Life Energy Motion

E-Mobility solutions
DCBM 400/600 Series - DC Energy Meter
Communication protocols manual

18May2024/Version 3

TABLE OF CONTENTS

1. SAFETY RULES

1.1. SAFETY WARNING

In order to guarantee safe operation of the product and to be able to make proper use of all features and functions, please read these instructions thoroughly!

Safe operation can only be guaranteed if the product is used for the purpose it has been designed for and within the limits of the technical specifications. Ensure you get up-to-date technical information that can be found in the latest associated datasheet under www.lem.com.

Terminal protection cover delivered with the product must be installed to obtain proper electrical protection. The data link cable used between the product's elements shall be the one delivered by LEM.

Please note

Electrical equipment should be installed, operated, serviced and maintained only by qualified personnel.

No responsibility is assumed by LEM International SA for any consequences arising out of the use of this material. A qualified person is one who has skills and knowledge related to the construction, installation, and operation of electrical equipment and has received safety training to recognize and avoid the hazards involved.

The meter must be installed inside an enclosure IP51 (indoor) or IP54 (outdoor) according to

EN 50470:2007.

DANGER! Electrical hazard - Fire hazard

⚠

When installing or changing the product, the conductor to which the product is connected must be de-energized. Ignoring the warnings can lead to serious injury and/or cause damage!

Notice! Damage or hazards
The appropriate torque is defined by LEM (see Installation manual section "6. Connection")
The appropriate crimping of the connection elements is defined by the nationalities in force.

1.2. IMPORTANT NOTICES

- Time source to set product’s time must be provided by the customer. Product must be time synchronized to operate.
- Product’s Ethernet interface mustn’t be exposed to a public network; network must be private and secured.
- To ensure proper operation, product’s logbook completion must be checked periodically; the maximum number of entries is up to 39 999; product’s operation stops if logbook is full.
- The product is designed with IP20, and is intended to be mounted in an enclosure with a suitable IP rating for the final application.

Symbols and conventions

The following symbols point out critical information. They can be found either in this document or on the product.

Instruction manual	Heating	Warning
Electrical hazard	Double insulation	

The following symbols aim at improving reader's experience by highlighting sections.

Note

Tip

2. Document information

2.1. Document overview

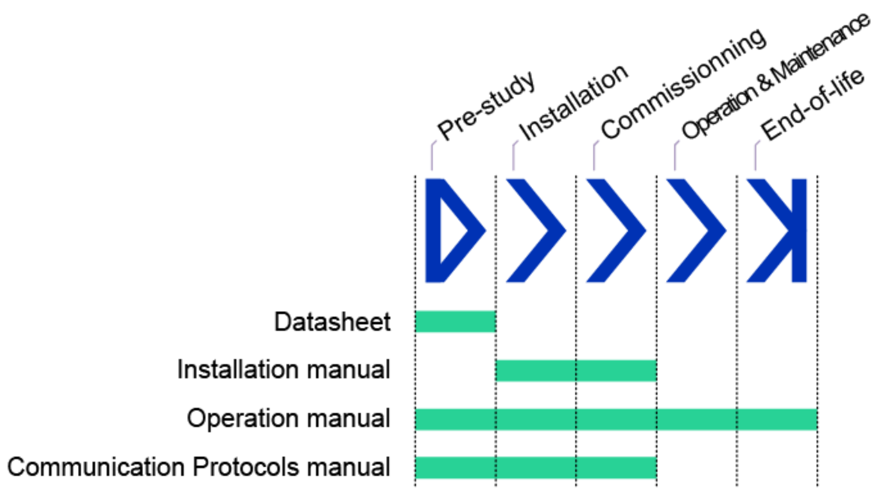
This document relates to the DCBM 400/600 product family. Those products are direct connected energy meters for DC applications.

This manual provides detailed information for interfacing with the products. This includes:

- Checking and setting the configuration
- Monitoring status and measurements
- Managing new transactions
- Retrieving the stored data: transaction & event logbooks

The document is structured following the APIs of the products, it explains available fields together with examples.

This document is intended to be used in combination with the DCBM 400/600 Operation manual which describes the relevant concepts. More generally, below illustration depicts the set of documents for DCBM 400/600 product family with associated steps in product lifetime.



2.2. Document issue

Release scope: Public

Targeted products: DCBM 400/DCBM 600 (version NxD or NxM)

Applies to software versions:

APPLICATION_VERSION = 2.3.0.1 – Authentification tag: 0xBC9595BD5D619F909A4B6F93
LEGAL_VERSION = 2.3.0.1 – Authentification tag: 0x7BE605E0439539EECE15E856

Products	Meter Unit SW version	Sensor Unit SW Version	Sensor Unit CRC
DCBM 400	2.3.0.1	0.0.8.0	0x540F
DCBM 600		0.1.3.0	0x3CBB

Release scope: Public

Targeted products: DCBM 400/DCBM 600 (version N00)

Applies to software versions:

APPLICATION_VERSION = 1.2.0.1 – Authentification tag: 0xA250DA16E47AFCBF65FBE860
LEGAL_VERSION = 1.2.0.1 – Authentification tag: 0x2912C50DE009088E4F39BF19

Products	Meter Unit SW version	Sensor Unit SW Version	Sensor Unit CRC
DCBM 400	1.2.0.1	0.0.8.0	0x540F
DCBM 600		0.1.3.0	0x3CBB

2.3. Company details

LEM International SA
Rte du Nant-d'Avril 152
1217 Meyrin
Switzerland

2.4. Version history

Version	Date	Changes
0	2 July 21	First issue
1	6 February 2023	Modification page 10 of software versions
2	24 July 2023	Second release for API V2 Add new transaction fields (TT, UV, UD). Add note for max size of LEM/OCMF format. Add note for max Energy registering. Update Pagination Counter description.
3	16 May 2024	Update the threshold of Temperature detected Add the version N00 (version 1.2.0.1) Add note for TCP socket connection flow Update the table of contents

2.5. Related documents

Name	Revision
Datasheet	
Operation manual	
Installation manual	
EN 50470-1	2006
EN 50470-3	2006
IEC 62477-1	
EN 62052-31	2015
IEC 62052-11	2003
IEC 61000-6-2	2016
IEC 61000-6-3	2016
RoHS	
OCPP 1.6	1.6, 2.6
WELMEC 7.2 Software Guide	2015
REA-6A	
PTB 50.7	
VDE-AR-E 2418-3-100	2019-07-16

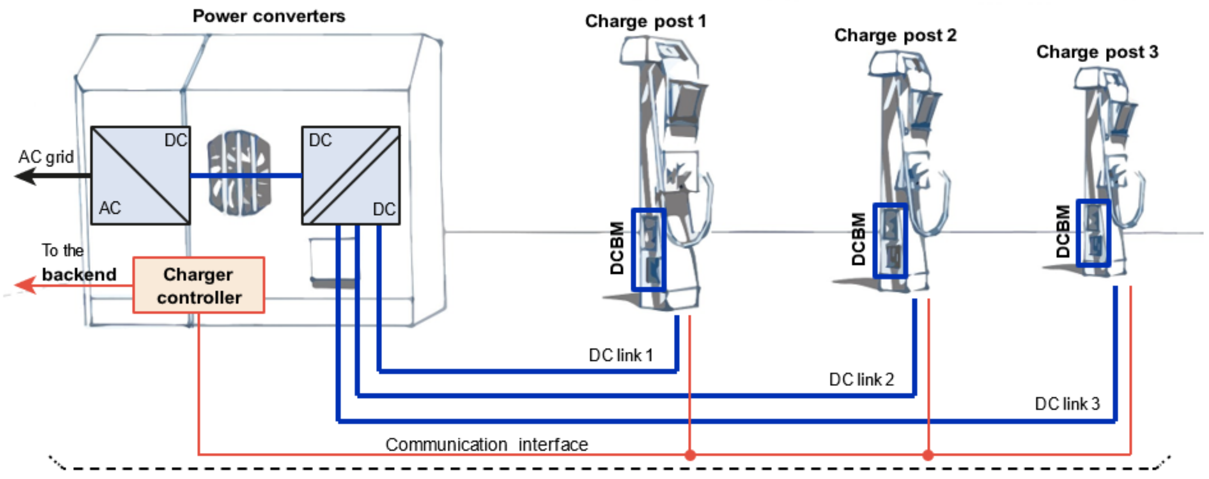
2.6. ABBREVIATIONS

Abbreviation	Description
ABS	Absolute value
ADC	Analog to Digital Converter
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CR / LF	Carriage Return / Line Feed
CRC	Cyclic Redundancy Check
DC	Direct Current
DCBM	DC Billing Meter
DER	Distinguished Encoding Rules
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DST	Daylight Saving Time
ECDSA	Elliptic curve digital signature algorithm
EV	Electric Vehicle
EVSE	Electric Vehicle Supply Equipment
FF	Fatal Failure
GMT	Greenwich Mean Time
GPIO	General Purpose Input / Output
HTTP[S]	Hypertext Transfer Protocol [Secured]
ID	Identifier
IP	Internet Protocol
JSON	JavaScript Object Notation
LEN	Length
LR / LNR	Legally relevant / Legally non-relevant
MSB / LSB	Most Significant Bit / Least Significant Bit
MU	Meter Unit
N/A	Not Applicable
NTP	Network Time Protocol
OBIS	Object Identification System
OCMF	Open Charge Metering Format
OCPP	Open Charge Point Protocol
PLMN	Public Land Mobile Network
REST	Representational state transfer
RFID	Radio Frequency Identification
RNG	Random Number Generator

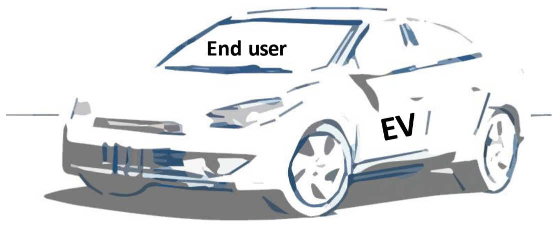
SC	Screen Choice
SHA	Secure Hash Algorithm
SMS	Short Message Service
SNTP	Simple Network Time Protocol
SU	Sensor Unit
TT	Tarification Text
UV	Customer data
UD	Customer data
UDP	User Datagram Protocol
UID / UUID	[Universally] Unique Identifier
URL / URI	Uniform Resource Locator / Identifier
UTC	Coordinated Universal Time
UTF	Universal Character Set Transformation Format
XTAL	Cristal (oscillator)

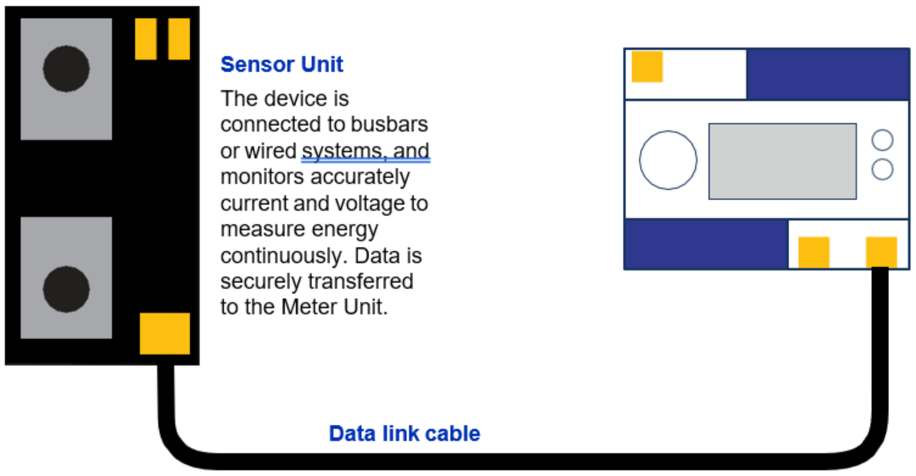
2.7. Glossary

Key words used in this document, designating DCBM construction or DCBM's ecosystem equipment and protagonists, are illustrated in below visuals.



EVSE/Charging station, operated by the **Charge point operator**





Sensor Unit

The device is connected to busbars or wired systems, and monitors accurately current and voltage to measure energy continuously. Data is securely transferred to the Meter Unit.

Meter Unit

The device stores, displays, communicates and authenticates measurement data. The Meter Unit delivers power supply to the Sensor Unit.

The Meter Unit offers communication with the charging station through Ethernet /REST APIs.

Data link cable

Delivered with the product, this cable connects Meter Unit to Sensor Unit, both terminals can be sealed.

2.8. Disclaimer

LEM cannot be held liable for damage, injury or any legal responsibility incurred directly or indirectly from non product quality issues such as other use of the DCBM than according to LEM written installation instructions (see Operation manual section "4. Device description and mechanical integration") or other external factors.

The user shall observe safe and lawful practices, including, but no limited to, those set forth in this document. Before any operation or use, please read “Safety” section carefully.

LEM reserves the right to carry out modifications on its product and documentation at the sole discretion of LEM. Always make sure to have the latest information before placing an order. For up-to-date product information, visit www.lem.com or contact your nearest LEM sales representative.

2.9. Intellectual Property Rights

© Copyright 2023 LEM INTERNATIONAL SA. All rights reserved.

This document shall not be reproduced, copied, adapted, translated, arranged or modified without written permission from LEM, and the content, in whole or in any part, shall not be used for any purpose other than describing LEM DCBM. LEM will retain all intellectual property rights in and to this document. No license is granted by LEM to any intellectual property right. All rights not expressly granted are reserved by LEM.

Disassembly, decompilation, reverse-engineering, decryption, or alteration of the DCBM, including its software, are prohibited.

“LEM” and any related logos are protected trademarks owned by LEM HOLDING SA or trade names and cannot be used for other purposes than LEM usage without specific prior written permission.

Unless otherwise expressly granted by LEM in writing, the sale of the DCBM will not confer any license under any patents, trademarks, trade names, or other proprietary rights owned or controlled by LEM, its affiliates or suppliers, all rights being reserved.

2.10. Third party licensing

The DCBM includes software developped by SEGGER Microcontroller GmbH (SystemView RTT), STMicroelectronics (STM32Cube) and ARM LIMITED (CMSIS), available under BSD license, with following conditions.

Copyright © 2014 - 2020 SEGGER Microcontroller GmbH

Copyright © 2016 STMicroelectronics

Copyright © 2020 ARM LIMITED

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The DCBM includes software developped by Amazon Inc (FreeRTOS), available under MIT license, with following conditions.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

2.11. Warranty

For information about applicable warranty for the DCBM, contact your nearest LEM sales representative.

In the absence of any written agreement with LEM governing the sale of the DCBM to you, LEM general terms and conditions of sale as referred on the order confirmation shall apply. LEM disclaims all warranties of any kind, except as expressly provided in the above agreement or terms and conditions of sale, whether express or implied, relating to the DCBM and its documentation, including any implied warranty of merchantability, fitness for a particular purpose or non-infringement.

3. Introduction

Ethernet is the communication channel for the DCBM. It supports HTTP/REST communication to receive requests and provide measurements and other data.

The REST-compatible API is an application programming interface that uses HTTP requests to obtain (GET), place (PUT) and publish (POST) data. A RESTful API conforms to the Representational State Transfer or "REST" model. This interface is using JSON format for the data payload.

The following APIs are available to communicate with the DCBM:

- N00, NxD, NxM : v1 is designed for basic billing services.
- NxD, NxM: v2 is designed for billing services considering the Ad-Hoc charging application with Tariff OCMF.

To avoid compatibility issues between DCBM and EVSE, API v1 is duplicated to API v2 with new parameters and behaviours of Tarif OCMF (TT,UV,UD).

The DCBM REST interface is structured as follows with:

URL with default IP (X = API version 1 or 2)	Methods	Description
192.168.1.2/vX/status	GET	Status of the DCBM
192.168.1.2/vX/settings	GET, PUT	Settings of the DCBM
192.168.1.2/vX/logbook	GET	Event logger of the DCBM
192.168.1.2/vX/livemeasure	GET	Live measurements
192.168.1.2/vX/legal	GET, PUT, POST	Transaction management (start & stop) and transaction data structure, current or stored, in LEM proprietary format
192.168.1.2/vX/ocmf	GET	Transaction data structure, current or stored, in certified, billable, OCMF-compliant format
192.168.1.2/vX/certificate	GET	HTTPS certificate of the DCBM

In this document, request and response headers are described for all types of REST requests supported by the DCBM. Those headers specify how to properly configure a client to ensure functional communication.

The DCBM supports the following methods of RESTful API

- POST: publish a data
- GET: obtain a data
- PUT: place a data

In this document the IP address and port are set to the default value (set in production):

URI: http://192.168.1.2:80/

When describing in section below a REST API header, the following formalism is used:

    <COMMAND> <PATH> HTTP/1.1

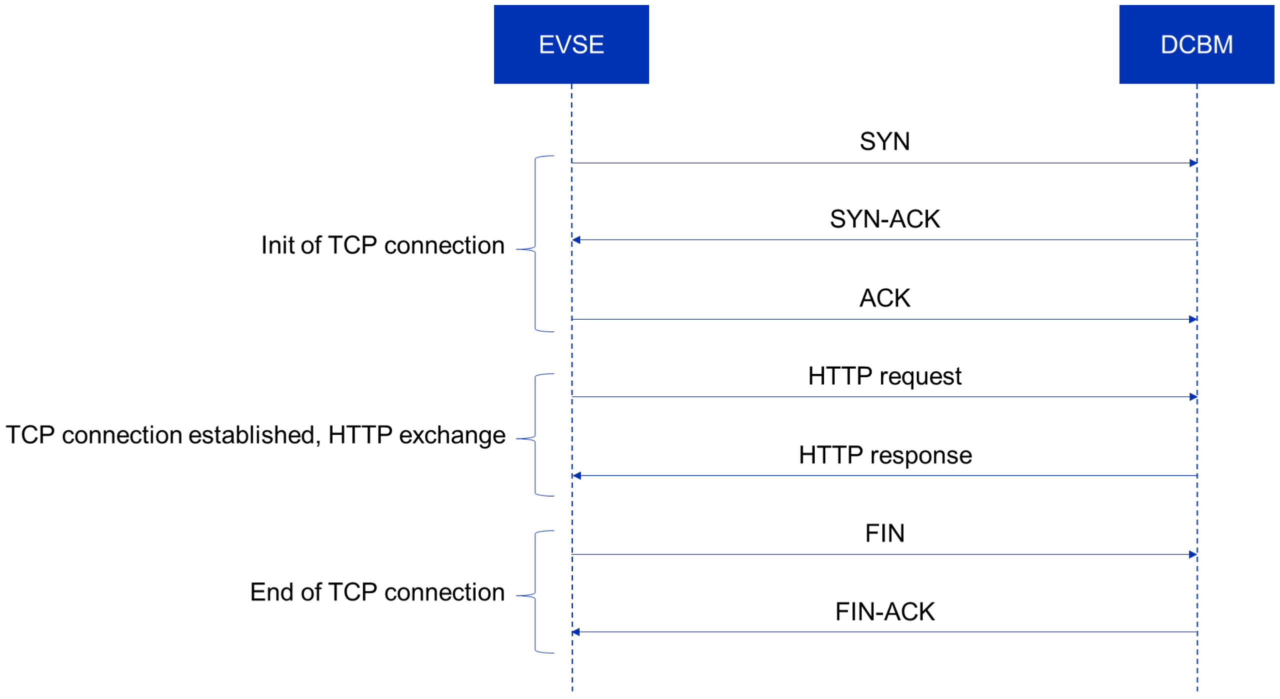
With

- <COMMAND> = REST command (ex: POST)
- <PATH> = path to add to the URI (ex: /v1/legal)

Remarks: All fields size in this document are given in bytes, without counting ending '\0' NULL characters needed for storing a string in C language.

⚠ HTTP request and response are implemented using socket and TCP connection sequence. This sequence is defined with a SYN/SYN-ACK then exchange HTTP Request/Response and always end with a FIN/FIN-ACK to comply with TCP connection flow standard.

Example of TCP socket standard connection flow:



3.1. Request header

3.1.1. Header formatting

The following request headers shall be formatted as follows:

```
POST /vX/legal HTTP/1.1
Host: <DCBM IP>
Content-Type: application/json
Content-Length: strlen(<BODY>)

<BODY>
```

With <BODY> = message sent to the DCBM

⚠ *Close the line above with "\r\n" (named "CRLF", or carriage return and line feed)*

⚠ *Extra CRLF needed before the BODY*

3.1.2. GET request

```
GET <PATH> HTTP/1.1
Host: <DCBM IP>
```

3.1.3. PUT request

```
PUT <PATH> HTTP/1.1
Host: <DCBM IP>
Content-Type: application/json
Content-Length: <SIZE>

<BODY>
```

with <SIZE> = strlen(<BODY>)

3.1.4. POST request

```
POST <PATH> HTTP/1.1
Host: <DCBM IP>
Content-Type: application/json
Content-Length: <SIZE>

<BODY>
```

with <SIZE> = strlen(<BODY>)

3.2. RESPONSE HEADER

3.2.1. HEADER FORMATING

The following response headers are formatted as follows:

```
HTTP/1.1 <ERROR_CODE> <STATUS>
Connection: close
```

With:

- <ERROR_CODE> = response code sent by the DCBM.
- <STATUS> = HTTP status (example : OK / Forbidden / ...)

HTTP/1.1 Transfer encoding in chunked block is supported, with max size of blocks = 0x100 = 256 bytes

3.2.2. SUCCESS CASE

```
HTTP/1.1 200 OK
Connection: close
Content-Type: application/json
Transfer-Encoding: chunked
```

3.2.3. FAILING CASE

```
HTTP/1.1 400 Bad Request
Connection: close
```

3.2.4. POSSIBLE HTTP STATUS CODES

Below are the implemented HTTP error codes, used in responses.

Code number	Meaning	PUT	POST	GET
200	OK	used	used	used
201	Created		used	
308	Permanent redirect	used	used	used
400	Bad request	used	used	used
403	Forbidden		used	
404	Not found			used
405	Method not allowed	used	used	
412	Preconditions failed	used	used	
500	Internal server error	used	used	used
501	Not implemented	used	used	used

3.2.5. CHUNKED TRANSFER

Typical HTTP chunked response is with max block size of 256 bytes, with length indicated at beginning of the data:

HTTP chunked response

```
100
<BODY_CHUNK>
100
<BODY_CHUNK>
...
```

The last block is identified with:

```
<REMAINING_LENGTH_IN_HEXADECIMAL_FORMAT>
<LAST_BODY_CHUNK>
0
```

⚠ *Length before chunked body responses are expressed in hexadecimal format, without the "0x" prefix*

Here is a summary of the maximum byte size for storing all the fields (worst cases). Add +1 if it is stored as a string to terminate with \0 character.

REST interface	max body size (byte)
Status	1400
settings	1339
legal	1272
ocmf	1100
logbook	10 Mbyte
livemeasure	210

4. /Settings API

⚠ *Some fields of /settings are freely settable and saved into flash memory (static memory).*

Thesefreely settable fields are: ipAddress, dhcp, ntp, time, http, ocmf

In order to ensure the /settings that are changes are well stored in memory it is recommended to:

- Ensure that the DCBM has been powered **for 2 minutes before writing a settings.**
- Wait for **0.5s before powering-down** the DCBM after writing a settings.

4.1. Overview

```
{
  "meterId": string,
  "cableConf": [
    {
      "cableSpId": integer,
      "cableSpName": string,
      "cableSpRes": integer
    },
    {
      "cableSpId": integer,
      "cableSpName": string,
      "cableSpRes": integer
    },
    {
      "cableSpId": integer,
      "cableSpName": "string,
      "cableSpRes": integer
    },
    {
      "cableSpId": integer,
      "cableSpName": string,
      "cableSpRes": integer
    },
    {
      "cableSpId": integer,
      "cableSpName": string,
      "cableSpRes": integer
    },
    {
      "cableSpId": integer,
      "cableSpName": string,
      "cableSpRes": integer
    },
    {
      "cableSpId": integer,
      "cableSpName": string,
      "cableSpRes": integer
    },
    {
      "cableSpId": integer,
      "cableSpName": string,
      "cableSpRes": integer
    }
  ],
  "ntp": {
    "servers": [
      {
        "ipAddress": string,
        "port": integer
      },
      {
        "ipAddress": string,
        "port": integer
      }
    ],
    "syncPeriod": integer,
    "ntpActivated": boolean,
    "syncTimeout": integer
  },
  "dhcp": {
    "ipAddress": string,
    "serverPort": integer,
    "clientPort": integer,
    "activation": boolean
  },
  "ipAddress": string,
  "http": {
    "tls_on": boolean,
    "httpPort": integer
  },
  "pulseOutputRate": integer,
  "pulseOutputFreq": integer,
  "time": {
    "utc": string,
    "tz": string,
    "dst": {
      "activated": boolean,
      "offset": integer,
      "start": {
        "order": string,
        "day": string,
        "month": string,
        "hour": string
      },
      "end": {
        "order": string,
        "day": string,
        "month": string,
        "hour": string
      }
    }
  },
  "ocmfId": {
    "IL": integer,
    "IF": {
      "Rfid": integer,
      "Ocpp": integer,
      "Iso15118": integer,
      "Plmn": integer
    },
    "IT": integer
  }
}
```

The following fields are read-only:

- `/meterId`
- `/cableConf`
- `/ntp/servers/syncTimeout`
- `/pulseOutputRate`
- `/pulseOutputFreq`

4.2. Fields description

4.2.1. meterId

The DCBM Serial Number, a string of max size 37.

Example:

```
"meterId": "12024072805"
```

⚠ *This field is not editable, set in factory.*

4.2.2. cableConf

This correspond to modelisation of the resistance of the cable which is used for estimating voltage drop between the Sensor Unit and the car cable connector, and is used to exclude from energy measurement the cable power lossses in case a configuration difference from four-wire measurement is used.

This is a 3 fields structure, nested in a 8 entries array.

Example:

```
"cableConf": [
    {
       "cableSpId": 0,
       "cableSpName": "no cable",
       "cableSpRes": 0
    },
    {
       "cableSpId": 1,
       "cableSpName": "2 mOhm",
       "cableSpRes": 20
    },
    ...
],
```

- cableSpId = cable index value, use for a START of transaction (values from 0 to 7)
- cableSpName = cable name (max size 19 bytes)
- cableSpRes = cable resistance value in mOhm (value from 0 to 255).

⚠ *This fields are not editable, set in factory*

4.2.3. NTP

Field used for UTC time synchronization by NTP protocol, allowing synchronisation with 2 servers. Using the same address for the 2 servers is allowed. Details can be found in Operation manual, section “NTP synchronization”.

Example:

```
"ntp": {
  "servers": [
    {
      "ipAddress": "192.168.1.1",
      "port": 123
    },
    {
      "ipAddress": "192.168.1.1",
      "port": 123
    }
  ],
  "syncPeriod": 21600,
  "ntpActivated": false,
}
```

- servers = an array of 2-objects structure
- ipAddress = IP address offering the SNTP service (UDP protocol) :
  - Using following format : "W.X.Y.Z", with each letter coding a integer of one byte size.
  - Or can be also an URL (Maximum size is 255 chars)
- port = corresponding UDP port (max = 65535)
- syncPeriod = period of NTP synchronization (in seconds , min 900, max 2^32-1)
- ntpActivated = boolean to enable/disable SNTP time synchronization. If disabled, "command time synchronization" shall be used (see /settings/time/utc)

The DCBM is rated as INFO time (and not SYSTEM time) by default.

In case ntpActivated = true while time synchronization by command is used, NTP is automatically disabled (i.e. ntpActivated = false)

4.2.4. DHCP

DHCP feature can be enabled or disabled with the activation flag "dhcp":{"activation":boolean} (with true = enabled, false = disabled).

- In case it is disabled (default settings), the IP of the DCBM is the one in the /settings "ipAddress" field.
- In case it is activated, the IP is received from the network. If no IP is received from the network, the IP displayed on maintenance screen is : 0.0.0.0(which allows no HTTP communication).
- Remark: It is possible to set a specific DHCP server address in the "ipAddress" of the "dhcp" field, as well as server port and client port.

Current IP address can be displayed in the maintenance screens. See Operation manual, section “Maintenance state”. On the contrary, ipAddress at the root of /settings API will not be updated according to dynamic DHCP address (only an input for static addressing, see description below).

⚠ *Field dhcp/ipAddress allows specifying a specific DHCP server address. Do not confuse it with/settings field ipAddress*

⚠ *Any value other than “0.0.0.0” shall be set carefully:*

1. *Take note of the set IP; this address might not be readable after application (until a proper server is set).*
2. *Communication will only be possible after setting a DHCP server at this specific address*

Example of a dhcp field:

```
"dhcp": {
  "ipAddress": "0.0.0.0",
  "serverPort": 67,
  "clientPort": 68,
  "activation": false
},
```

- ipAddress = (optional) the IP address server that offers the DCHP service (UDP protocol), using following format : "W.X.Y.Z", each letter coding a integer of one byte.
- serverPort = UDP port on server side (max = 65535)
- clientPort = UDP port on client side (max = 65535)
- activation = boolean, activation of the DHCP feature

4.2.5. IPADDRESS

Field to set the IP address when DHCP is disabled.

Example of an ipAddress field (default value):

```
"ipAddress": "192.168.1.2",
```

ipAddress = using following format: "W.X.Y.Z", each letter coding an integer of max one byte.

Z is limited from 1 to 254.

The subnet mask is “0.0.0.0”. It cannot be changed. This value allows access from any other IP address (i.e. no subnet mask restriction).

The DCBM can also be reached using its DNS name, just like using its IP address. The DCBM’s DNS name is “lem-meter”.

4.2.6. HTTP

Field to activate the HTTPS feature and configure the HTTP port

Example with default values:

```
"http": {
  "tls_on": false,
  "httpPort": 80
}
```

tls_on = boolean to enable/disable HTTPS

httpPort = port for HTTP usage (80 is default value)

In case of usage of HTTPS, the DCBM certificate needs to be accepted by the charging controller (it is not signed by a Central Authority, as the duration of the certificate is set to 999 years)

When HTTPS feature is enabled, the DCBM will accept any HTTPS certificate from the device that will connect to the DCBM.

Default port numbers as defined by IANA are:

- *80 for HTTP*
- *443 for HTTPS*

Toggling tls_on does not automatically change the port to use; do not forget to set it

4.2.7. PULSEOUTPUTRATE

This field is not writable, reserved.

Example with default value:

"pulseOutputRate" : 1

4.2.8. PULSEOUTPUTFREQ

This field is not writable, reserved

Example with default value :

"pulseOutputFreq" : 50

4.2.9. TIME

Field description:

- **"utc" =** the UTC legal time
- **"tz" =** the timezone of the location of the DCBM: it can go from -11 to +14 for hour, and 00, 15, 30, 45 for minutes
    - The timezone is the time shift compared to UTC time.

⚠ *If tz = "+00:00" this correspond to the UTC time zone, and time in /settings and /legal will be displayed as an UTC timestamp (with ending "Z" letter)*

ℹ *Some countries have several time zones and some countries use non-integer timezone (example : Iran is UTC+3:30)*

⚠ *In southern hemisphere, DST starts around October, (i.e. start and end are reversed compared to northern hemisphere).*

DST offset is used between "start" and "end" fields below

- **"dst"** = the Daylight Saving Time (DST) settings
- **"activated"** = JSON boolean (true/false) that activates the DST
- **"offset"**= the number of minutes that consists in the deviation applied for the DST activation. Shall be a postive value (usually 60 minutes, the default value).
    - **"start" =** start of DST
        - **"order"** = "first", "second", "last"
        - **"day"** = "monday"..."sunday"
        - **"month"** = "january"..."december"
        - **"hour"** = hour when DST starts, expressed in local time ("01:00") or UTC reference ("T00:00Z")

⚠ *The "hour" fields, when expressed in **local time without DST (without the final 'Z' suffix)**, uses **local time reference** without DST for both fields*

- **"end" =** end of DST
    - **"order"**= "first", "second", "last"
        - **"day"** = "monday"..."sunday"
        - **"month"** = "january"..."december"

- **"hour"** = hour when DST ends, expressed in local time ("01:00") or UTC reference ("T00:00Z")

⚠ *The "hour" fields, when expressed in **local time without DST (without the final 'Z' suffix)**, uses **local time reference** without DST for both fields*

Here is an example of the time JSON struct settings

```
"time" : {
  "utc":"2019-07-17T14:46:26Z",  // UTC manual time settings
  "tz":"+01:00",                 // Time zone offset (Here is UTC+1)
  "dst" : {                      // Daylight Saving Time fields
    "activated":true,            // Activation of the Daylight Saving Time (Here: enable DST)
    "offset":60,                 // Offset of DST (Here: apply +60mn for time with DST)
    "start" : {                  // Start of DST (i.e. when DST offset start being applied)
// (Here, DST starts on last Sunday of March at 1 am UTC)
      "order":"last",            // first, second or last
      "day":"sunday",            // day of week when DST start
      "month":"march",           // month to start DST
      "hour":"T01:00Z"           // hour to start DST (Here apply UTC time "T01:00Z" = local time
"02:00")
      },
    "end" : {                    // End of DST (i.e. when DST offset is no more applied)
// (Here, DST ends on last Sunday of October at 1 am UTC)
      "order":"last",            // first, second or last
      "day":"sunday",            // day of week when DST ends
      "month":"october",         // month to end DST
      "hour":"T01:00Z"           // hour to end DST (Here apply UTC time "T01:00Z" = local time
"02:00")
    }
  }
}
```

4.2.9.1. Setting the UTC legal time through command time synchronization

The UTC legal time of the DCBM can be set with a JSON command, by writing into the field

```
"time" : {"utc":string}
```

Example of time set :

```
"time": {
  "utc":"2019-07-17T14:46:26Z"
}
```

⚠ *In case of NTP is not used (manual UTC time set), the time shall be written **at least once a day**.*

If after a period of 48h (ie. the value of field /settings/ntp/syncTimeout) the time was not set, the DCBM will be out-of-sync (no transaction possible).

4.2.10. OCMFID

The “ocmfId” field is used for /ocmf API for the identification of the user.

The charger controller is responsible for those settings.

This information depends on the protocol used for user identification, as well as on the level of security used and assessed.

The charger controller (or the user) can configure OCMF settings of the DCBM in /settings API.

These data are related to identification of the user and shall be set by the charger controller.

When not configured, all the fields are set by default to "0".

```
"ocmfId":{
  "IL": 0,
  "IF":{
    "Rfid": 0,
    "Ocpp": 0,
    "Iso15118": 0,
    "Plmn": 0
  },
  "IT": 0
}
```

- "IL" = Identification level, an integer from 0 to 10
- "IF" = Identification Flag to describe the identification medium used
  - "Rfid" = an integer from 0 to 3
  - "Ocpp" = an integer from 0 to 7
  - "Iso15118" = an integer from 0 to 1
  - "Plmn = an integer from 0 to 3
- "IT" = Identification type, an integer from 0 to 17

This field is static and saved into the configuration of the DCBM (it is possible to set it once only if these parameters are not changing).

Due to current software limitation the OCMF IT settings shall by configured once (or kept as default) and not changed during the lifetime of the DCBM.

4.2.10.1. IL FIELD: IDENTIFICATION LEVEL

0	"-"	The field is not specified.
1	"NONE"	Here is no user mapping. The other data on user assignment have no significance.
2	"HEARSAY"	The assignment is unsecured; e.g. by reading out an RFID UID.
3	"TRUSTED"	The assignment can be trusted to some extent, but there is no absolute reliability. Example: Authorization by Backend.
4	"VERIFIED"	The assignment has been verified by the signature component and specific actions.
5	"CERTIFIED"	The mapping was verified by the signature component using a cryptographic signature that certifies the mapping.
6	"SECURE"	The assignment was established by a safe feature (e.g. secure RFID card, ISO15118 with plugand batch, etc.)
7	"MISMATCH"	Error: UIDs don't match.
8	"INVALID"	Error: Certificate not correct (check negative).
9	"OUTDATED"	Error: Referenced trust certificate expired.
10	"UNKNOWN"	Error: Certificate could not be verified (no matching trust certificate found).

4.2.10.2. IF FIELD : IDENTIFICATION FLAGS: DETAILED USER MAPPING STATEMENTS

This section describes identification flag fields.

4.2.10.2.1. RFID

0	"RFID_NONE"	No assignment by RFID
1	"RFID_PLAIN"	Assignment via external RFID card reader
2	"RFID_RELATED"	Assignment via protected RFID card reader
3	"RFID_PSK"	A previously known common key (pre-shared key) was used, e.g. with a secured RFID card

4.2.10.2.2. OCPP

0	"OCPP_NONE"	No user assignment by OCPP
1	"OCPP_RS"	Mapping by OCPP RemoteStart method
2	"OCPP_AUTH"	OCPP Authorize method mapping
3	"OCPP_RS_TLS"	Transport layer security was used to transfer the mapping using the OCPP RemoteStart method
4	"OCPP_AUTH_TLS"	Transport layer security was used to transfer the mapping using the OCPP Authorize method
5	"OCPP_CACHE"	OCPP authorization cache mapping
6	"OCPP_WHITELIST"	Assignment by white-list of OCPP
7	"OCPP_CERTIFIED"	A certificate of the backend which certifies the user assignment was used

4.2.10.2.3. ISO15118

0	"ISO15118_NONE"	No user assignment by ISO15118
1	"ISO15118_PNC"	Plug & Charge was used

4.2.10.2.4. PLMN

0	"PLMN_NONE"	No assignment
1	"PLMN_RING"	Call
2	"PLMN_SMS"	Message

4.2.10.3. IT FIELD : IDENTIFICATION TYPE: TYPE OF IDENTIFICATION DATA

0	"NONE"	No assignment available
1	"DENIED"	Assignment is not currently available (eg. two-factor authorization)
2	"UNDEFINED"	Type not specified
3	"ISO14443"	UID of an RFID card according to ISO 14443. Shown as 4 or 7 bytes in hexadecimal notation
4	"ISO15693"	UID of an RFID card according to ISO 15693. Shown as 8 bytes in hexadecimal notation
5	"EMAID"	Electro-Mobility Account ID according to ISO/IEC 15118 (string with length 14 or 15)
6	"EVCCID"	ID of an electric vehicle according to ISO/IEC 15118 (maximum length 6 characters)
7	"EVCOID"	EV contract ID according to DIN 91286.
8	"ISO7812"	Identification card format according to ISO/IEC 7812 (credit and bank cards, etc.)
9	"CARD_TXN_NR"	Card transaction number (CardTxNbr) for a payment with a credit or debit card used in a terminal at the charging point
10	"CENTRAL"	Centrally generated ID. No exact format defined, for example, can be a UUID (OCPP 2.0)
11	"CENTRAL_1"	Centrally generated ID, e.g. by starting via SMS. No exact format defined (up to OCPP 1.6)
12	"CENTRAL_2"	Centrally generated ID, e.g. by start by operator. No exact format defined (up to OCPP 1.6)
13	"LOCAL"	Locally generated ID. No exact format defined, for example, can be a UUID (OCPP 2.0)
14	"LOCAL_1"	Locally generated ID, e.g. ID generated internally by the load point. No exact format defined (up to OCPP 1.6)
15	"LOCAL_2"	Locally generated ID, for other cases. No exact format defined (up to OCPP 1.6)
16	"PHONE_NUMBER"	International telephone number with leading "+"
17	"KEY_CODE"	User-related, private key code. No exact format defined

4.3. Allowed requests

4.3.1. PUT - Write /settings

Settings can be set individually or grouped in a single HTTP request.

If several settings are written at once, their order shall be maintained.

Any writing attempt of settings shall be validated using DCBM acknowledge. The DCBM acknowledge is the result field in its response (see “Response format” below).

Request:

X = API version (1 for all version and 2 only for NxD, NxM)

```
PUT /vX/settings HTTP/1.1
Content-Type: application/json
Content-Length: strlen(<BODY>)

<BODY>
```

Example:

```
PUT /v1/settings HTTP/1.1
Content-Type: application/json
Content-Length: 41


{"time": {"utc": "2019-10-23T15:07:05Z"}}
```

4.3.1.1. Response body

HTTP chuncked response for the previous example.

```
{"meterId":"12024072805","result":1}
```

When "result"= 1 it indicates the DCBM acknowledged the request

When "result"= 0 it indicates that the request was rejected (for instance if a value set is out of range)

4.3.1.2. Possible HTTP status codes

Code number	Meaning
200	OK(but `result` must be checked)
400	Bad request

4.3.2. GET - READ /SETTINGS API

GET /vX/settings HTTP/1.1

X = API version (1 for all version and 2 only for NxD, NxM)

4.3.2.1. RESPONSE BODY

See section 4.1 Overview

4.3.2.2. POSSIBLE HTTP STATUS CODES

Code number	Meaning
200	OK

4.4. EXAMPLES

4.4.1. READ SETTINGS

Linux/Windows bash

```
curl -X GET http://192.168.1.2/v1/settings
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/settings" | Select-Object -Expand Content
```

4.4.2. WRITE SETTINGS

4.4.2.1. WRITE ALL AT ONCE

In this example we write all the following settings in one command:

- ntp
- dhcp
- ipAddress
- http
- ocmfId
- time

Linux/Windows bash

```
curl -d
'{"ntp":{"servers":~[{"ipAddress":"192.168.1.1","port":123},{"ipAddress":"192.168.1.1","port":123}],"
syncPeriod":900},"dhcp":{"ipAddress":"0.0.0.0","serverPort":67,"clientPort":68,"activation":false},"i
pAddress":"192.168.1.2","http":{"tls_on":false,"httpPort":80},"ocmfId":{"IL":1,"IF":{"Rfid":0,"Ocpp":1,
"Iso15118":1,"Plmn":0},"IT":5},"time":{"tz":"+01:00","dst":{"activated":true,"offset":60,"start":{"order"
:"last","day":"sunday","month":"march","hour":"T01:00Z"},"end":{"order":"last","day":"sunday","mon
th":"october","hour":"T01:00Z"
}' -H "Content-Type: application/json" -X PUT http://192.168.1.2/v1/settings
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/settings" -ContentType "application/json" -Method
PUT -Body
'{"ntp":{"servers":~[{"ipAddress":"192.168.1.1","port":123},{"ipAddress":"192.168.1.1","port":123}],"
syncPeriod":900},"dhcp":{"ipAddress":"0.0.0.0","serverPort":67,"clientPort":68,"activation":false},"i
pAddress":"192.168.1.2","http":{"tls_on":false,"httpPort":80},"ocmfId":{"IL":1,"IF":{"Rfid":0,"Ocpp":1,
"Iso15118":1,"Plmn":0},"IT":5},"time":{"tz":"+01:00","dst":{"activated":true,"offset":60,"start":{"order"
:"last","day":"sunday","month":"march","hour":"T01:00Z"},"end":{"order":"last","day":"sunday","mon
th":"october","hour":"T01:00Z"
}' | Select-Object -Expand Content
```

4.4.2.2. WRITE ONLY ONE SETTING

Writing one setting in a command (Set time/utc)

In this example we set the UTC time value to "2019-03-25T14:53:06Z"

Linux/Windows bash

```
curl -d '{"time":{"utc":"2019-03-25T14:53:06Z"}}' -H 'Content-Type: application/json' -X PUT
http://192.168.1.2/v1/settings
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/settings" -ContentType "application/json" -Method
PUT -Body '{"time":{"utc":"2019-03-25T14:53:06Z"}}' | Select-Object -Expand Content
```

5. /STATUS API

5.1. OVERVIEW

All status fields are read-only (GET method only)

```
{
  "status": {
    "value": integer,
    "bits": {
      "suLinkStatusIsOk": boolean,
      "muFatalErrorOccured": boolean,
      "transactionIsOnGoing": boolean,
      "tamperingIsDetected": boolean,
      "timeSyncStatusIsOk": boolean,
      "overTemperatureIsDetected": boolean,
      "reversedVoltage": boolean,
      "suMeasureFailureOccurred": boolean
    }
  },
  "version": {
    "applicationFirmwareVersion": string,
    "applicationFirmwareAuthTag": string,
    "legalFirmwareVersion": string,
    "legalFirmwareAuthTag": string,
    "sensorFirmwareVersion": string,
    "sensorFirmwareCrc": string
  },
  "time": string,
  "ipAddress": string,
  "meterId": string,
  "errors": {
    "value": integer,
    "bits": {
      "muInitIsFailed": boolean,
      "suStateIsInvalid": boolean,
      "versionCheckIsFailed": boolean,
      "muRngInitIsFailed": boolean,
      "muDataIntegrityIsFailed": boolean,
      "muFwIntegrityIsFailed": boolean,
      "suIntegrityIsFailed": boolean,
      "logbookIntegrityIsFailed": boolean,
      "logbookIsFull": boolean,
      "memoryAccessIsFailed": boolean,
      "muStateIsFailed": boolean,
    }
  },
  "publicKey": string,
  "publicKeyOcmf": string,
  "indexOfLastTransaction": integer,
  "numberOfStoredTransactions": integer,
  "EV#": integer // Only for NxD/NxM
}
```

5.2. Fields description

5.2.1. STATUS

This field indicates the current status of the DCBM

Example:

```
"status": {
  "value": 17,
  "bits": {
    "suLinkStatusIsOk": true,
    "muFatalErrorOccured": false,
    "transactionIsOnGoing": false,
    "tamperingIsDetected": false,
    "timeSyncStatusIsOk": true,
    "overTemperatureIsDetected": false,
    "reversedVoltage": false,
    "suMeasureFailureOccurred": false
  }
},
```

- "value" = is the decimal value (max 255) and is calculated as the integer value corresponding to the binary value made of the associated bit flags. More explanations can be found in the Operation manual, section “Status/errors value field description”.

The nominal value is =17 (suLinkStatusIsOk is true and timeSyncStatusIsOk is true). If value is =1 (suLinkStatusIsOk), the time synchronization period has expired (UTC time shall be set again) "bits" fields, corresponding to value field broken down by unitary flags

Name	Bit index	Meaning	Nominal value	Severity	Associated event logbook entry name
suLinkStatusIsOk	0	true: the sensing head is present and working properly	true	Blocking	STATUS_SENSOR_LINK
muFatalErrorOccured	1	true: there is a fatal error (FF) detected	false	Blocking	STATUS_FATAL_ERROR
transactionIsOnGoing	2	true: a transaction is running	false	Blocking	STATUS_START_STOP_TRANSACTION
tamperingIsDetected	3	true: the anti-tampering GPIO has detected a cover open	false	Informational	STATUS_TAMPERING
timeSyncStatusIsOk	4	true: the UTC time is properly synchronized (INFO time)	true	Blocking	STATUS_TIME_SYNC
overTemperatureIsDetected	5	true: an over temperature event has occured (above 140°)	false	Informational	STATUS_OVERHEAT
reversedVoltage	6	true: the voltage probe is inverted	false	Informational	STATUS_REVERSED_VOLTAGE
suMeasureFailureOccurred	7	true: an ADC measure failure of the sensing	false	Informational	STATUS_MEASURE_ERROR

⚠ *If reversedVoltage = true, this indicates that a voltage level below -50V is seen. The polarity of the voltage sensor probes shall be checked, this may indicate a wrongful connection on busbar.*

⚠ *"Blocking" level indicates that if different of normal value, it will reject a transaction request*

5.2.2. VERSION

This field is used to track version and checksum of the DCBM firmware parts.

Example:

```
"version": {
  "applicationFirmwareVersion": "0.1.4.0",
  "applicationFirmwareAuthTag": "663A7BA7A685BD6A7C43F136",
  "legalFirmwareVersion": "0.1.4.0",
  "legalFirmwareAuthTag": "E2C03AFCB73E0464827200E5",
  "sensorFirmwareVersion": "0.0.8.0",
  "sensorFirmwareCrc": "540F"
},
```

5.2.3. TIME

Display the local time and time deviation in ISO8601 extended dateformat.

⚠ *if /settings/time/tz = "+00:00" this corresponds to the UTC time zone (also known as GMT) and DST is not present, the time in this field will be displayed as a UTC timestamp (with ending "Z" letter).*

⚠ *It is recommanded to interpret the "Z" letter as "+00:00" if read*

Example:

```
"time": "2019-10-24T15:45:33+02:00",
```

5.2.4. IPADDRESS

Display the current IP address of the DCBM.

Example:

```
"ipAddress": "192.168.1.2",
```

ℹ *IP address can be displayed through the technical screens.*

ℹ *Default IP address of the DCBM is 192.168.1.2.*

⚠ *In case of DHCP activated, and no IP adress received from the network, the IP displayed is : 0.0.0.0*

5.2.5. METERID

Display the meterId value (serial number) of the DCBM. Max size is 37 characters.

Example:

```
"meterId": "12024072805",
```

5.2.6. ERRORS

Display the error status of the DCBM.

Example:

```
"errors": {
  "value": 0,
  "bits": {
    "muInitIsFailed": false,
    "suStateIsInvalid": false,
    "versionCheckIsFailed": false,
    "muRngInitIsFailed": false,
    "muDataIntegrityIsFailed": false,
    "muFwIntegrityIsFailed": false,
    "suIntegrityIsFailed": false,
    "logbookIntegrityIsFailed": false,
    "logbookIsFull": false,
    "memoryAccessIsFailed": false,
    "muStateIsFailed": false,
  }
},
```

Any error prevents a correct usage of the DCBM: "Fatal Error" level. A new transaction will be refused, the DCBM shall be changed if the error persist.

- "value" field: In normal operation shall be 0. In case it is non-null, the decimal value is displayed on the screen. When bit parsing the value (0 is LSB) it is possible to know the errors sets.More explanations can be found in the Operation manual, section “Status/errors value field description”.
- "bits" fields: corresponding to value field broken down by unitary flags (please note: indexes are not contiguous):

Error bit name	Bit index	Description	Severity	Frequency of checks	Associated event logbook entry name
muInitIsFailed	0	error when initializing the internal data structure	Fatal Error	At startup	EV_INIT_ERROR
suStateIsInvalid	1	error in the state of the SU	Fatal Error	Instantaneous	EV_SU_INVALID_STATE
versionCheckIsFailed	2	error in SW version (SU version vs MU version pair)	Fatal Error	At startup	EV_VERSION_INCONSISTENCY
muRngInitIsFailed	3	error at hardware random number generator initialization	Fatal Error	At startup	EV_RNG_INIT_TEST_FAIL
muDataIntegrityIsFailed	4	error with CRC check of the parameter structure of the Meter Unit or in stored transaction.	Fatal Error	At startup (5s offset) + cyclic (30mn)	EV_LR_FIRMWARE_CHECK_FAILED
muFwIntegrityIsFailed	5	errror with CRC check of the Meter Unit firmware	Fatal Error	At startup (5s offset) + cyclic (30mn)	EV_LR_FIRMWARE_CHECK_FAILED
suIntegrityIsFailed	6	error with CRC check of the Sensor Unit parameters and firmware	Fatal Error	Instantaneous	EV_SU_INTEGRITY_ERROR
logbookIntegrityIsFailed	8	error in integrity of the event logbook	Fatal Error	At startup (5s offset) + cyclic (30mn)	EV_EXTERNAL_MEMORY_INTEGRITY
logbookIsFull	9	error on event logbook storage (max capability reached)	Fatal Error	At startup (5s offset) + On new event	EV_LOGBOOK_FULL
memoryAccessIsFailed	10	error on software interface memory access	Fatal Error	Instantaneous	EV_INVALID_MEMORY_ACCESS
muStateIsFailed	13	metrology memory error (stack overflow)	Fatal Error	At startup (5s offset) + cyclic (30mn)	EV_STACK_OVERFLOW

⚠ Any raised fatal error prevents further use of the DCBM. The latter shall be checked.

5.2.7. PUBLICKEY

Display the public key of the DCBM in the ASN.1 DER octetstring format

Example:

"publicKey":
"797B79B8E0ACBDA9646ED19B03B85C39CCE56F5A179988E874BA75FB8303199C255A492
936EE27D58AAAFC0DE53B29931D3022ADD96CB6AD95CC59B757C6A154",

5.2.8. PUBLICKEYOCMF

Display the public key of the DCBM in the ASN.1 DER octetstring format with RFC5480 header

Example:

"publicKeyOcmf":
"3059301306072A8648CE3D020106082A8648CE3D03010703420004797B79B8E0ACBDA9646
ED19B03B85C39CCE56F5A179988E874BA75FB8303199C255A492936EE27D58AAAFC0DE53
B29931D3022ADD96CB6AD95CC59B757C6A154",

This is the format expected by the OCMF transparency software: https://safe-ev.org/

5.2.9. INDEXOFLASTTRANSACTION

Indicate the last transaction storage index in memory (max size 2^32-1)

The transaction is then stored in /vX/legal/<index> address

X = API version (1 for all version and 2 only for NxD, NxM)

Example:

"indexOfLastTransaction": 14,

Remark: A value of -1 means that no transaction is stored at all.

5.2.10. NUMBEROFSTOREDTRANSACTIONS

Indicate the number of stored transactions currently stored in the DCBM.

When the transaction buffer is full, wrap around occurs and the oldest transaction is deleted when writing of a new one. (see Operation manual section “Memory depth”).

Example:

"numberOfStoredTransactions": 15

5.2.11. EVENTNUMBER

EV# indicate the number of logbook events.
This field is available only for NxD or NxM.

Example:

```
{
"status": {
"value": 17,
...
"indexOfLastTransaction": 0,
"numberOfStoredTransactions": 0,
"EV#": 156
}
```

5.3. ALLOWED REQUESTS

5.3.1. GET - READ /STATUS

GET /vX/status HTTP/1.1

X = API version (1 for all version and 2 only for NxD, NxM)

5.3.1.1. RESPONSE BODY

See section 5.1 Overview

5.3.1.2. POSSIBLE HTTP STATUS CODES

Code number	Meaning
200	Ok

5.4. EXAMPLES

Linux/Windows bash

```
curl -X GET http://192.168.1.2/v1/status
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/status" | Select-Object -Expand Content
```

6. /LEGAL API

6.1. OVERVIEW

This fields contains all legal information needed for the billing process and remote display of energy measurement information. This format is proprietary (LEM) ie. manufacturer specific.

This field marks the end of the metrological chain of the DCBM.

Energy register can handle up to 9 digits with 3 decimals, it is equivalent to 999999999,999 kWh.
This max value will be never reached after following estimation:
DC max power = 600 kW (600A, 1000V)
Time to reach this value @ DC max power is equivalent to 190 years.

It is possible to read the /legal fields during a transaction (intermediate reading), or after a transaction (final reading).

Below is the JSON structure type for /legal interface.

```
{
  "paginationCounter": integer,
  "transactionId": string,
  "evseId": string,
  "clientId": string,
  "tariffId": integer,
  "cableSp": {
    cableSpName : string
    cableSpId : integer,
    cableSpRes : integer,
  },
  "userData" : string,
  "meterValue": {
    "timestampStart": string,
    "timestampStop": string,
    "transactionDuration": integer,
    "intermediateRead": boolean,
    "transactionStatus" : boolean,
    "sampleValue": {
      "energyUnit": string,
      "energyImport": number,
      "energyImportTotalStart": number,
      "energyImportTotalStop": number,
      "energyExport": number,
      "energyExportTotalStart": number,
      "energyExportTotalStop": number
    }
  },
  "meterId": string,
  "signature": string,
  "publicKey": string,
}
```

6.2. Fields description

6.2.1. Response description

Here is an example of a transaction result:

- this is the 6th read of the DCBM
- transactionId is "azAZ09*-_=:+|,@"
- evseId is "+49*DEF*E123ABC",
- clientId is "12"
- tariffId is 2
- cable compensation parameters : cableId = 1, name = "no cable", resistance = 0 ohm
- transaction occured the 10th of December of 2019
- transaction has started at 16:39:15 local time
- transaction has ended at 16:40:25 local time
- the transaction duration was 70 seconds
- the transaction status is not nominal (ie different of 17, see /status/status API for more details)
- the energy transfer from the charging station to the EV was 7.637 kWh
- the energy transfer from the EV to the charging station was null

```
{
  "paginationCounter": 6,
  "transactionId": "azAZ09*-_=:+~|,@",
  "evseId": "+49*DEF*E123ABC",
  "clientId": "12",
  "tariffId": 2,
  "cableSp": {
    "cableSpName": "no cable",
    "cableSpId": 1,
    "cableSpRes": 0
  },
  "userData": "",
  "meterValue": {
    "timestampStart": "2019-12-10T16:39:15+01:00",
    "timestampStop": "2019-12-10T16:40:25+01:00",
    "transactionDuration": 70,
    "intermediateRead": false,
    "transactionStatus": 25,
    "sampleValue": {
      "energyUnit": "kWh",
      "energyImport": 7.637, // on NxD or NxM
      "energyImportTotalStart": 188.977, // on NxD or NxM
      "energyImportTotalStop": 196.614, // on NxD or NxM
      "energyExport": 0.000, // on NxD or NxM
      "energyExportTotalStart": 0.000, // on NxD or NxM
      "energyExportTotalStop": 0.000 // on NxD or NxM
    }
  },
  "meterId": "12024072805",
  "signature":
"304502203DC38FBC722D216568D6ECB4B352577A999B6D184EA6AD48BDCAE7766
DB1D628022100A7687B4CB5573829D407DD4B17D41C297917B7E8307E5017711B5A
3A987F6801",
  "publicKey":
"A80F10D968E1122F8820F288B23C4E1C0DA912F35B48481274ADFEFE66D7E87E13
0C7CF2B8047C45CF105041C8C3A57DD242782F755C9443F42DABA9404A67BF"
}
```

6.2.2. PAGINATIONCOUNTER

Pagination counter is only incremented for each transaction start accepted, the signature is generated and is the same even for a past transaction read twice.

Example:

    "paginationCounter":26

6.2.3. TRANSACTIONID - INPUT PARAMETER

This is an input parameter (string) to identify the transaction. Max size = 37 char.

Example:

    "transactionId": "azAZ09*-_=:+~|,@",

Authorised characters are : ASCII encoding

6.2.4. EVSEID - INPUT PARAMETER

This is an input parameter (string) to identify the charging point. Max size = 37 char.

Example:

    "evseId": "+49*DEF*E123ABC",

Authorised characters are : ASCII encoding

6.2.5. CLIENTID - INPUT PARAMETER

This is an input parameter (string) to identify the end user customer (client). Max size = 37 char.

Example:

    "clientId": "client12657",

Authorised characters are : ASCII encoding

⚠ The clientId is also displayed during a transaction on the screen, so the value used with the DCBM must not be confidential

6.2.6. TARIFFID - INPUT PARAMETER

This is an input parameter, an integer (from 0 to 3) used for a unique transaction tariff designation.

⚠ *The DCBM is rated with INFO time (not SYSTEM time), so no tariff changes are possible during a transaction.*

Example:

```
"tariffId": 2,
```

6.2.7. CABLEID - INPUT PARAMETER

This field refers to the /settings/cableConf table.

This is an input parameter: an integer (from 0 to 7).

The value shall correspond to one of the *cableConf/../cableSpId* array value

This allows compensating the measurements of the DCBM with a resistance value, selectable within a table.

Example:

```
"cableId": 2,
```

6.2.8. CABLESP

This field reflects the selected */settings/cableConf* table, selected with *cableId* value (see previous paragraph).

6.2.8.1. CABLESPNAME

A string of maximum size 19 chars. (read only)

This field reflects the selected /settings/cableConf/cableSpName table, selected with cableId value (see previous paragraph).

6.2.8.2. CABLESPID

An integer from 0 to 7. (read only)

This field reflects the selected /settings/cableConf/cableSpId table, selected with cableId value (see previous paragraph).

6.2.8.3. CABLESPRES

An integer from 0 to 255 encoding the resistance value in mOhm (read only).

This field reflects the selected /settings/cableConf/cableSpRes table, selected with cableId value (see previous paragraph).

6.2.9. USERDATA - INPUT PARAMETER

This is an input parameter (string) that can be used to include specific information within the legal data. Max size = 128 bytes.

All the UTF-8 and UTF-16 characters set can be used but must fit into 128 byte to be accepted.

Example:

"userData" : "",

6.2.10. METERVALUE

6.2.10.1. TIMESTAMPSTART

Timestamp at the time of the start command, expressed in ISO 8610 date time local format, with timezone information

If /settings/time/tz = "+00:00" this corresponds to the UTC time zone, then time in this field will be displayed as an UTC timestamp (with terminal "Z" letter), without the +00:00 field.

Example:

"timestampStart": "2019-10-28T10:41:55+01:00",

6.2.10.2. TIMESTAMPSTOP

Timestamp at the time of the stop (or the time of the read in case of intermediate reading) command, expressed in ISO 8610 date time local format, with timezone information

If /settings/time/tz = "+00:00" this corresponds to the UTC time zone, then time in this field will be displayed as an UTC timestamp (with terminal "Z" letter), without the +00:00 field

Example:

"timestampStop": "2019-10-28T11:39:57+01:00",

6.2.10.3. TRANSACTIONDURATION

Informational register, reflecting the difference between stop timestamp and start timestamp (in seconds).

Example:

```
"transactionDuration": 3482,
```

In case of a transaction on-going (see next field) this register increases along with the stop timestamp (timestampStop is the timestamp of reading in case of an on-going transaction)

6.2.10.4. INTERMEDIATEREAD

Boolean expressing whether the reading is from a past transaction (=*false*) or for a current transaction (ie. an intermediate reading) (=*true*).

Example for a past transaction:

```
"intermediateRead": false,
```

6.2.10.5. TRANSACTIONSTATUS

This field indicates the status of the DCBM at time of read (intermediate reading case) or time of end of transaction (past transaction case).

Unlike the live status register from the /status API, this one accumulates the status bit changes during the transaction, ensuring traceability of any occuring event. Some of the flags can invalidate the transaction. Refer to status for detail of the bits meanings.

The nominal value, ie. corresponding to a DCBM ready for a new transaction, is 17. This corresponds to:

- *"suLinkStatusIsOk": true,*
- *"timeSyncStatusIsOk": true*,
- and the rest to *false*.

Meaning that the data link connection of the sensor part of the DCBM is correct, and the time synchronization status is OK.

Example:

```
"transactionStatus": 17,
```

6.2.10.6. SAMPLEVALUE

JSON fields containing the value of the DC energy measurement.

6.2.10.6.1. ENERGYUNIT

Field indicating the unit of the energy register.

This field is static, measurement unit cannot change.

Example:

"energyUnit": "kWh",

6.2.10.6.2. ENERGYIMPORT

Field indicating the difference of imported energy between the stop and the start command, in kWh with 3 decimal digits for NxD or NxM, and 4 decimal digits for N00.

Example for 3 decimal digits:

"energyImport": 511.994,

6.2.10.6.3. ENERGYIMPORTTOTALSTART

Field indicating the imported energy total register at the time of the start command, in kWh with 3 decimal digits for NxD or NxM, and 4 decimal digits for N00.

Corresponding OBIS code is "1-0:1.8.0".

Example for 3 decimal digits:

"energyImportTotalStart": 18.775,

6.2.10.6.4. ENERGYIMPORTTOTALSTOP

Field indicating the imported energy total register at the time of the stop/read command, in kWh with 3 decimal digits for NxD or NxM, and 4 decimal digits for N00.

Corresponding OBIS code is "1-0:1.8.0".

Example for 3 decimal digits:

"energyImportTotalStop": 530.769,

In case of a transaction on-going this register increases along with timestampStop (intermediate reading case).

6.2.10.6.5. ENERGYEXPORT

Field indicating the difference of exported energy between the stop and the start commands, in kWh with 3 decimal digits for NxD or NxM, and 4 decimal digits for N00.

Example for 3 decimal digits:

    "energyExport": 0.000,

6.2.10.6.6. ENERGYEXPORTTOTALSTART

Field indicating the exported energy total register at the time of the start command, in kWh with 3 decimal digits for NxD or NxM, and 4 decimal digits for N00.

Corresponding OBIS code is "1-0:2.8.0".

Example for 3 decimal digits:

    "energyExportTotalStart": 0.000,

6.2.10.6.7. ENERGYEXPORTTOTALSTOP

Field indicating the imported energy total register at the time of the stop/read command, in kWh with 3 decimal digits for NxD or NxM, and 4 decimal digits for N00.

Corresponding OBIS code is "1-0:2.8.0".

Example for 3 decimal digits:

    "energyExportTotalStop": 0.000

6.2.11. METERID

Display the meterId value of the DCBM (corresponding to its serial number). Max size is 37 characters.

Example:

    "meterId": "12024072805",

6.2.12. SIGNATURE

Signature of the transaction in octet string format, with ASN1 DER encoding, using ECDSA secp256r1 and SHA256 methods.

Example:

```
  "signature":
"304502205C7B5B67C012E2691738B4CE5365AEE1191D0F59AAB81D6C0C0C1BC74303FDB
9022100A79E1BBA77EA6B110E19C81D84D44750C0361A04E5662783D13D5F1BFDEF66D7"
```

6.2.13. PUBLICKEY

Display the public key of the DCBM in the ASN.1 DER octetstring format

Example:

```
  "publicKey":
"797B79B8E0ACBDA9646ED19B03B85C39CCE56F5A179988E874BA75FB8303199C255A492
936EE27D58AAAFC0DE53B29931D3022ADD96CB6AD95CC59B757C6A154",
```

6.3. Allowed requests for API

6.3.1. POST - Start a transaction on v1/legal for NxD, NxM and N00

⚠ *To start a transaction, all fields are needed, and order shall be observed.*

- *evseId*
- *transactionId*
- *clientId*
- *tariffId*
- *cableId*
- *userData*

⚠ *The DCBM accepts multiple transactions with the same transactionId. In this case, on retrieval (GET) by transactionId, the latest is fetched.*

⚠ *For the transaction start to be accepted, the DCBM time shall be synchronized. If the DCBM time is out-of-sync, the transaction start will be rejected.*

Command:

```
POST /v1/legal HTTP/1.1
Content-Type: application/json
Content-Length: strlen(<BODY>)

<BODY>
```

Example:

```
POST /v1/legal HTTP/1.1
Content-Type: application/json
Content-Length: 91


{"evseId":"evse458877","transactionId":"transac5000","clientId":"client12","tariffId":2,"cableId":2,"u
serData":""}
```

6.3.1.1. RESPONSE HEADER

The response header for /legal POST is distinctive, it contains the transaction storage index in the HTTP header location field:

Success case:

```
HTTP/1.1 201 Created
Location: http://192.168.1.2:80/v1/legal/3
Connection: close
Content-Type: application/json
Transfer-Encoding: chunked
```

Failing case:

```
HTTP/1.1 403 Forbidden
Connection: close
```

6.3.1.2. RESPONSE BODY

The response body contains all the input fields set (except userData), and the "running" field.

⚠ *The userData field is not present into the response.*

Example:

```
{"evse458877","transactionId":"transac5000","clientId":"client12","tariffId":2,"cableId":2,"running":tr
ue}
```

6.3.1.3. POSSIBLE HTTP STATUS CODES

Code number	Meaning	Description
201	Created	The start request was accepted, a transaction storage was allocated.
308	Permanent Redirect	Port was redirected.
400	Bad request	An error was detected during fields parsing.
403	Forbidden	The transaction start was rejected (a transaction is already running).
405	Method not allowed	Wrong HTTP method used.
412	Precondition failed	The DCBM is not ready for a new transaction (ie. a status flag is raised or an error was detected).
501	Not implemented	HTTP invalid format request.

6.3.2. POST - Start a transaction on v2/legal for NxD or NxM

To start a transaction, all fields are needed, and order shall be observed.

- *evseId*
- *transactionId*
- *clientId*
- *tariffId*
- *TT*
- *UV*
- *UD*
- *cableId*
- *userData*
- *SC*

The DCBM accepts multiple transactions with the same transactionId. In this case, on retrieval (GET) by transactionId, the latest is fetched.

For the transaction start to be accepted, the DCBM time shall be synchronized. If the DCBM time is out-of-sync, the transaction start will be rejected.

Command:

```
POST /v2/legal HTTP/1.1
Content-Type: application/json
Content-Length: strlen(<BODY>)

<BODY>
```

Example:

```
POST /v2/legal HTTP/1.1
Content-Type: application/json
Content-Length: 203


{"evseId":"evse15674","transactionId":"transac5765","clientId":"client44678","tariffId":2,"TT":"0,55euro/kWhTTC","UV":"UserSWVersion","UD":"275gCO2/kWh","cableId":1,"userData":"Testofatransaction","SC":15}
```

6.3.3. PUT - Stop a transaction on legal

To stop the on-going transaction, the transactionId is required.

The stop command by transactionId goes through an URI. Special characters can be sent with or without query-string percent-encoding.

Command :

```
PUT /vX/legal?transactionId=<transactionId> HTTP/1.1
Content-Type: application/json
Content-Length: 21

{"running": false}
```

X = API version (1 for all version and 2 only for NxD, NxM)

Example:

```
PUT /v1/legal?transactionId=5000 HTTP/1.1
Content-Type: application/json
Content-Length: 21

{"running": false}
```

6.3.3.1. Response body

HTTP chuncked response content is the same as a read content structure.

Example:

```
{"paginationCounter":14,"transactionId":"azAZ09*-
_=:+~|,@","evseId":"+49*DEF*E123ABC","clientId":"12","tariffId":2,"cable
Sp":{"cableSpName":"2
mohm","cableSpId":1,"cableSpRes":2},"userData":"","meterValue":{"timesta
mpStart":"2019-12-10T17:22:54+01:00","timestampStop":"2019-12-
10T17:27:56+01:00","transactionDuration":302,"intermediateRead":false,"t
ransactionStatus":17,"sampleValue":{"energyUnit":"kWh","energyImport":33
.499,"energyImportTotalStart":96.659,"energyImportTotalStop":130.158,"en
ergyExport":0.000,"energyExportTotalStart":0.000,"energyExportTotalStop"
:0.000}},"meterId":"12024072805","signature":"304502200C22B3EAB7A27FE60C
5DF58B404563843A3A4C3DB636FCCA42B7D7B8DCDD37FE022100C31D72C47D7CF565F16E
A8ED5820B1F0739781756B55FA3F1B28FBA4A51E8AB1","publicKey":"D47C8ACBA2E18
E93BD57C361C2CA7E7BA19157DF7913E20DCECD387DEE5138F2CE3BCD98CFA51C17D006F
6878958C23818EDA88B3568E0B2F3A6CEC1D04EE44C"
```

6.3.3.2. Possible HTTP status codes

Code number	Meaning	Example
200	Ok	The read request was accepted.
308	Permanent Redirect	Port was redirected.
400	Bad request	An error was detected during field parsing (running field not set to false).
405	Method not allowed	Wrong HTTP method used.
412	Precondition failed	No transaction running.
501	Not implemented	HTTP invalid format request.

6.3.4. GET - READ LEGAL FOR NxD OR NxM

The DCBM can store up to 16319 transactions.

Command :

- By current / latest:

  GET /vX/legal HTTP/1.1

- By *transactionId*:

  GET /vX/legal?transactionId=<transactionId> HTTP/1.1

This solution only gives access to the latest 839 transactions.

The DCBM accepts multiple transactions with the same transactionId. In this case, on retrieval (GET) by transactionId, the latest is fetched.

- By internal transaction index ("absolute" storage index):

  GET /vX/legal/<index_value> HTTP/1.1

The <index_value> is the one returned when starting a transaction into the HTTP header field (see 1.4.3.1.1 - POST Start a transaction - Response header)

*The "absolute" storage index has a limited duration: it cover a maximum of **16319** transactions, after which wrapping around occurs.*

- By internal chronological transaction index value :

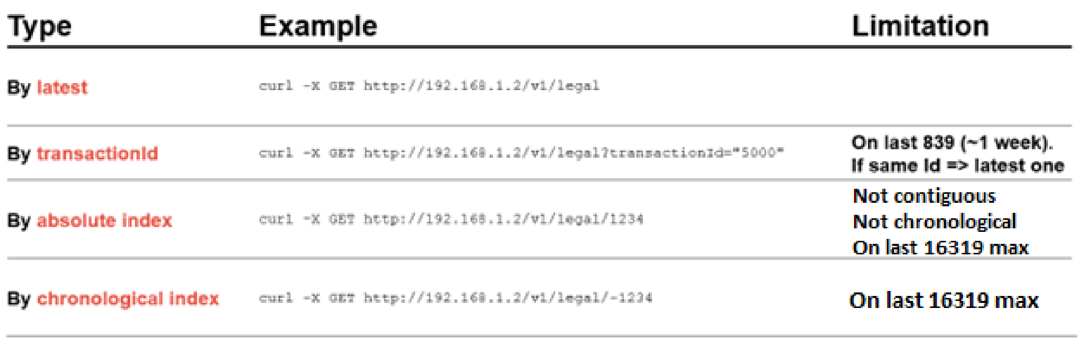
  GET /vX/legal/-<index_value> HTTP/1.1

This solution allows requesting the whole list of transactions, successively.

Note:

- "/-1" leads to the penultimate stored transaction.
- "/-16319" leads to the oldest possible transaction (is buffer has been filled).
- "/-16320" can never be reached.

Summary:



Type	Example	Limitation
By latest	`curl -X GET http://192.168.1.2/v1/legal`	
By transactionId	`curl -X GET http://192.168.1.2/v1/legal?transactionId="5000"`	On last 839 (~1 week). If same Id => latest one
By absolute index	`curl -X GET http://192.168.1.2/v1/legal/1234`	Not contiguous Not chronological On last 16319 max
By chronological index	`curl -X GET http://192.168.1.2/v1/legal/-1234`	On last 16319 max

6.3.5. GET - Read legal for N00

The DCBM can store up to 20137 transactions.

Command :

- By current / latest:

  GET /v1/legal HTTP/1.1

- By *transactionId*:

  GET /v1/legal?transactionId=<transactionId> HTTP/1.1

This solution only gives access to the latest 839 transactions.

The DCBM accepts multiple transactions with the same transactionId. In this case, on retrieval (GET) by transactionId, the latest is fetched.

- By internal transaction index ("absolute" storage index):

  GET /v1/legal/<index_value> HTTP/1.1

The <index_value> is the one returned when starting a transaction into the HTTP header field (see 1.4.3.1.1 - POST Start a transaction - Response header)

The "absolute" storage index has a limited duration: it cover a maximum of ***20137*** *transactions, after which wrapping around occurs.*

- By internal chronological transaction index value :

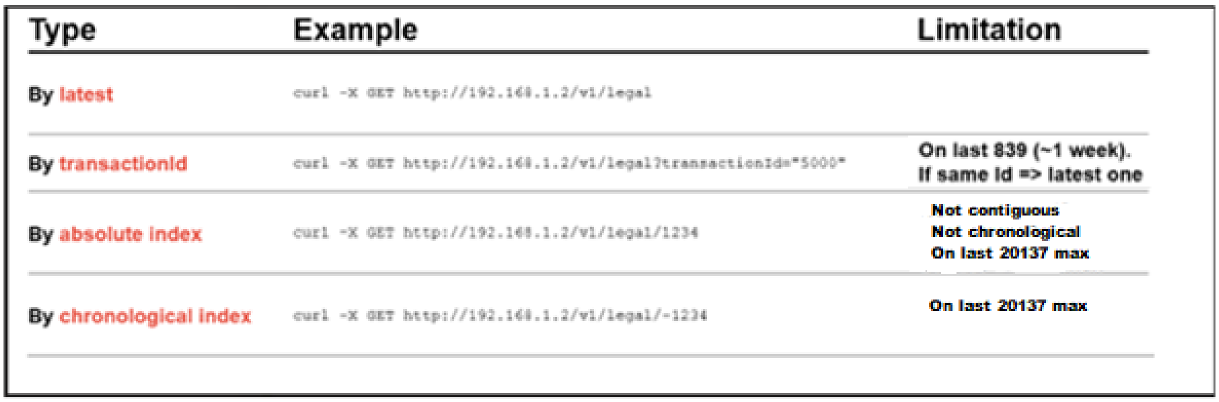
  GET /v1/legal/-<index_value> HTTP/1.1

This solution allows requesting the whole list of transactions, successively.

Note:

- "/-1" leads to the penultimate stored transaction.
- "/-20137" leads to the oldest possible transaction (is buffer has been filled).
- "/-20138" can never be reached.

Summary:



Type	Example	Limitation
By latest	`curl -X GET http://192.168.1.2/v1/legal`	
By transactionId	`curl -X GET http://192.168.1.2/v1/legal?transactionId="5000"`	On last 839 (~1 week). If same Id => latest one
By absolute index	`curl -X GET http://192.168.1.2/v1/legal/1234`	Not contiguous Not chronological On last 20137 max
By chronological index	`curl -X GET http://192.168.1.2/v1/legal/-1234`	On last 20137 max

6.3.5.1. Response body

See section 6.1 Overview

6.3.5.2. Possible HTTP Error Code

Code number	Signification
200	Ok

6.4. Examples

6.4.1. READ

- Get current or last transaction:

Linux/Windows bash

```
curl -X GET http://192.168.1.2/v1/legal
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/legal" | Select-Object -Expand Content
```

- Get a past transaction with the transactionId:

Linux/Windows bash

```
curl -X GET http://192.168.1.2/v1/legal?transactionId=transac5000
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/legal?transactionId=transac5000" | Select-Object -
Expand Content
```

- Get a past transaction with the chronological index:

Linux/Windows bash

```
curl -X GET http://192.168.1.2/v1/legal/-1
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/legal/-1" | Select-Object -Expand Content
```

6.4.2. START

In the following examples we set:

evseId	evse458877
transactionId	transac5000
clientId	client12657
tariffId	2
cableId	1
userData	""

Linux/Windows bash

```
curl -d
'{"evseId":"evse458877","transactionId":"transac5000","clientId":"client12657","tariffId":2,"cableId":
1,"userData":""}' -H 'Content-Type: application/json' -X POST http://192.168.1.2/v1/legal
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/legal" -ContentType "application/json" -Method
POST -Body '{
"evseId":"evse458877","transactionId":"transac5000","clientId":"client12657","tariffId":2,"cableId":1
,"userData":"" }' | Select-Object -Expand Content
```

6.4.3. STOP

The transactionId is required to stop the on-going transaction. It can be recovered with a read.

In this example we stop the transaction with *transactionId="transac5000"*

Linux/Windows bash

```
curl -d '{"running":false}' -H 'Content-Type: application/json' -X PUT
http://192.168.1.2/v1/legal?transactionId=transac5000
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/legal?transactionId=transac5000" -ContentType
"application/json" -Method PUT -Body '{"running": false}'  | Select-Object -Expand Content
```

7. /OCMF API

In parallel to the /legal format used to start, read and stop a transaction, the /ocmf API format is also available to :

- read a current transaction

- read the last transaction

- read a past transaction

Following section details the used OCMF format and more details can also be found into the Operation Manual.

OCMF readouts can be verified using official OCMF transparency software. Refer to Operation manual, section “Data authenticity”.

7.1. Overview

The OCMF structure falls into 3 parts:

OCMF|JSON1|JSON2

where:

- OCMF is a fixed header
- JSON1 is the response in JSON format
- JSON2 is the signature in JSON format

For more information, please,see the OCMF specification.

Here is an introduction of reading as an example:

```
OCMF|
{
  "FV": string,      // ocmf v1.0
  "GI": string,      // fixed, identify DCBM version
  "GS": integer,     // Serial number of the DCBM
  "GV": string,      // identify DCBM Rest API version
  "PG": string,      // pagination counter
  "MV": string,      // fixed, identify LEM manufacturer
  "MS": string,      // Serial number of the DCBM
  "MF": string,      // firmware version
  "IS": boolean,     // cf /settings ocmf field
  "IL": string,      // cf /settings ocmf field
  "IF": [            // cf /settings ocmf field
        string,
        ...
       ],
  "IT": string,      // cf /settings ocmf field
  "ID": string,      // transactionId
  "CT": string,      // fixed, label for CI field
  "CI": string,      // evseId
  "TT": string,       // Tarif Information, only for transaction v2 (NxD or NxM)
  "UV": string,       // Custom field, only for transaction v2 (NxD or NxM)
  "UD": string,       // Custom field, only for transaction v2 (NxD or NxM)
  "RD":               // readings
[
    {
      "TM": string,    // timestamp of start + time status
      "TX": string,    // B = begin
      "RV": number,    // reading value
      "RI": string,    // obis code imported energy
      "RU": string,    // unit of reading
      "RT": string,    // DC charger
      "EF": string,    // error flag
      "ST": string,    // status flag, G = good
      "UC": {          // LEM Specific JSON field info on selected cable (compensation applied)
           "UN": string,     // Name of the cable
           "UI": integer,    // Id of the cable  (used for selection on start command)
           "UR": integer     // Resistance value of the cable
         }
    },
    {
      "RV": number,    // reading value
      "RI": string,    // obis code exported energy
      "RU": string,    // unit of reading
      "ST": string     // status flag, G = good
    },
    {
      "TM": string,    // timestamp of stop/read intermediate  + time status
      "TX": string,    // C = charging, E = end
      "RV": number,    // reading value
      "RI": string,    // obis code imported energy
      "RU": string,    // unit of reading
      "EF": string,    // error flag display when different from first reading
      "ST": string     // status flag, G = good
    },
    {
      "RV": number,    // reading value
      "RI": string,    // obis code exported energy
      "RU": string,    // unit of reading
      "ST": string     // status flag, G = good
    }
]
}|
{
"SA": string,               // signature type, cf the OCMF spec
"SD": string                // signature value, done on JSON1 field (string) without spaces
}
```

The maximum size required when reading DCBM data in LEM or OCMF format should not exceed 1.2 kBytes; customers advised to configure their buffer length with this minimum value.

7.2. Fields description

7.2.1. JSON1

7.2.1.1. FV field

FV	String	Format-Version: = "1.0"

7.2.1.2. GI field

GI	String	Gateway identification= "LEM DCBM".

7.2.1.3. GS field

GS	String	Gateway material = DCBM serial number (string of 37 char max)

7.2.1.4. GV field

GV	String	Gateway version = "v1"/"v2", the HTTP REST API version

7.2.1.5. PG field

PG	String	Pagination of the entire dataset = string of "T<value>" with value increased for each start transaction accepted. Pagination counter read in OCMF will be always the same when reading a transaction. DCBM does not increment the Pagination counter when reading multiple time the same Transaction.

7.2.1.6. MV field

MV	String	Meter-Vendor = "LEM"

7.2.1.7. MS field

MS	String	Meter-Serial = DCBM serial number (string of 37 char max)

7.2.1.8. MF field

MF	String	Meter-Firmware: Legal (Metrological) Firmwares parts. For example = "MU-2.3.0.1_SU-0.0.8.0" for DCBM400

7.2.1.9. IS FIELD

⚠ *Value of this field depends on the /settings/ocmfId/IL field value: it has to be set by the charging controller.*

IS	Boolean	Identification status: General status for user assignment: true: Users successfully assigned, false: Users not associated.

Set to true if IL field is set to following values in /settings/ocmfId/IL field, false otherwise :

2	"HEARSAY"	The assignment is unsecured; e.g. by reading out an RFID UID.
3	"TRUSTED"	The assignment can be trusted to some extent, but there is no absolute reliability. Example: Authorization by Backend
4	"VERIFIED"	The assignment has been verified by the signature component and specific actions.
5	"CERTIFIED"	The mapping was verified by the signature component using a cryptographic signature that certifies the mapping.
6	"SECURE"	The assignment was established by a safe feature (e.g. secure RFID card, ISO15118 with plug and charge, etc.)

7.2.1.10. IL FIELD

⚠ *Value of this field depends on the /settings/ocmfId/IL field value: it must be set by the charging controller.*

IL	String	Identification level: JSON Array .

See /settings/ocmfId/IL chapter for corresponding values

7.2.1.11. IF FIELD

⚠ *Value of this field depends on the /settings/ocmfId/IF field value: it must be set by the charging controller.*

IF	Array of String	Identification flags for RFID, OCPP, ISO15118 and PLMN protocol

Set according to settings/ocmfId/IF chapter

7.2.1.12. IT FIELD

⚠ *Value of this field depends on the /settings/ocmfId/IT field value: it must be set by the charging controller.*

IT	String	Identification-Type: "string"

Set as per settings/ocmfId/IT chapter

⚠️ For correct usage the IT fields shall be set once to the corresponding protocol used (or kept to "NONE" as per default value), and not be changed during the lifetime of the DCBM due to current software limitation.

7.2.1.13. ID FIELD

⚠️ *Value of this field depends on the /settings/ocmfId/IT field value: it must be set by the charging controller.*

ID	String	Identification-Data: "string", set according to : - /legal START command - and /settings/ocmfId/IT fields Table below indicates the redirection of ID fields to /legal/transactionId or /legal/clientId

This depends of the settings/ocmfId/IT fields.

0	"NONE"	/legal/transactionId
1	"DENIED"	/legal/transactionId
2	"UNDEFINED"	/legal/transactionId
3	"ISO14443"	/legal/clientId
4	"ISO15693	/legal/clientId
5	"EMAID"	/legal/clientId
6	"EVCCID"	/legal/clientId
7	"EVCOID"	/legal/clientId
8	"ISO7812"	/legal/clientId
9	"CARD_TXN_NR"	/legal/clientId
10	"CENTRAL"	/legal/transactionId
11	"CENTRAL_1"	/legal/transactionId
12	"CENTRAL_2"	/legal/transactionId
13	"LOCAL",	/legal/transactionId
14	"LOCAL_1"	/legal/transactionId
15	"LOCAL_2"	/legal/transactionId
16	"PHONE_NUMBER"	/legal/clientId
17	"KEY_CODE"	/legal/clientId

7.2.1.14. CT FIELD

CT	String	Charge-Point-Identification-Type: "EVSEID"

7.2.1.15. CI FIELD

CI	String	Charge-Point-Identification: string = /legal/evseId value

7.2.1.16. TT FIELD

⚠ *This field will be visible only if you perform a transaction in v2; otherwise, it will remain hidden.*

TT	String	Tarif Information field: • string = /legal/TT value • Max size = 20 char • Authorised characters are : ASCII encoding

7.2.1.17. UV FIELD

⚠ *This field will be visible only if you perform a transaction in v2; otherwise, it will remain hidden.*

UV	String	Custom field: • string = /legal/UV value • Max size = 20 char • Authorised characters are : ASCII encoding

7.2.1.18. UD FIELD

⚠ *This field will be visible only if you perform a transaction in v2; otherwise, it will remain hidden.*

UD	String	Custom field: • string = /legal/UD value • Max size = 20 char • Authorised characters are : ASCII encoding

7.2.1.19. RD FIELDS (READINGS)

7.2.1.19.1. TM FIELD

TM	String	Time: Indication of the system time of reading and synchronization state. = "<localtime>,000<deviation> <time_sync_status_letter>"

with <locatime> = local time (in datetime format) of the DCBM at the time of the reading, ISO8601 extended format

with <deviation> = signed deviation from local time to UTC, ISO8601 extended format

with <time_sync_status_letter> = see table below

letter:	Description:
U	Unknown, unsynchronisiert
I	Informative (Info watch)
S	Synchronized
R	Relative time billing with a quartz timer based on an info watch.

The DCBM uses "R" (INFO time) + XTAL timer level

7.2.1.19.2. TX FIELD

TX	String	Transaction: Reason for reading, reference to the transaction, noted as capital letter: first reading is start of transaction : "B" third reading is stop of transaction : "E" or "C" for intermediate reading

7.2.1.19.3. RV FIELD

RV	Number	Reading Value: the energy register value up to 9 digits with 3 decimals for NxD or NxM, and 4 decimals for N00.

- **1st reading tuple**: imported energy register at start of transaction
- **2nd reading tuple**: exported energy register on stop of transaction
- **3rd reading tuple:** imported energy register at end of transaction / during transaction depending on the context (see TX field)
- **4th reading tuple**: exported energy register on end of transaction / during transaction depending on the context (see TX field)

7.2.1.19.4. RI FIELD

RI	String	Reading Identification : OBIS code

- 1st reading tupple : "1-0:1.8.0" (Total Imported Energy)
- 2nd reading tupple : "1-0:2.8.0" (Total Exported Energy)
- 3rd reading tupple : "1-0:1.8.0" (Total Imported Energy)
- 4th reading tupple : "1-0:2.8.0" (Total Exported Energy)

7.2.1.19.5. RU FIELD

RU	String	Reading unit: = "kWh"

7.2.1.19.6. RT FIELD

RT	String	Reading Current type: = "DC"

7.2.1.19.7. EF FIELD

EF	String	Error flags

Value	Meaning
""	No error
"E"	Error in the energy register
"t"	Error in the time status
"Et"	Error in the energy registers and the time status

7.2.1.19.8. ST FIELD

OCMF field :

ST	String	Status = the letter is set according to status bit (see folowing table)

Abbreviation:	Identifier:	Description:
N	NOT_PRESENT	The counter has not been found existent
G	OK	Counter in order (Good)
T	TIMEOUT	Time-crossing while trying to control the counter
M	MANIPULATED	Manipulation detected
E	OTHER_ERROR	Other, unknown errors
F	READ_ERROR	Meter registers do not read correctly; Value of the reading is not valid

7.2.1.19.9. UC FIELD

This field reflects the /settings/cableConf selected table for the transaction by the /legal/cableId input parameter

This is a LEM specific field, using specific IDs:

UN	string	cable name (max size 9 char)
UI	integer	cable ID (value from 0 to 7)
UR	integer	resistance value of the cable (value from 0 to 255)

7.2.2. JSON2

7.2.2.1. SA FIELD

SA	String	"ECDSA-secp256r1-SHA256"

This field is a fixed value indicating the signature algorithm used and set according to OCMF specification

7.2.2.2. SD FIELD

SD	String	signature performed on JSON1 field , octet string DER format,

Example:

```
{
  "SA": "ECDSA-secp256r1-SHA256",
  "SD":
"3045022100B3EB273433278F102D4E18EC871B575533D4AFC62AC28229FA61428AB74DBA
9602204A98B7517866F82370EEDF170A8CEF17221759146A54FB7A830E7D111C3A30F9"
}
```

7.3. ALLOWED REQUESTS

7.3.1. GET - READ /OCMF

Rest command:

- By latest:

    GET /vX/ocmf HTTP/1.1

- By absolute transaction index:

    GET /vX/ocmf/<index_value> HTTP/1.1

- By chronogical transaction index:

    GET /vX/ocmf/-<index_value> HTTP/1.1

- By *transactionId (limited to latest 839 transactions)*

⚠ *When requesting a past transaction by transactionId, if multiple transactions have the same transactionId, the latest is fetched.*

    GET /vX/ocmf?transactionId=<past_transactionId_value> HTTP/1.1

X = API version (1 for all version and 2 only for NxD, NxM)

7.3.1.1. RESPONSE BODY

See section 7.1. Overview

7.3.1.2. POSSIBLE HTTP STATUS CODES

Code number	Signification
200	Ok

7.4. EXAMPLES

- Get current or last transaction:

Linux/Windows bash

```
curl -X GET http://192.168.1.2/v1/ocmf
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/ocmf"  | Select-Object -Expand Content
```

- Get a past transaction by *transactionId*: (limited to latest 839 transactions)

⚠ *When requesting a past transaction by transactionId, if multiple transactions have the same transactionId, the latest is fetched.*

Linux/Windows bash

```
curl -X GET http://192.168.1.2/v1/ocmf?transactionId=transac5000
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/ocmf?transactionId=transac5000"  | Select-Object
-Expand Content
```

- Get a past transaction by storage index:

Linux/Windows bash

```
curl -X GET http://192.168.1.2/v1/ocmf/0
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/ocmf/0"  | Select-Object -Expand Content
```

- Get a past transaction by chronogical index:

Linux/Windows bash

```
curl -X GET http://192.168.1.2/v1/ocmf/-12
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/ocmf/-12"  | Select-Object -Expand Content
```

8. /LOGBOOK API

Read-only, contains the metrological and non-metrological events during the lifetime of the DCBM.

Non-metrological events are prefixed by EV_APP.

8.1. OVERVIEW

The JSON object is as follows.

The "logbook" field is a JSON array of events.

All fields are provided as strings.

⚠ *When the logbook is full, the DCBM ceases to operate (FF error, event: "EV_LOGBOOK_FULL").* ***The DCBM shall be changed.***

⚠ *The logbook can contain up to* ***39'999*** *events.*

```
{
  "meterId": string,
  "logbook": [
    {
      "timestamp": string,
      "eventCode": string,
      "status": [
        string,
        string,
        ...
      ]
    },
    ...
  ],
  "signature": string
}
```

8.2. FIELDS DESCRIPTION

8.2.1. METERID

Display the meterId value (serial number) of the DCBM. Max size is 37 characters.

Example:

```
"meterId": "12024072805",
```

8.2.2. Logbook

The *logbook* field is a JSON array that contains event tuples. An example is given below.

```
"logbook": [
  {
   "timestamp": "2019-10-28T09:40:07Z",
   "eventCode": "EV_TIME_SYNC_SUCCEEDED",
   "status": [
     "STATUS_SENSOR_LINK",
     "STATUS_TIME_SYNC"
   ]
  },
  ...
],
```

8.2.2.1. TIMESTAMP

UTC timestamp of the event, in ISO8601 extended datetime format

⚠ *Unlike /legal or /ocmf APIs, the timestamp of the logbook is expressed in UTC time ("Z" suffix) and does not depends on local time settings*

Example:

```
"timestamp": "2019-10-28T09:40:07Z",
```

8.2.2.2. EVENTCODE

The eventCode of the event, one of following list.

Some of the events are linked to a fatal error, preventing new transactions permanently. Some are blocking, preventing new transactions temporarily.

Name	Description	Incidence and links
EV_LNR_FIRMWARE_UPDATE_SUCCEEDED	Application firmware update succeeded.	
EV_LNR_FIRMWARE_UPDATE_FAILED	Application firmware update failed.	
EV_TIME_SYNC_FAILED	Time synchronization expired	Is blocking until EV_TIME_SYNC_SUCCEEDED Logged on falling status timeSyncStatusIsOk
EV_TIME_SYNC_SUCCEEDED	Time synchronization succeeded.	Logged on raised status timeSyncStatusIsOk
EV_TAMPERING	Cover was detected open.	Logged on status tamperingIsDetected
EV_LR_FIRMWARE_CHECK_FAILED	Metrology firmware check failed.	Is linked to a fatal error Logged on errors muDataIntegrityIsFailed and muFwIntegrityIsFailed
EV_LNR_FIRMWARE_CHECK_FAILED	Application firmware check failed.	
EV_MEMORY_LOW	Allocated stack overflow.	
EV_MEASUREMENT_CONSISTENCY_FAILING	Measurement error occured, data on the data link was missed.	Is blocking Logged on falling status suLinkStatusIsOk
EV_POWER_SUPPLY_FAILURE	Power fail occurred.	
EV_BATTERY_MONITORING_LOW	Obsolete.	
EV_ECSDA_KEY_CHANGED	ECDSA private/public keyset was changed.	
EV_PAIRING_KEY_CHANGED	SU/MU pairing key was changed.	
EV_MANUF_KEY_CHANGED	CRC on production parameters was changed.	
EV_INVALID_MEMORY_ACCESS	Invalid memory access was detected.	Reboots the DCBM Is linked to a fatal error Logged on error memoryAccessIsFailed
EV_LOGBOOK_FULL	The logbook is full.	Is linked to a fatal error Logged on error logbookIsFull
EV_SU_INVALID_STATE	The Sensor Unit is in an invalid state.	Is linked to a fatal error Logged on error suStateIsInvalid and falling status suLinkStatusIsOk

EV_SU_INTEGRITY_ERROR	Integrity error was detected inside the Sensor Unit.	Is linked to a fatal error Logged on error suIntegrityIsFailed
EV_SU_MEASURE_ERROR	Error occurred in the measurement module of the Sensor Unit.	Logged on status suMeasureFailureOccurred
EV_VERSION_INCONSISTENCY	The Sensor Unit firmware version is not the one the Meter Unit expects.	Is linked to a fatal error Logged on error versionCheckIsFailed
EV_STACK_OVERFLOW	Allocated stack overflew in metrology firmware.	Is linked to a fatal error Logged on error muStateIsFailed
EV_INIT_ERROR	Error occurred during initialization of the DCBM.	Is linked to a fatal error Logged on error muInitIsFailed
EV_TEMPERATURE_OVERHEAT	Detected temperature is outside acceptable range.(>140°)	Logged on raised status overTemperatureIsDetected
EV_TEMPERATURE_NORMAL	Detected temperature reached back the acceptable range.(<135°)	Logged on falling status overTemperatureIsDetected
EV_RNG_INIT_TEST_FAIL	Random Number Generation initialization test has failed.	Is linked to a fatal error Logged on error muRngInitIsFailed
EV_COMMISSIONING	Commissioning event of the DCBM.	
EV_EXTERNAL_MEMORY_INTEGRITY	The logbook internal CRC check is wrong.	Is linked to a fatal error Logged on error logbookIntegrityIsFailed
EV_LNR_FIRMWARE_UPDATE_ATTEMPT	Application firmware update was initiated.	
EV_RTC_FALLBACK	RTC fallback at startup.	
EV_APP_NO_EVENT	No event.	
EV_APP_DHCP_CONFIGURATION_CHANGED	DHCP configuration changed and DHCP is activated.	
EV_APP_IP_ADDRESS_CHANGED	static IP address changed.	
EV_APP_TIME_SERVER_CONFIGURATION_CHANGED	SNTP configuration changed.	

Example:

```
"eventCode": "EV_TIME_SYNC_SUCCEEDED",
```

8.2.2.3. STATUS

"status" field is a copy of the status bit at the time of the event. The status name are displayed when the corresponding status bits are set. Refer to /status API description.

Example:

```
"status": [
 "STATUS_SENSOR_LINK",
 "STATUS_TIME_SYNC"
]
```

8.2.3. SIGNATURE

Signature of the logbook in octet string format, with ASN1 DER encoding, using ECDSA secp256r1 and SHA256 methods.

Example:

```
  "signature":
"304502205C7B5B67C012E2691738B4CE5365AEE1191D0F59AAB81D6C0C0C1BC74303FDB
9022100A79E1BBA77EA6B110E19C81D84D44750C0361A04E5662783D13D5F1BFDEF66D7"
```

8.3. ALLOWED REQUESTS

8.3.1. GET - READ /LOGBOOK

REST command :

```
GET /vX/logbook HTTP/1.1
```

X = API version (1 for all version and 2 only for NxD, NxM)

8.3.1.1. RESPONSE BODY

Seesection 8.1. Overview

8.3.1.2. POSSIBLE HTTP STATUS CODES

Code number	Meaning
200	Ok

8.4. EXAMPLES

Linux/Windows bash

```
curl -X GET http://192.168.1.2/v1/logbook
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/logbook" | Select-Object -Expand Content
```

9. /LIVEMEASURE API

9.1. OVERVIEW

All */livemeasure* fields are read-only (GET method).

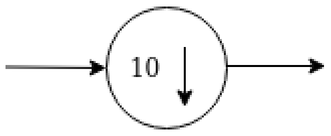
```
{
  "voltage": number,
  "current": number,
  "power": number,
  "temperatureH": number,
  "temperatureL": number,
  "energyImportTotal": number,
  "energyExportTotal": number,
  "timestamp" : string
}
```

9.2. FIELDS DESCRIPTION

Type: Fields are **read-only**

⚠ *All measurements displayed on /livemeasure field are updated every second.*

This correspond to a downsampling of the DC measurements from 10 Hz to 1 Hz:



⚠ *There is no anti-aliasing low-pass filtering or average.*

9.2.1. VOLTAGE

Voltage measurement, in Volt unit, with 3 decimal digits. Can be negative (with status reversedVoltage under -50 V).

Example:

```
"voltage": 1150.079,
```

9.2.2. CURRENT

Current measurement, in Ampere unit, with 3 decimal digits. Can be negative.

Example:

"current": 461.152,

9.2.3. POWER

Power measurement, in kiloWatt unit, with 3 decimal digits. Can be negative.

This is a computation of ABS(Voltage) * Current fields.

Example:

"power": 530.361,

9.2.4. TEMPERATUREH

Temperature measurement on side “I1” of the Sensor Unit, in celcius degrees unit, with 1 decimal digit.

Example:

"temperatureH": 50.7,

⚠ *In the maintenance screens, value is labelled “T°2”.*

9.2.5. TEMPERATUREL

Temperature measurement on side “I2” of the Sensor Unit, in celcius degrees unit, with 1 decimal digit.

Example:

"temperatureL": 52.6,

⚠ *In the maintenance screens, value is labelled “T°1”.*

9.2.6. ENERGYIMPORTTOTAL

Total energy register of DC imported Energy, in kiloWatt Hour (kWh) unit (OBIS = 1-0:1.8.0) up to 9 digits and 3 decimals for NxD or NxM, 4 decimals for N00.

Example:

"energyImportTotal": 36.739,

9.2.7. ENERGYEXPORTTOTAL

Total energy register of DC exported energy, in kiloWatt Hour (kWh) unit (OBIS = 1-0:2.8.0) with 9 digits and 3 decimals for NxD or NxM, 4 decimals for N00.

Example:

```
"energyExportTotal": 0.000
```

9.2.8. TIMESTAMP

Timestamp of the current livemeasure data set, in UTC time in ISO8601 extended dateformat (ending with the "Z" suffix).

Example:

```
"timestamp": "2020-10-12T09:55:13Z"
```

9.3. ALLOWED REQUESTS

9.3.1. GET - READ /LIVEMEASURE

Request:

```
GET /vX/livemeasure HTTP/1.1
```

X = API version (1 for all version and 2 only for NxD, NxM)

9.3.1.1. RESPONSE BODY

See section 9.1. Overview

9.3.1.2. POSSIBLE HTTP STATUS CODES

Code number	Meaning
200	Ok

9.4. EXAMPLES

Linux/Windows bash

```
curl -X GET http://192.168.1.2/v1/livemeasure
```

Windows (PowerShell)

```
Invoke-WebRequest -uri "http://192.168.1.2/v1/livemeasure" | Select-Object -Expand Content
```