity into the home energy hubs, or other reasons.

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 deciding for the definition of a new data standard, JSON encoding was chosen as the optimal compromise between compactness, readability, and extensibility. The detailed descriptions and examples of all message types for this, first version of the eBADGE data standard are given in this document, accompanied by sequence diagrams for the more complex communication scenarios. The data standard was designed to be generally applicable rather than just for this project. While we do not claim to have covered all use cases, the standard allows third-party extensions. Furthermore, we plan to cover other often used scenarios in later versions of the core standard.

7.1 Planned Updates in Later Versions

This data standard will be updated and refined in deliverables D3.1.2 and D3.1.3, due 12 and 24 months later than 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, as the project progresses and the pilot is implemented, the standard 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).

Finally, the message sequences in all possible scenarios have to be analyzed and the protocol consistency formally checked.

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

iGorenje v15, 2012: iGorenje specification (classified).

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

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

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