EXECUTIVE SUMMARY

This document is owned by the Technical Working Group (TWG) of the EnOcean Alliance. It is maintained and will be progressed within the authority of the Chairman of the TWG.

Following approval, this specification is now in the status RELEASED.

Changes to this document have to be proposed to the TWG for decision.

Submitted to the TWG: 31st August 2020
Approved by TWG for release: 28th September 2020
Approved by BoD for release: 28th October 2020
## System Specification

<table>
<thead>
<tr>
<th>Ver.</th>
<th>Editor</th>
<th>Change</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>TM</td>
<td>First Draft, based on DC input</td>
<td>Mar 18, 2016</td>
</tr>
<tr>
<td>0.2</td>
<td>TM</td>
<td>Second Draft after EnOcean Feedback</td>
<td>Mar 22, 2016</td>
</tr>
<tr>
<td>0.3</td>
<td>TM</td>
<td>DC Review Points added</td>
<td>May 09, 2016</td>
</tr>
<tr>
<td>0.4</td>
<td>TM</td>
<td>New format for the JSON Object table format</td>
<td>May 30, 2016</td>
</tr>
<tr>
<td>0.5</td>
<td>TM</td>
<td>Feedback integrated</td>
<td>Jun 7, 2016</td>
</tr>
<tr>
<td>0.6</td>
<td>LC</td>
<td>Editorial changes</td>
<td>June 22, 2016</td>
</tr>
<tr>
<td>0.7</td>
<td>TM</td>
<td>Status errors code added based on DC input</td>
<td>June 24, 2016</td>
</tr>
<tr>
<td>0.8</td>
<td>ToH</td>
<td>Editorial changes and cleanup</td>
<td>December 2016</td>
</tr>
<tr>
<td>0.9</td>
<td>ToH</td>
<td>More cleanup</td>
<td>December 2016</td>
</tr>
<tr>
<td>1.0</td>
<td>NM</td>
<td>Editorial modifications following approval by TWG</td>
<td>Mar 09, 2017</td>
</tr>
<tr>
<td>2.0</td>
<td>AP</td>
<td>Moved REST API to extra document</td>
<td>Aug 21, 2020</td>
</tr>
<tr>
<td>2.1</td>
<td>AP</td>
<td>Moved json objects from EoIP specification</td>
<td>Mar 18, 2021</td>
</tr>
</tbody>
</table>

Copyright © EnOcean Alliance Inc. 2012-2021. All rights Reserved.

**Disclaimer**

This information within this document is the property of the EnOcean Alliance and its use and disclosure are restricted. Elements of the EnOcean Alliance specifications may also be subject to third party intellectual property rights, including without limitation, patent, copyright or trademark rights (such a third party may or may not be a member of the EnOcean Alliance.)

The EnOcean Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights. This document and the information contained herein are provided on an “as is” basis and the EnOcean Alliance disclaims all warranties express or implied, including but not limited to (1) any warranty that the use of the information herein will not infringe any rights of third parties (including any intellectual property rights, patent, copyright or trademark rights, or (2) any implied warranties of merchantability, fitness for a particular purpose, title or non-infringement.

In no event will the EnOcean Alliance be liable for any loss of profits, loss of business, loss of use of data, interruption of business, or for any other direct, indirect, special or exemplary, incidental, punitive or consequential damages of any kind, in contract or in tort, in connection with this document or the information contained herein, even if advised of the possibility of
such loss or damage. All Company, brand and product names may be trademarks that are the
sole property of their respective owners.
The above notice and this paragraph must be included on all copies of this document that are
made.

The EnOcean Alliance “EnOcean over IP Specification” is available free of charge to companies,
individuals and institutions for all non-commercial purposes (including educational research,
technical evaluation and development of non-commercial tools or documentation.)

This specification includes intellectual property („IPR“) of the EnOcean Alliance and joint intel-
lectual properties („joint IPR“) with contributing member companies. No part of this specifica-
tion may be used in development of a product or service for sale without being a participant or
promoter member of the EnOcean Alliance and/or joint owner of the appropriate joint IPR.

These errata may not have been subjected to an Intellectual Property review, and as such, may
contain undeclared Necessary Claims.

EnOcean Alliance Inc.
5000 Executive Parkway, Suite 302
San Ramon, CA 94583
USA
Graham Martin
Chairman & CEO EnOcean Alliance
# Table of Contents

1. **Introduction** ................................................................................................................................. 5  
   1.1. Scope and Purpose ......................................................................................................................... 5  
   1.2. Definitions ...................................................................................................................................... 6  
   1.3. Conformance Levels ....................................................................................................................... 6  
   1.4. Documents ...................................................................................................................................... 7  
   1.5. References ....................................................................................................................................... 7  
       1.5.1. *EnOcean Alliance* ................................................................................................................... 7  
       1.5.2. *Internet Engineering Task Force Documents* ................................................................. 7  
       1.5.3. *Others* .................................................................................................................................... 7  

2. **EnOcean Over IP** ............................................................................................................................. 8  
   2.1. Motivation ....................................................................................................................................... 8  
   2.2. IP Representation ............................................................................................................................ 8  
   2.3. Message flow ................................................................................................................................... 9  
   2.4. EEP IP Representations ............................................................................................................... 10  
   2.5. Transport protocols ...................................................................................................................... 11  
   2.6. Available JSON objects ................................................................................................................. 11  
       2.6.1. *systemInfo object* .................................................................................................................. 11  
       2.6.2. *profiles array object* ............................................................................................................ 12  
       2.6.3. *profile object* .......................................................................................................................... 13  
       2.6.4. *devices array object* .............................................................................................................. 17  
       2.6.5. *device object* .......................................................................................................................... 18  
       2.6.6. *state / states object* ................................................................................................................ 19  
       2.6.7. *telegram / telegrams object* ..................................................................................................... 20
1. Introduction

1.1. Scope and Purpose

This document is intended for consumers and producers of EAGs (EnOcean Alliance IP Gateways). For manufacturers of an EAG, this document specifies how different profiles and EnOcean technologies are mapped to IP based communication independent of the transfer protocol.

Chapter 1 mentions the general scope, acronyms, definitions contributors, Chapter 2 describes the

Example of possible use cases of the EAG is shown in Figure 1. It shows a smartphone app which is directly connected to an EAG. The smartphone displays the current status of devices and send messages to devices via the EAG.

![Figure 1: Example of possible use cases](image-url)
1.2. Definitions

API – Application Programming Interface

BaseID - is a base Address which may be used to send telegrams to different actuators using different source IDs. It is possible to transmit a telegram with the base ID of an EnOcean device as the source address and any ID up to the base ID + 127.

EAG – EnOcean Alliance IP Gateway

EEP - EnOcean Equipment Profile; Specification to define the structure of over-the-air data bytes for EnOcean transmitters. Also see Generic Profiles.

EURID – EnOcean Unique Radio Identifier, a unique and non-changeable identification number assigned every EnOcean transmitter during its production process.

GP - Generic Profiles; Specification to define the structure of over-the-air data bytes for EnOcean transmitters. Also see EEP.

HTTP – Hypertext Transfer Protocol

IoT – Internet Of Things

JSON - Java Script Object Notation

REST - Representational State Transfer

1.3. Conformance Levels

MUST - This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

MUST NOT - This phrase, or the phrase "SHALL NOT", means that something is prohibited by the specification.

SHOULD - This word, or the word "RECOMMENDED", means that, in particular cases, there may be valid reasons not to follow a point of the specification. However the full implications must be understood and weighed before choosing a different course.

SHOULD NOT - This phrase, or the phrase "NOT RECOMMENDED" means that, in particular cases, there may be valid reasons where certain behavior is acceptable or even useful. However the full implications must be understood and weighed before choosing to implement anything described with this phrase.

MAY - This word, or the adjective "OPTIONAL", means that an item may or may not be implemented. One vendor may choose to include the item because a particular marketplace
requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. In either case, both MUST be prepared to interoperate with each other, though perhaps with reduced functionality.

### 1.4. Documents

The EnOcean over IP Specification consists of a main document (this document) and several descriptions of example implementations with different transport protocols. You find them in the technical specifications section of the EnOcean Alliance homepage.

The EEP Viewer tool provides the IP representation description of the various EEPs.

### 1.5. References

#### 1.5.1. EnOcean Alliance

[E1] EEP (EnOcean Equipment Profiles) Specification
https://www.enocean-alliance.org/EEP/

[E2] EnOcean GP(Generic Profiles)
https://www.enocean-alliance.org/gp/

https://www.enocean-alliance.org/reman/


[E5] EEP Viewer
http://tools.enocean-alliance.org/EEPViewer/

[E6] EnOcean over IP REST API Implementation example specification
https://www.enocean-alliance.org/specifications/

[E7] EnOcean over IP MQTT Implementation example specification
https://www.enocean-alliance.org/specifications/

#### 1.5.2. Internet Engineering Task Force Documents

[RFC1] RFC 7159: The JavaScript Object Notation (JSON) Data Interchange Format


#### 1.5.3. Others

[O1] JSON website
http://www.json.org/
2. EnOcean Over IP

2.1. Motivation

With the emergence of IoT technology it is necessary for EnOcean devices become a part of the IoT world. Due to the limitation of Energy Harvesting Devices and the EnOcean protocol, it is not possible to integrate the devices directly into the IP world with e.g. a 6LoWPAN Adaption Layer. This specification suggests and specifies how data transmitted to or from EnOcean radio devices has to be represented to allow for an easy integration of an EnOcean ECO-System into an IP Network, Intranet or Internet. Another main motivation is to hide the complexity of the EnOcean Radio and the different Application Protocols (e.g. EEP/GP/ReCom) from the user of the EAG. Currently the IP Representation is based only on EEPs, no GP/ReCom.

2.2. IP Representation

The EEP payload data which is transmitted from or to an EnOcean radio device contains parameters. These parameters can be the measured values, set points, error messages, states, flags or any other type of data needed in the application. In the IP Representation such a parameter is named “key”. Each EEP has different keys depending on the data to transfer.

Examples for keys are: “temperature”, “humidity”, “contact”, “rampingTime”, “basicSetpoint”, “powerFailureDetected”, “switch”, “errorCode”.

Each key has a “value”. For simple measurements it is the corresponding measurement value in numbers. For flags or enum keys it’s the represented enumeration name, or just “true” or “false”. Sometimes error messages are also coded in the keys, like overflow errors or some kind of HW detected failures. The corresponding value can then consist of either the measurement value, or the error enumeration name.

Examples for values are: “23.5”, “7”, “false”, “measurementsTooLow”.

Each key/value pair has a “meaning”. It is a textual description of the content of the key/value pair, this pairs are designed to be “human readable”, in order to avoid, that new keys with same meaning to be created while submitting a new EEP.

Key names are radio agnostic, there is no EnOcean radio specific key name used. Just the combinations of keys used are referring to an EEP.

The IP representation is organized in function groups. There is no IP representation for Teach-Telegrams or signals.
2.3. Message flow

Temperature & Humidity sensor

EAG decodes EnOcean Telegram into KEY’s and VALUE’s

Then sends it with several possible transmit protocols (HTTP, MQTT, AMQP, …) to cloud services

{sensor originator info}
KEY = temperature
VALUE = 22
KEY = humidity
VALUE = 48

Figure 2: Message flow
Figure 2 shows the message flow.

The radio telegram coming from an EnOcean device is usually coded with an EEP. The EAG decodes the EEP and separates all parameters into key/value pairs.

Depending from the used transport protocol a set of key/value pairs, of course with data to identify the source device, are transmitted to the destination.

2.4. EEP IP Representations

To use the IP representations of EEPs the EEP Viewer Tool [E5] provides access to all defined IP representations.

In the responding IP-Representation file are the key/value/meanings of the EEP listed.
2.5. Transport protocols

The definition of the key/value pairs is completely independent from the transport protocol used to transmit the data from/to the application, which uses or generates the data.

The EAG can be implemented using a REST-API, MQTT, AMQP, or any future transport protocol. An essential component of EAG-compliant gateways is the usage of the sets of key/value pairs defined in the EEPs. This specification purposely abstains from describing the way data is encoded or which EEPs should be supported. EAG-manufacturers are free to transmit any other additional information they consider necessary.

It is intentionally allowed to use the key/value sets with other radio protocols in parallel to the EnOcean radio. This enables the EAG to support different radio standards.

2.6. Available JSON objects

This chapter lists all available JSON objects used in existing example implementation specifications.

2.6.1. systemInfo object

The systemInfo object is used to represent state of the system and display information about the EAG.

Used by following Resources:

- [E6] GET /system/info

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value/Formatting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>string</td>
<td></td>
<td>contains the current Software version, date and time of Interface</td>
</tr>
<tr>
<td>baseId</td>
<td>string</td>
<td>4 byte hex value</td>
<td>current BaseID of Interface</td>
</tr>
<tr>
<td>possibleBaseIdChanges</td>
<td>integer</td>
<td>number (10..0)</td>
<td>remaining number of BaseID changes (maximum 10 times, according to specifications of the chip)</td>
</tr>
<tr>
<td>eurid</td>
<td>string</td>
<td>4 byte hex value</td>
<td>EURID of the built-in EnOcean Device</td>
</tr>
<tr>
<td>frequency</td>
<td>integer</td>
<td>868, 902, 928</td>
<td>Used frequency of the Interface</td>
</tr>
</tbody>
</table>

Table 1: systemInfo Object
JSON for systemInfo object

```
"systemInfo" : {
    "version" : "Interface v0.9.1 2016.03.15 14:43",
    "baseId" : "FFA04700",
    "possibleBaseIdChanges" : 10,
    "eurId" : "0185408E",
    "frequency" : 868
}
```

Figure 3: systeminfo JSON example

2.6.2. profiles array object

The profiles array object is used to represent all supported profiles by the EAG.

Used by following Resources:

- [E6] GET /profiles

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value/Formatting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eep</td>
<td>String</td>
<td>xx-xx-xx</td>
<td>EnOcean Equipment Profiles, definition of EnOcean radio telegram structure</td>
</tr>
<tr>
<td>title</td>
<td>String</td>
<td>May be translated</td>
<td>short description of eep profile</td>
</tr>
<tr>
<td>variations</td>
<td>Array of objects</td>
<td>[[]]</td>
<td>Information about available versions of the profile</td>
</tr>
</tbody>
</table>

Table 2: profiles array object

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value/Formatting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>direction</td>
<td>String</td>
<td>from</td>
<td>to</td>
</tr>
<tr>
<td>version</td>
<td>Float</td>
<td>x.x</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: variations object
2.6.3. profile object

The profile object is used to represent a full description of an EEP. It can be used to query all information about a used EEP.

Used by following Resources:

- [E6] GET /profiles/{EEP}
- [E6] GET /device/profile

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value/Formatting</th>
<th>Description</th>
<th>Opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>eep</td>
<td>string</td>
<td>xx-xx-xx</td>
<td>EnOcean Equipment Profiles, definition of EnOcean radio telegram structure. not shown in /device/profile</td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>string</td>
<td></td>
<td>description of eep profile not shown in /device/profile</td>
<td></td>
</tr>
<tr>
<td>functionGroups</td>
<td>array of objects</td>
<td>[{}],</td>
<td>array of functionGroup objects</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: profile object

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value/Formatting</th>
<th>Description</th>
<th>Opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>String</td>
<td></td>
<td>Title of the group</td>
<td></td>
</tr>
<tr>
<td>direction</td>
<td>string</td>
<td>from</td>
<td>to</td>
<td>direction of transport (from, to device or both)</td>
</tr>
<tr>
<td>functions</td>
<td>array of objects</td>
<td>[{}],</td>
<td>array of function objects</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: functionGroups object
### Property | Type | Value/Formatting | Description | Opt
--- | --- | --- | --- | ---
key | String | | key value, as a function identifier | 

description | String | Description of the key, may be translated to native client language. | description of function | opt

values | array of objects | [[]] | array of value objects | 

defaultValue | String | In case of direction “to”: Default value which will be send if no value has been set | opt | 

**Table 6: functions object**

### Property | Type | Value/Formatting | Description | Opt
--- | --- | --- | --- | ---
value | String | | Value (if not present, range needs to be present!) | opt

meaning | String | Description of the value, may be translated to native client language. | meaning of value | opt

range | Object | {} | range object (if not present, value needs to be present!) | opt | 

**Table 7: values object**

### Property | Type | Value/Formatting | Description | Opt
--- | --- | --- | --- | ---
min | Float | | Representation of the valid range of values with increment and, where appropriate, the corresponding unit | 

max | Float | | 

step | Float | | 

unit | String | [unit] | | opt

**Table 8: range object**
JSON for profile object

```
"profile": {
  "eep": "A5-02-01",
  "title": "Temperature Sensors, Temperature Sensor Range -40°C to 0°C",
  "functionGroups": [{
    "direction": "from",
    "functions": [{
      "key": "temperature",
      "description": "Temperature (linear)",
      "values": [{
        "range": {
          "min": -40,
          "max": 0,
          "step": 0.157,
          "unit": "°C"
        }
      }]
    }]
  }]
}
```

Figure 5: profile A5-02-01 JSON example

---

```
"profile": {
  "eep": "D2-01-0A",
  "title": "Electronic switches and dimmers with Energy Measurement and Local Control",
  "functionGroups": [{
    "title": "Actuator Set Output",
    "direction": "to",
    "functions": [{
      "key": "switch",
      "values": [{
        "value": "off",
        "meaning": "Output value OFF"
      }, {
        "value": "on",
        "meaning": "Output value ON"
      }]
    }],
    "title": "Configure Actuator",
    "direction": "to",
    "functions": [{
      "key": "defaultState",
      "values": [{
        "value": "off",
        "meaning": "Default state: OFF"
      }, {
        "value": "on",
        "meaning": "Default state: ON"
      }, {
        "value": "previousState",
        "meaning": "Default state: remember previous state"
      }],
      "defaultValue": "previousState"
    }, {
      "key": "localControl",
      "values": [{
        "value": "off",
        "meaning": "Disable local control"
      }, {
        "value": "on",
        "meaning": "Enable local control"
      }]
    }]
  }
}
```


```json

} ],
  "defaultValue" : "on"
],
"key" : "powerFailureDetection",
"values" : [ {  
  "value" : "off",
  "meaning" : "Disable Power Failure Detection"
],
  "value" : "on",
  "meaning" : "Enable Power Failure Detection"
},
"defaultValue" : "on"
],
"key" : "taughtInDevices",
"values" : [ {  
  "value" : "off",
  "meaning" : "Disable taught-in devices (with different EEP)"
],
  "value" : "on",
  "meaning" : "Enable taught-in devices (with different EEP)"
},
"defaultValue" : "on"
],
"key" : "userInterfaceIndication",
"values" : [ {  
  "value" : "day",
  "meaning" : "User interface indication: day operation"
}, {  
  "value" : "night",
  "meaning" : "User interface indication: night operation"
},
"defaultValue" : "day"
}
],
"title" : "Actuator Status Response",
"direction" : "from",
"functions" : [ {  
  "key" : "localControl",
  "values" : [ {  
    "value" : "off",
    "meaning" : "Local control disabled / not supported"
}, {  
    "value" : "on",
    "meaning" : "Local control enabled"
} ]
}, {  
  "key" : "powerFailureDetected",
  "values" : [ {  
    "value" : "false",
    "meaning" : "Power Failure not detected/not supported/disabled"
}, {  
    "value" : "true",
    "meaning" : "Power Failure Detected"
} ]
}, {  
  "key" : "powerFailureDetection",
  "values" : [ {  
    "value" : "off",
    "meaning" : "Power Failure Detection disabled/not supported"
}, {  
    "value" : "on",
    "meaning" : "Power Failure Detection enabled"
} ]
},
"key" : "switch",
"values" : [ {  
    "value" : "off",
    "meaning" : "Output value OFF"
} ]
```


2.6.4. devices array object

The devices array object is used to represent an array of all learned in devices to the EAG.

Used by following Resources:

- [E6] GET /devices

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value/Formatting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deviceId</td>
<td>String</td>
<td>4 byte hex value (8 characters with leading zeros)</td>
<td>SenderID of EnOcean device, EnOcean modules usually send telegrams with their unique 32-bit Chip ID. In the other case, these are the Base ID of the EnOcean device.</td>
</tr>
<tr>
<td>friendlyId</td>
<td>String</td>
<td></td>
<td>user-assigned name of EnOcean device</td>
</tr>
<tr>
<td>physicalDevice</td>
<td>String</td>
<td></td>
<td>group name of actuators and sensors of the same physical device</td>
</tr>
</tbody>
</table>

Table 9: devices object

JSON for devices array object

```json
"devices" : [ {
    "deviceId" : "FEFFFF60",
    "friendlyId" : "BathroomSwitch"
}, {
    "deviceId" : "01843197",
    "friendlyId" : "OutDoorToiletTemperature"
}, {
    "deviceId" : "00435678",
    "friendlyId" : "ButtonSwitch",
    "physicalDevice" : "MultiDevice"
}, {
    "deviceId" : "00435679",
```
2.6.5. device object

The device object is used to represent all information about a specific device.

Used by following Resources:

- [E6] GET /devices
- [E7] EnOcean/[EAG-Identifier]/stream/devices/{Device-Identifier}
- [E7] EnOcean/[EAG-Identifier]/getAnswer/devices/{Device-Identifier}
- [E7] EnOcean/[EAG-Identifier]/post/devices/{Device-Identifier}

Table 10: device Object

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value/Formatting</th>
<th>Description</th>
<th>Opt</th>
</tr>
</thead>
<tbody>
<tr>
<td>deviceId</td>
<td>string</td>
<td>4 byte hex value</td>
<td>SenderID of EnOcean device, EnOcean modules usually send telegram with their unique 32-bit Chip ID. In the other case, these are the Base ID of the EnOcean device.</td>
<td></td>
</tr>
<tr>
<td>friendlyId</td>
<td>string</td>
<td></td>
<td>user-assigned name of EnOcean device</td>
<td></td>
</tr>
<tr>
<td>physicalDevice</td>
<td>string</td>
<td></td>
<td>group name of actuators and sensors of the same physical device</td>
<td>opt</td>
</tr>
<tr>
<td>eeps</td>
<td>array</td>
<td>[]</td>
<td>EEP supported by the device</td>
<td></td>
</tr>
<tr>
<td>manufacturer</td>
<td>string</td>
<td></td>
<td>Based on manufacturer ID if send</td>
<td>opt</td>
</tr>
<tr>
<td>location</td>
<td>string</td>
<td></td>
<td>Location of the device</td>
<td>opt</td>
</tr>
<tr>
<td>dbm</td>
<td>integer</td>
<td></td>
<td>dBm of last received telegram</td>
<td>opt</td>
</tr>
<tr>
<td>firstSeen</td>
<td>DateTime</td>
<td>yyyy-mm-ddT hh:mm:ss.sss+hhmm</td>
<td>timestamp on which the device has been detected for the first time</td>
<td></td>
</tr>
<tr>
<td>lastSeen</td>
<td>DateTime</td>
<td>yyyy-mm-ddT hh:mm:ss.sss+hhmm</td>
<td>timestamp on which the device has been seen last time</td>
<td>opt</td>
</tr>
</tbody>
</table>

Table 10: device Object

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value/Formatting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eep</td>
<td>String</td>
<td>XX-XX.XX</td>
<td>EnOcean Equipment Profiles, definition of EnOcean radio telegram structure</td>
</tr>
<tr>
<td>version</td>
<td>string</td>
<td>x.x</td>
<td>API Version of the profile</td>
</tr>
</tbody>
</table>
2.6.6. state / states object

The state object is used to represent the current/last known state of a device.

Used by following Resources:

- [E6] GET /devices/states
- [E6] GET /devices/{id}/state
- [E6] GET /devices/stream
- [E6] GET /devices//{id}/stream

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value/Formatting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deviceId</td>
<td>string</td>
<td>4 byte hex value</td>
<td>SenderID of EnOcean device, EnOcean modules usually send telegrams with their unique 32-bit Chip ID. In the other case, these are the Base ID of the EnOcean device.</td>
</tr>
</tbody>
</table>
friendIId | string | user-assigned name of EnOcean device
physicalDevice | string | group name of actuators and sensors of the same physical device
functions | array of objects | array of function objects

### Table 12: state / states Object

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value/Formatting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>string</td>
<td></td>
<td>Group of state functions, consisting of a key value and its last saved value, the receiving timestamp and and the appropriate age in milliseconds of the last message that contains the value. Additional optional parameters are represented by: meaning (conceptual sense of a value), channel (the addressed channel) and the associated unit.</td>
</tr>
<tr>
<td>channel</td>
<td>integer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>float</td>
<td>[unit]</td>
<td></td>
</tr>
<tr>
<td>unit</td>
<td>string</td>
<td>[unit]</td>
<td></td>
</tr>
<tr>
<td>meaning</td>
<td>string</td>
<td></td>
<td>Description of the value, may be translated to native client language.</td>
</tr>
<tr>
<td>timestamp</td>
<td>DateTime</td>
<td>yyyy-mm-ddT hh:mm:ss+hhmm</td>
<td></td>
</tr>
<tr>
<td>age</td>
<td>integer</td>
<td>In ms</td>
<td></td>
</tr>
</tbody>
</table>

### Table 13: functions array object

```json
"state": {
    "deviceId": "01910188",
    "friendlyId": "Opus-Bridge-2K",
    "functions": [
        { "key": "switch", "channel": 1, "value": "on", "meaning": "Output value ON", "timestamp": "2016-11-30T13:39:32.250+0100", "age": 411515 }
    ,
        { "key": "switch", "channel": 0, "value": "off", "meaning": "Output value OFF", "timestamp": "2016-11-30T12:51:33.217+0100", "age": 3290548 }
    ]
}
```

**Figure 9: state JSON example**

#### 2.6.7. telegram / telegrams object

The states object is used in streaming rest calls to represent updates of devices.
Used by following Resources:

- [E6] GET /devices/stream
- [E6] GET /devices/(id)/stream
- [E6] GET /devices/telegrams
- [E6] GET /devices/(id)/telegrams
- [E7] EnOcean/[EAG-Identifier]/stream/telegram/[Device-Identifier]/from
- [E7] EnOcean/[EAG-Identifier]/stream/telegram/[Device-Identifier]/from/{key}
- [E7] EnOcean/[EAG-Identifier]/stream/telegram/[Device-Identifier]/to
- [E7] EnOcean/[EAG-Identifier]/put/devices/[Device-Identifier]/state

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value/Formatting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deviceId</td>
<td>4 byte hex</td>
<td>xxxxxxxx</td>
<td>SenderID of EnOcean device, EnOcean modules usually send telegrams with their unique 32-bit Chip ID. In the other case, these are the Base ID of the EnOcean device.</td>
</tr>
<tr>
<td>friendlyId</td>
<td>string</td>
<td></td>
<td>user-assigned name of EnOcean device</td>
</tr>
<tr>
<td>physicalDevice</td>
<td>string</td>
<td></td>
<td>group name of actuators and sensors of the same physical device</td>
</tr>
<tr>
<td>timestamp</td>
<td>DateTime</td>
<td>yyyy-mm-ddT hh:mm:ss.sss+ hhmm</td>
<td>telegram transmission time in UTC format</td>
</tr>
<tr>
<td>direction</td>
<td>string</td>
<td>from</td>
<td>to</td>
</tr>
<tr>
<td>functions</td>
<td>array of objects</td>
<td>[[]]</td>
<td></td>
</tr>
<tr>
<td>telegramInfo</td>
<td>object</td>
<td>()</td>
<td>telegramInfo object</td>
</tr>
</tbody>
</table>

**Table 14: telegram object**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value/Formatting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>string</td>
<td></td>
<td>Group of state functions, consisting of a key value and its last saved value, the receiving timestamp and the appropriate age in milliseconds of the last message that contains the value. Additional optional parameters are represented by: meaning (conceptual sense of a value), channel (the addressed channel) and the associated unit.</td>
</tr>
<tr>
<td>channel</td>
<td>integer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>float</td>
<td></td>
<td></td>
</tr>
<tr>
<td>unit</td>
<td>string</td>
<td>[unit]</td>
<td></td>
</tr>
<tr>
<td>meaning</td>
<td>string</td>
<td></td>
<td>Description of the value, may be translated to native client language.</td>
</tr>
<tr>
<td>timestamp</td>
<td>DateTime</td>
<td>yyyy-mm-ddT hh:mm:ss.sss+ hhmm</td>
<td></td>
</tr>
</tbody>
</table>
### Table 15: functions array object

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>value/formatting</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>string</td>
<td>x byte hex value</td>
<td>Payload of ERP telegrams or ESP packets</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td></td>
<td>identifies if the subtelegram is transmitted from a repeater and the type of integrity control mechanism used (note: not present in a switch telegram)</td>
</tr>
<tr>
<td>dbm</td>
<td>integer</td>
<td></td>
<td>signal strength of received / transmitted telegram</td>
</tr>
<tr>
<td>rorg</td>
<td>string</td>
<td>1 byte hex value</td>
<td>identifier for subtelegram type</td>
</tr>
</tbody>
</table>

### Table 16: telegramInfo object

#### JSON for telegram Object

```
"telegram" : {  
  "deviceId" : "01843197",  
  "friendlyId" : "temp",  
  "physicalDevice" : "testdevice",  
  "timestamp" : "2016-03-16T18:16:16.134+0100",  
  "direction" : "from",  
  "functions" : [  
    {  
      "key" : "humidity",  
      "value" : "0",  
      "unit" : "%"  
    },  
    {  
      "key" : "temperature",  
      "value" : "0",  
      "unit" : "°C"  
    }  
  ],  
  "telegramInfo" : {  
    "data" : "0000000A",  
    "status" : "0",  
    "dbm" : -65,  
    "rorg" : "A5"  
  }  
}
```

**Figure 10: telegram JSON example**