System Specification

EnOcean over IP Specification

Main document

V 2.1

San Ramon, CA, USA
Original Release 1.0, 2017

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.

REVISION HISTORY

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

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 intellectual properties („joint IPR") with contributing member companies. No part of this specification 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.
2603 Camino Ramon, Suite 200

San Ramon, CA 94583
USA
Graham Martin
Chairman & CEO EnOcean Alliance

Table of Contents

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.

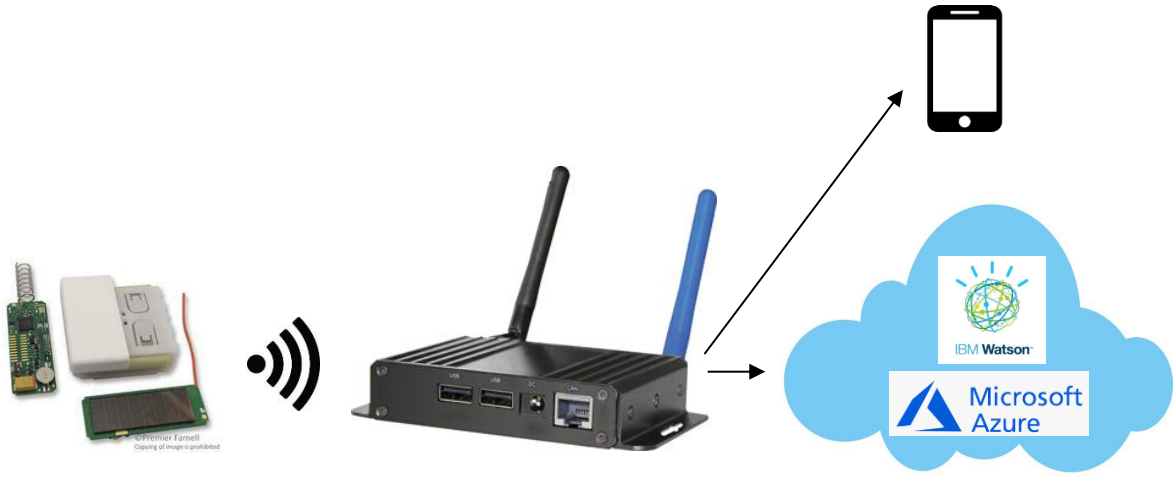

Figure 1: Example of possible use cases

1.2. Definitions

API – **A**pplication **P**rogramming **I**nterface

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 - **E**nOcean **E**quipment **P**rofile; Specification to define the structure of over-the-air data bytes for EnOcean transmitters. Also see Generic Profiles.

EURID – **E**nOcean **U**nique **R**adio **Id**entifier, a unique and non-changeable identification number assigned every EnOcean transmitter during its production process.

GP - **G**eneric **P**rofiles; Specification to define the structure of over-the-air data bytes for EnOcean transmitters. Also see EEP.

HTTP – **H**yper**t**ext **T**ransfer **P**rotocol

IoT – **I**nternet **O**f **T**hings

JSON - **J**ava **S**cript **O**bject **N**otation

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/

[E3] EnOcean Remote Management Specification.
https://www.enocean-alliance.org/reman/

[E4] EnOcean Wireless Standard
https://www.enocean-alliance.org/about-us/enocean-wireless-standard/

[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
https://tools.ietf.org/html/rfc7159

[RFC2] RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing
https://tools.ietf.org/html/rfc7230

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

4BS

RORG	Data				Sender ID				Status
A5	DB_3	DB_2	DB_1	DB_0	ID_3	ID_2	ID_1	ID_0	1Byte

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

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

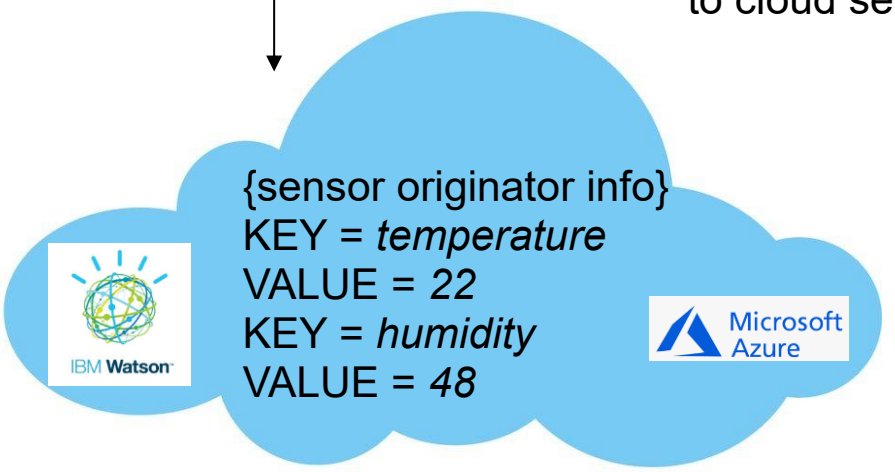

Figure 2: Message flow

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.

D2-01-00: Electronic switches and dimmers with Energy Measurement and Local Control

Direction "to": Actuator Set Output

Key	Value	Meaning
switch	off	Output value OFF
	on	Output value ON

Direction "from": Actuator Status Response

Key	Value	Meaning
localControl	off	Local control disabled / not supported
	on	Local control enabled
switch	off	Output value OFF
	on	Output value ON

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

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

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

Property	Type	Value/Formatting	Description
eep	String	xx-xx-xx	EnOcean Equipment Profiles, definition of Enocean radio telegram structure
title	String	May be translated	short description of eep profile
variations	Array of objects	[{}]	Information about available versions of the profile

Table 2: profiles array object

Property	Type	Value/Formatting	Description
direction	String	from | to	direction of transport (*from* or *to* device)
version	Float	x.x	Version of the EO/IP profile (not EEP version) .

Table 3: variations object

JSON for profiles object

```
 "profiles" : [ {
    "eep" : "A5-02-01",
    "title" : "Temperature Sensors, Temperature Sensor Range -40°C to 0°C",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-02",
    "title" : "Temperature Sensors, Temperature Sensor Range -30°C to +10°C",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  },
…
]
```

Figure 4: profiles JSON example

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

Property	Type	Value/Formatting	Description	Opt
eep	string	**xx-xx-xx**	EnOcean Equipment Profiles, definition of Enocean radio telegram structure. not shown in /device/profile	
title	string	May be translated	description of eep profile not shown in /device/profile	
functionGroups	array of objects	[{}]	array of *functionGroup* objects	

Table 4: profile object

Property	Type	Value/Formatting	Description	Opt
title	String	May be translated	Title of the group	
direction	string	from | to	direction of transport (*from*, *to* device or both)	
functions	array of objects	[{}]	array of function objects	

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 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

JSON for profile object

```
"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"
```

```
        } ],
        "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"
```

```
        }, {
          "value" : "on",
          "meaning" : "Output value ON"
        } ]
      } ]
    }, {
      "title" : "Actuator Query",
      "direction" : "to",
      "functions" : [ {
        "key" : "query",
        "values" : [ {
          "value" : "status",
          "meaning" : "Query status"
        } ]
      } ]
    } ],
    "version" : 0.9
  }
```

Figure 6: profile D2-01-0A JSON example

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

Property	Type	Value/Formatting	Description
deviceId	String	4 byte hex value (8 characters with leading zeros)	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.
friendlyId	String		user-assigned name of EnOcean device
physicalDevice	String		group name of actuators and sensors of the same physical device

Table 9: devices object

JSON for devices array object

```
"devices" : [ {
    "deviceId" : "FEFFFF60",
    "friendlyId" : "BathroomSwitch"
  },
{
    "deviceId" : "01843197",
    "friendlyId" : "OutDoorToiletTemperature"
  }, {
    "deviceId" : "00435678",
    "friendlyId" : "ButtonSwitch",
    "physicalDevice" : "MultiDevice"
  }, {
    "deviceId" : "00435679",
```

```
    "friendlyId" : "Controler",
    "physicalDevice" : "MultiDevice"
  } ]
}
```

Figure 7: devices array JSON example

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}
-

Property	Type	Value/Formatting	Description	Opt
deviceId	string	4 byte hex value	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.	
friendlyId	string		user-assigned name of EnOcean device	
physicalDevice	string		group name of actuators and sensors of the same physical device	opt
eeps	array	[{}]	EEPS supported by the device	
manufacturer	string		Based on manufacturer ID if send	opt
location	string		Location of the device	opt
dbm	integer		dBm of last received telegram	opt
firstSeen	DateTime	yyyy-mm-ddT hh:mm:ss.sss+hhmm	timestamp on which the device has been detected for the first time	
lastSeen	DateTime	yyyy-mm-ddT hh:mm:ss.sss+hhmm	timestamp on which the device has been seen last time	opt

Table 10: device Object

Property	Type	Value/Formatting	Description
eep	String	XX-XX.XX	EnOcean Equipment Profiles, definition of Enocean radio telegram structure
version	string	x.x	API Version of the profile

direction	String	from | to | both	Supported directions	
baseIdChannel	integer		BaseID Channel transmitted to device, this channel (0 - 128) is automatically assigned from Interface.	

Table 11: eep Object

JSON for device object

```
"device" : {
    "deviceId" : "01910188",
    "friendlyId" : "Opus-Bridge-2K",
    "eeps" : [ {
      "eep" : "D2-01-11",
      "version" : 0.9,
      "direction" : "both"
    }, {
      "eep" : "F6-02-01",
      "version" : 0.9,
      "direction" : "from"
    }, {
      "eep" : "F6-03-01",
      "version" : 0.9,
      "direction" : "from"
    } ],
    "manufacturer" : "Jaeger Direkt",
    "firstSeen" : "2016-10-27T07:59:07.137+0200",
    "lastSeen" : "2016-11-30T12:51:33.350+0100",
    "dbm" : -70
  }
```

Figure 8: device JSON example

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

Property	Type	Value/Formatting	Description
deviceId	string	4 byte hex value	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.

friendlyId	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

Property	Type	Value/Formatting	Description	Opt
key	string		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*	
channel	integer			opt
value	float			
unit	string	**[unit]**		opt
meaning	string	Description of the value, may be translated to native client language.		opt
timestamp	DateTime	yyyy-mm-ddT hh:mm:ss.sss+ hhmm		
age	integer	In ms		

Table 13: functions array object

JSON for state object

```
"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

Property	Type	Value/Formatting	Description
deviceId	4 byte hex value	xxxxxxxx	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.
friendlyId	string		user-assigned name of EnOcean device
physicalDevice	string		group name of actuators and sensors of the same physical device
timestamp	DateTime	yyyy-mm-ddT hh:mm:ss.sss+hhmm	telegram transmission time in UTC format
direction	string	from | to	direction of transport (*from* device sent / *to* device sent)
functions	array of objects	[{}]	array of function objects
telegramInfo	object	{}	telegramInfo object

Table 14: telegram object

Property	Type	Value/Formatting	Description	Opt
key	string		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*	
channel	integer			opt
value	float			
unit	string	**[unit]**		opt
meaning	string	Description of the value, may be translated to native client language.		opt
timestamp	DateTime	yyyy-mm-ddT hh:mm:ss.sss+hhmm		

age	integer			

Table 15: functions array object

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

Table 16: telegramInfo object

JSON for telegram Object

```
"telegram" : {
    "deviceId" : "01843197",
    "friendlyId" : "temp",
    "physicalDevice" : "testdevice",
    "timestamp" : "2016-03-16T18:12:16.134+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