System Specification

enocean alliance
Building Smarter Connectivity

EnOcean over IP REST API Implementation example Specification

V 1.2

Approved for release: 28th October 2020

San Ramon, CA, USA, 2021

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

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
1.1	AP	Derived from former EoIP specification	Aug 17, 2020
1.2	AP	Moved json objects to EoIP specification	March 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 REST API Implementation 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.
5000 Executive Parkway, Suite 302
San Ramon, CA 94583
USA
Graham Martin
Chairman & CEO EnOcean Alliance

Table of Contents

Index of Figures

Index of Tables

1. Introduction

1.1. Scope and Purpose

This document is intended for consumers and producers of EAGs (EnOcean Alliance IP Gateways) to implement or use a general REST (Representational State Transfer) JSON (java script object notation) API (application programming interface), which simplifies the EnOcean Alliance World to third parties. All techniques, explained here, are based on standardized IP and EnOcean Alliance Technologies. One of the main aims of the specification is to remove the complexity of the EnOcean profiles for consumers and to allow them to use/integrate EnOcean technology simply in their products. For manufacturers of an EAG, this document specifies how different profiles and EnOcean technologies are mapped to a JSON-REST API. This document is not intended to create a User Manual or Product Specification for an EAG, the implementer may add additional features to his gateway. The specification should be a base line for an EAG, which allows a basic usage for EnOcean Alliance Products. The API does not contain any information nor has the aim to generate a logic/rule machine for automatic interaction with EnOcean devices. The API is a baseline to allow other vendors/app developer or rule machines to connect to the EAG.

Chapter 1 mentions the general scope, acronyms, definitions,
Chapter 2 describes the general REST API and gives a short introduction to REST
Chapter 3 describes the available resources in details.

Possible use cases of the API are shown in Figure 1 and Figure 2. Figure 1 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 2 shows the general possibility to use the API to connect EnOcean with other APIs for IoT/Smart Home Systems, or as direct connector for the manufacturer of a Smart Home Rule engine.

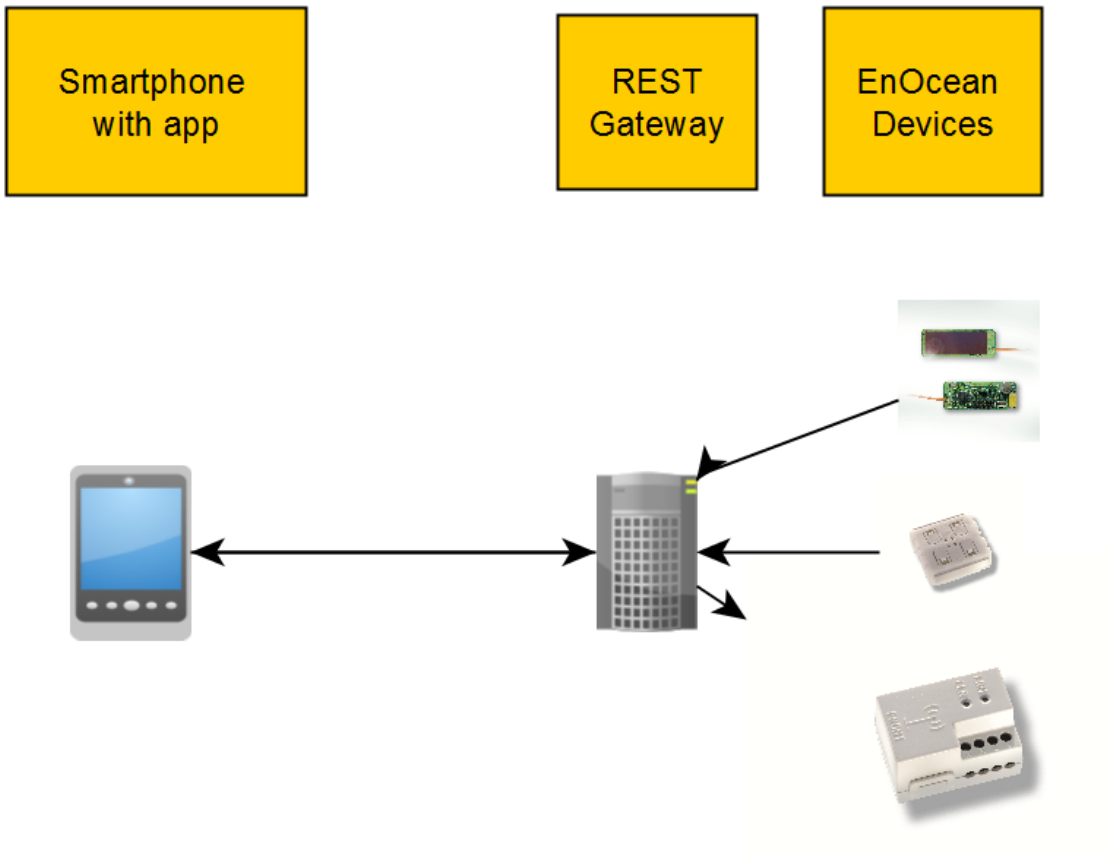

Figure 1: A possible use case. API for smartphone apps

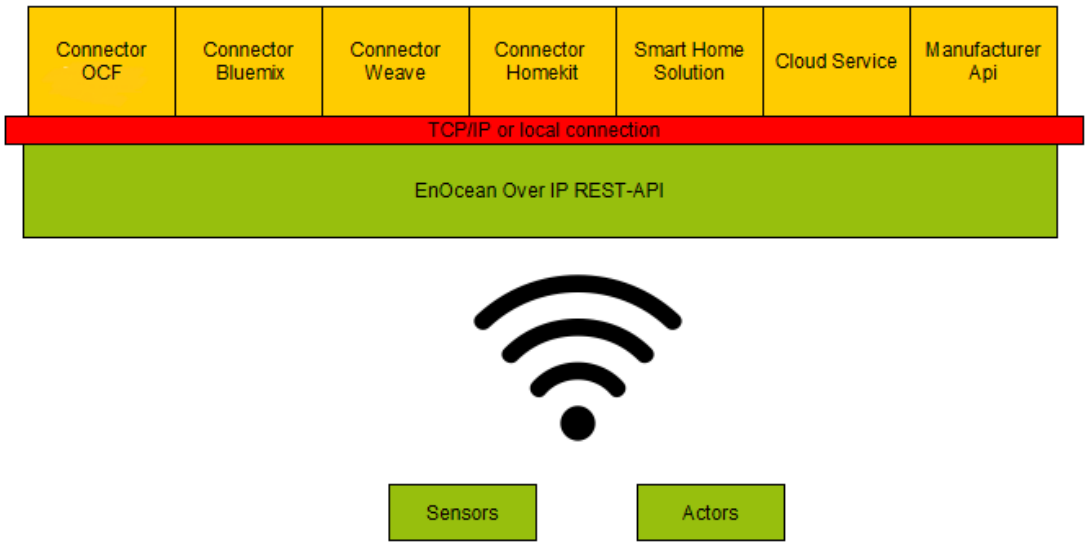

Figure 2: A possible use case. API as connector to other APIs or smart home gateway

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 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/

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 – A REST Interface

2.1. Motivation

With the emergence of IoT technology it is necessary that 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 a REST Gateway to allow 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 API

JSON-REST has been chosen as it is based on open standards, such as HTTP 1.1 and JSON, and it is field proven to be easy to use and implement.

2.2. Implementation Requirements

An EAG shall implement all resources defined in chapter 3. The JSONs shall contain all the data fields as specified. A manufacturer may extend the API, REST resources and JSON to add additional features or data to their system. It should allow multiple connections to the system, so that multiple clients may use the system at the same time.

As some EnOcean Devices only transmit periodically, the EAG needs to store the last state of the device in memory and depending on the parameters described in chapter 3.1.5. report them differently.

The EAG shall allow atomic addressing of device functions, e.g. if only the light intensity parameter is changed for one device, but the EEP contains other data fields, the EAG shall construct the whole EEP including default values for other data fields.

Due to upcoming enhancements and vendor specific extensions a client implementing the API has to respect the robustness principle ("be liberal in what you accept and conservative in what you send"). In terms of the API this especially implies a certain tolerance in consuming REST responses, which could be extended by additional data

New versions of JSON API will always be backwards compatible using a *never remove, only add* strategy.

2.3. Security recommendations

The Security of the REST API is not part of this spec and manufacturer specific, but the EAG should use TLS/HTTPS and an authentication method, which is state of the art.

2.4. JSON-REST API

2.4.1. Introduction to RESTful HTTP

REST is an architectural design to define how HTTP requests and URIs should be used. The key principles of REST are to give everything which is useable an ID, link things together, Make use of existing standards (e.g. HTTP), stateless communication and to allow a self-describing use of the URI/API.

The REST API of an EAG shall be based on standard HTTP Request (see chapter 2.4.2.). It shall provide the URI Resources described in chapter 3.3 and the Resources described in chapter 3.2. The information shall be displayed and described using JSON Format.

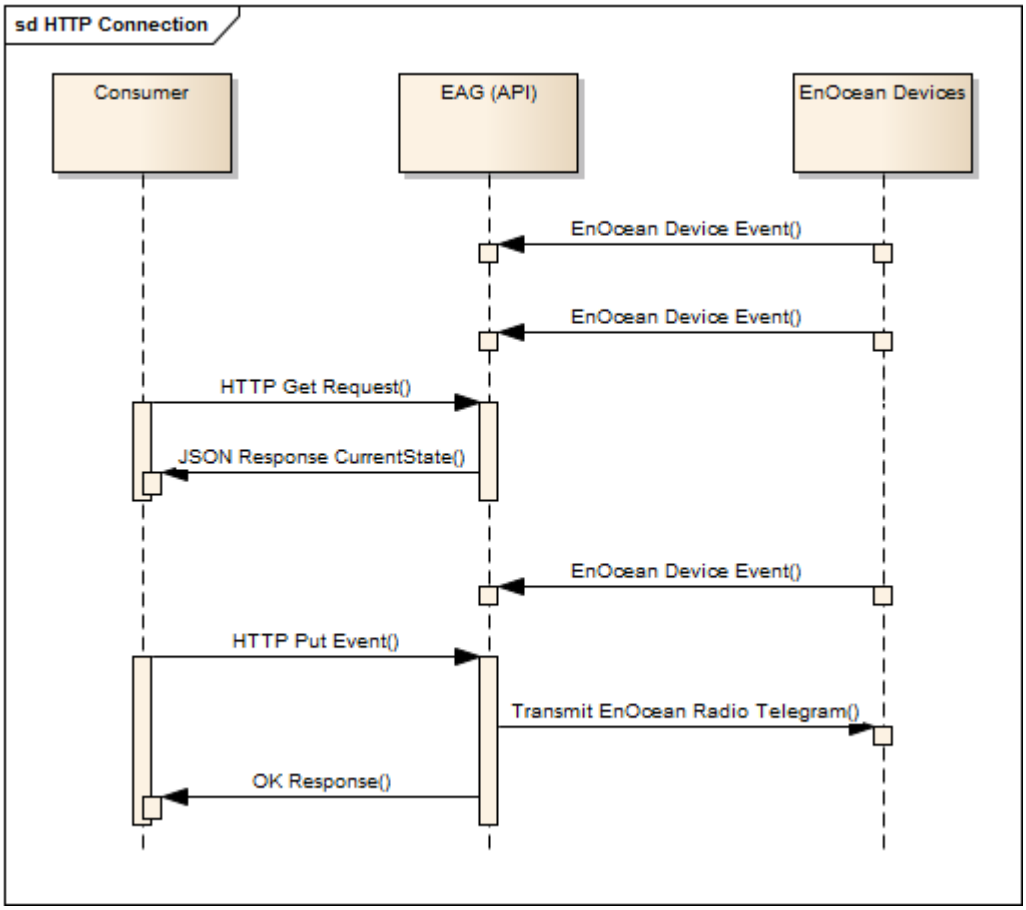

Figure 3: High level description of the HTTP connection

2.4.2. Used HTTP methods

GET: Request information about a resource and, if available, the resource is displayed.

It is used to:

- Get the system state
- Get a profile description
- Get device Information
- Get the state of a device (Last transmitted state)

PUT: Updates (or creates) the specified resource

- Update the state of a device (triggers a transmission to the device)

2.4.3. HTTP Responses

HTTP Response Code	Description
200 (OK)	Returned when an application request is successful.
202 (Accepted)	The request has been accepted and has not been yet completed.
400 (Bad Request)	Returned when an error has occurred. In addition, an error key in the header object is transferred
401 (Unauthorized)	Returned when the application name/password is invalid.
403 (Forbidden)	Returned when an application is not allowed to make the request, such as when the application is inactive or the client's IP address is not in the list of known application addresses.
404 (Not Found)	Returned when the requested resource does not exist. For example, the profile does not exist, the device does not exist.
500 (Internal Error)	Returned when a error on Interface side has occurred. In addition, an error key in the header object is transferred

Table 1: HTTP codes

2.4.4. Introduction to JSON

JSON is a type of syntax for storing and exchanging data in lightweight data-interchange format.

JSON Example

```
{
  “variableName” : ”AString”,
  “ArrayName” : [
    { “Name”:”RogerRabbit”, ”Company”:”ACME”},
    { “Name”:”Brainiac”, ”Company”:”ACME Development”}
   ],
  “object”: {
    “objectMember1”:”Value1”,
    “objectMember2”:”Value2”
  }
}
```

Figure 4: JSON Example

The example in Figure 4: JSON Example shows the basic structure of a JSON file. It uses ‘{‘ and ‘}’ to start and close objects.

“Variable Name” : “Variable Value” is used to give a variable a value. Variable values can be of the type string, character, integer, Boolean, object and null.

Arrays are opened with ‘[‘ and closed with ‘]’.

For more information about JSON, the JSON website is a good reference [O11].

3. REST API Description

3.1. Introduction

3.1.1. Basic JSON structures

The JSON Interface shall use UTF-8 for the encoding of the content. Each response shall contain a header and shall have the following structure

Basic Response Message

```
{
  "header": {
     "status": “HTTP status code”,
     "code": "Internal Gateway status code",
     "message" : "Internal Gateway status code",
     "content": "content of Message",
     "gateway" : "current software version",
     "timestamp": "transmission time in UTC format"
  },
  "object":[
     {
        ".." : "..",
        "...": "..."
     }
  ]
}
```

Figure 5: Response Message Structure

Property	Type	Value / Formatting	Description	Opt
status	number	response code	HTTP Response Code	
code	number	code	Status Information Code	opt
message	string		Status information message	opt
content	string	systemInfo, profiles, profile, devices, device, telegrams, state, stream	message type	
gateway	string	Interface build	Software version of Interface	
timestamp	DateTime	yyyy-mm-ddT hh:mm:ss.sss+ hhmm	transmission time in UTC format	

Table 2: Header object description

3.1.2. Return codes of the base Object

An EAG shall always add a code in the header object. The response codes are fragmented in four main areas.

- 1000-1999: Request OK
- 2000-2999: Error - URL or Syntax error
- 3000-3999: Error – Syntax OK but wrong content
- 8000-8999: Error – Internal error of the Gateway
- 4000-7999 and 9000-9999 are reserved.

1000-1999: Request OK

Code	HTTP Status	Meaning	Suggested Message
1000	200		OK
1002	202	Request ok, it will be executed later.	Will send telegram later
1500-1999		Reserved for manufacturer specific codes	

Table 3 Return codes 1000-1999

2000-2999: Error - URL or Syntax error

Code	HTTP Status	Meaning	Suggested Message
2000	400	invalid REST Resource, e.g. /devices/<deviceId>/invalid	invalid URL %s
2001	404	unknown Path, e.g. /unknown/resource	%s not found
2002	400	e.g. DELETE /profiles/F6-02-01	unsupported request method %s
2003	400	e.g. /devices/telegrams?direction=invalid	parameter %s has unknown value %s
2004	400	POST without DATA	no data
2005	400	Malformed JSON	
2500-2999		Reserved for manufacturer specific codes	

Table 4 Return codes 2000-2999

3000-3999: Error – Syntax OK but wrong content

Code	HTTP Status	Meaning	Suggested Message
3000	400	e.g. D4-01-02	invalid EEP %s
3001	400	valid but not supported EEP	unsupported EEP %s
3100	404	e.g. /devices/unknown	unknown device %s
3101	400	device exists but is e.g. in learn in state	device %s in unsupported state
3102	400	POST of different devices with e.g. same friendlyId	duplicated ID %s
3103	400	POST of device without deviceID and without IoT label	Missing deviceId
3120	400	no function	no functions to set specified
3121	400	missing function(s)	incomplete request - function(s) missing: %s
3122	400	unknown function	unknown function %s
3123	400	duplicate function	duplicated function %s
3124	400	function without key	function without key
3125	400	function without value	missing value of function %s
3126	400	function with invalid value	invalid value %s of function %s
3127	400	function with invalid value	invalid value %s of function s, only scalar allowed
3128	400	function with value out of range	invalid value %s of function s, min: s, max:%s)
3500-3999		Reserved for manufacturer specific codes	

Table 5: Return codes 3000-3999

8000-8999: Error – Internal error of the Gateway

Code	HTTP Status	Meaning	Suggested Message
8000	500		internal error (%s)
8001	500		not implemented
8500-8999		Reserved for manufacturer specific codes	
9000-9999		Reserved	

Table 6: Return codes 8000-8001

3.1.3. Resources base path overview

A gateway shall implement the following base resources. From each base resources sub resources are available which are listened in the containing chapter.

Resource URI	Content
/system	Contains all information about the System (Chapter 3.2.)
/profiles	Contains information about supported profiles and specific Information about specific Profile. (Chapter 3.3.)
/devices	List of all devices, state of devices and access to each device separately. (Chapter 3.4.)

Table 7: Resource overview

3.1.4. Object overview

Following objects are used and generated in the response messages. These objects are intended as help and are not resources. All the following objects shall be generated by the different resources.

Object name	Content	Generated by following Resources
systemInfo	Information about the System	/system/info
profiles	List of all available profiles of the gateway.	/profiles
profile	Information about the used EEP-Profile. This is a list of all available states which can be set/read for a certain device.	/profiles/{EEP} /devices/{ID}/profile
devices	List of all devices	/devices
device	Information about a device	/devices/{ID}
state	Current state of a device	/devices/{ID}/state /devices/states
stream	Represent a received status update / telegram of a device	/devices/stream /devices/{ID}/stream
telegrams	Log of status updates/telegrams	/devices/telegrams /devices/{ID}/telegrams

Table 8: Object overview

3.1.5. State reporting Options

The EAG shall implement a feature which is used to decide when the state of each key is transmitted and it shall be possible to read or modify the setting. Three different options are available and shall be part of the device object. The default settings should be as described below, for each attribute, but may be modified by the manufacturer to better reflect different device types.

- **TransmitOnConnect**

  If this attribute is set to true, the resources which respond with a state object, shall report the last received state. In makes sense to set this attribute to false for rocker/button switches as the last State of a rocker Switch is in most cases “release” and the control (“press”) action has already happened.

  Attention: If the attribute TransmitOnConnect is not set for any device, the transfer of live states starts with the first received telegram of a known device after the connection of a client.

  Default for all devices should be: On

- **TransmitOnEvent**

  If this attribute is set, then the value shall be transmitted after each received telegram. The other case could be useful for some devices where not the whole range of EEP values is used. For example, a device uses an EEP which transmits temperature and moisture but the device itself has no temperature sensor. To suppress the permanent transfer of “Temp 0 degree” this value can be hidden in the API transmission.

  Default for all devices should be: On

- **TransmitOnDuplicate**

  If this attribute is set, then the Interface shall transmit received values of devices that have not changed. For example, a window sensor transmits cyclically every 10 minutes his "closed" state. If this attribute is not set, then the Interface suppresses the output to the API.

  Since the cyclic transmission of values is often an attribute of the EnOcean protocol, as a keep alive signal, in most cases it is sufficient to only transmit a value when it changes. As it could be interesting for maintenance or security reason to get the information of duplicated telegrams, this attribute can be modified.

  Default for all devices should be: Off

3.1.6. Streaming API Resources

The Streaming API Resources keep a HTTP connection open to push updates to the client. The EAG shall implement the possibility to keep the HTTP connection open and still allow other clients to connect the EAG for REST or additional streaming API connections. A streaming API connection has the advantage compared to polling that information updates are transferred immediately after an event has happened. This also allows for an optimized usage of hardware and network connection resources and is way faster than a connection establishment with HTTP(S) for a callback.

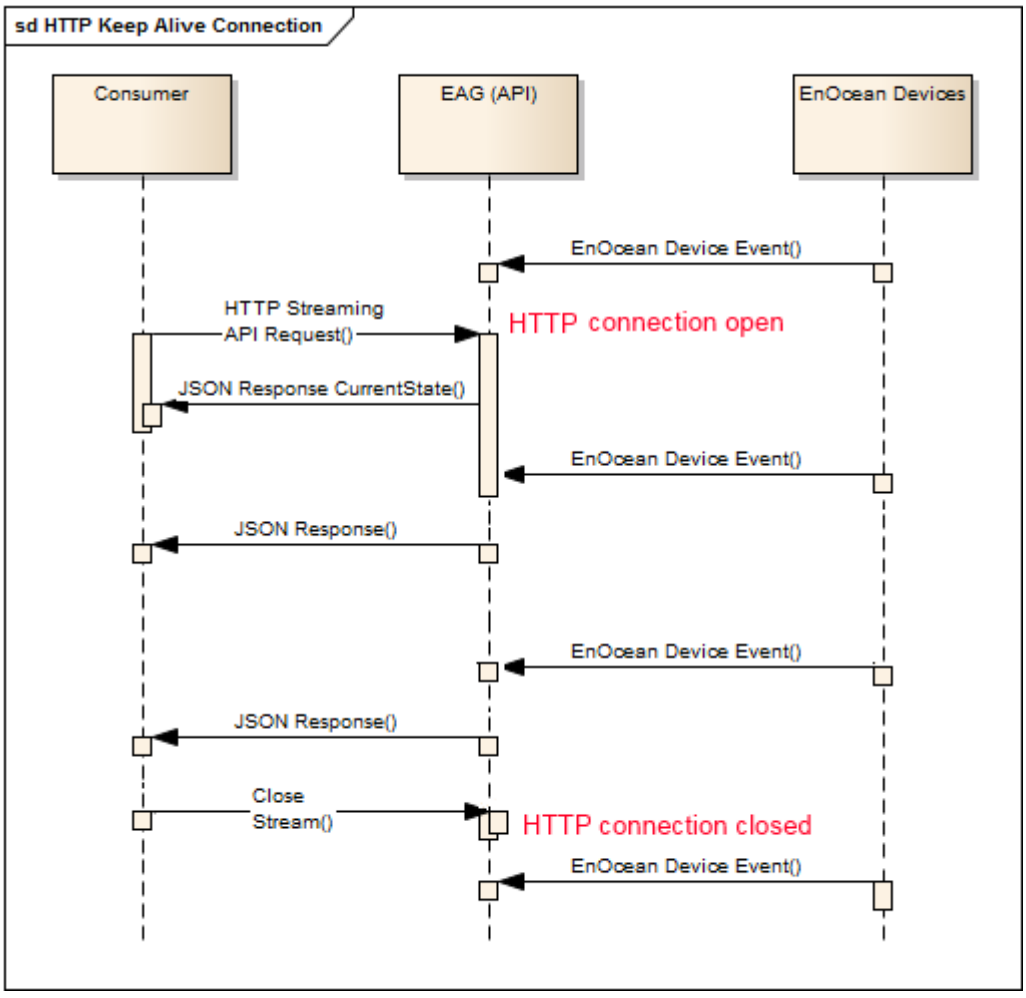

Figure 6: HTTP Keep-Alive Connection

3.1.7. JSON formatting and delimiting options for Streaming API

The URI resources which use streaming may be influenced in the way that their data is formatted and delimited.

Format

- **formatted (default)**
  The JSON Object are formatted in a way that they can be read by Humans. This option shall not be usable with newline.
- **singleLine**
  Each JSON Object is given out on a single line.

```
JSON for single Line Representation
{"header":{…},"states” : […]}]}
```

Figure 7 single Line Json Example

Delimiter

- **length**
  Every JSON Object is announced by its length in bytes followed by a CR+LF. (identical to "lengthBytes" because of conformity to popular APIs)
- **lengthBytes**
  Every JSON Object is announced by its length in bytes followed by a CR+LF.
- **lengthCharacters**
  Every JSON Object is announced by its length in characters followed by a CR+LF. (The code page of stream is UTF-8 and the number of bytes per character can differ)
- **newline**
  Every JSON Object is separated from the next Object by a CR+LF. The output format must be set to **singleLine**.
- **emptyLine (default)**
  Every JSON Object is separated from the next Object by a CR+LF+CR+LF

3.2. System Resources

3.2.1. GET /system/info

This command requests information about the system.

GET http://hostname:port/system/info

```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "systemInfo",
    "gateway" : "Interface",
    "timestamp" : "2016-03-15T13:13:49.916+0100"
  },
  "systemInfo" : {
    "version" : "Interface v0.9.1 2016.03.15 14:43",
    "baseId" : "FFA04700",
    "possibleBaseIdChanges" : 10,
    "eurid" : "0185408E",
    "frequency" : 868
  }}
```

Figure 8: /system/info JSON response

3.3. Profiles Resources

3.3.1. GET /profiles

This command returns an overview of all supported profiles and the profile version of the EAG Interface.

GET http://hostname:port/profiles


```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "profiles",
    "gateway" : "interface",
    "timestamp" : "2016-05-09T16:25:04.671+0200"
  },
  "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
    } ]
  },
…
  {
    "eep" : "D2-01-08",
    "title" : "Electronic switches and dimmers with Energy Measurement and Local Control",
    "variations" : [ {
      "direction" : "both",
      "version" : 1.0
    } ]
}
```


Figure 9: /profiles JSON response example

3.3.2. GET /profiles/{eepId}?version=x.x

This command returns information about a certain profile.

Format of {eepId} = RORG-FUNC-TYPE of the EEP to be shown in detail.

Parameters	Type	Value/formatting	Description
version	String	x.x	Profile description version to use

Table 9: optional parameters of /profiles

GET http://hostname:port/profiles/A5-20-04


```
"profile" : {
    "eep" : "A5-20-04",
    "title" : "HVAC Components, Heating Radiator Valve Actuating Drive with Feed and Room
Temperature Measurement, Local Set Point Control and Display",
    "functionGroups" : [ {
      "direction" : "from",
      "functions" : [ {
        "key" : "errorCode",
        "values" : [ {
          "value" : "batteryEmpty",
          "meaning" : "Battery is empty"
        }, {
          "value" : "blockedValve",
```

```
          "meaning" : "Blocked valve"
        }, {
          "value" : "endpointDetectionError",
          "meaning" : "End point detection error"
        }, {
          "value" : "frostProtection",
          "meaning" : "Frost protection"
        }, {
          "value" : "measurementError",
          "meaning" : "Measurement error"
        }, {
          "value" : "noError",
          "meaning" : "No error reported"
        }, {
          "value" : "noReponseFromController",
          "meaning" : "No response from controller"
        }, {
          "value" : "noValve",
          "meaning" : "No valve"
        }, {
          "value" : "notTaughtIn",
          "meaning" : "Not taught in"
        }, {
          "value" : "teachInError",
          "meaning" : "Teach-in error"
        } ]
      }, {
        "key" : "feedTemperature",
        "description" : "Current feed temperature value",
        "values" : [ {
          "range" : {
            "min" : 20,
            "max" : 80,
            "step" : 0.235,
            "unit" : "°C"
          }
        } ]
      }, {
        "key" : "locked",
        "description" : "Shows if all buttons on the actuator are locked",
        "values" : [ {
          "value" : "false",
          "meaning" : "Manual room temperature selection enabled"
        }, {
          "value" : "true",
          "meaning" : "Manual room temperature selection disabled"
        } ]
      }, {
        "key" : "measurement",
        "description" : "Measure feed + room temperature",
        "values" : [ {
          "value" : "off",
          "meaning" : "Measure feed + room temperature is inactive"
        }, {
          "value" : "on",
          "meaning" : "Measure feed + room temperature is active"
        } ]
      }, {
        "key" : "query",
        "description" : "Request for status from the controller",
        "values" : [ {
          "value" : "noChange",
          "meaning" : "No change"
        }, {
          "value" : "status",
          "meaning" : "Status requested"
        } ]
      }, {
        "key" : "roomTemperature",
```

```
        "description" : "Current room temperature",
        "values" : [ {
          "range" : {
            "min" : 10,
            "max" : 30,
            "step" : 0.078,
            "unit" : "°C"
          }
        } ]
      }, {
        "key" : "temperatureSetPoint",
        "description" : "Current temperature set point",
        "values" : [ {
          "range" : {
            "min" : 10,
            "max" : 30,
            "step" : 0.078,
            "unit" : "°C"
          }
        } ]
      }, {
        "key" : "valve",
        "description" : "Current valve position",
        "values" : [ {
          "range" : {
            "min" : 0,
            "max" : 100,
            "step" : 1,
            "unit" : "%"
          }
        } ]
      } ]
    }, {
      "direction" : "to",
      "functions" : [ {
        "key" : "command",
        "description" : "Initiates certain temporary service operations",
        "values" : [ {
          "value" : "closeValve",
          "meaning" : "Close valve"
        }, {
          "value" : "noChange",
          "meaning" : "No change"
        }, {
          "value" : "openValve",
          "meaning" : "Open valve"
        }, {
          "value" : "runInit",
          "meaning" : "Run initialisation"
        } ],
        "defaultValue" : "noChange"
      }, {
        "key" : "displayOrientation",
        "description" : "Adjusts the display orientation",
        "values" : [ {
          "range" : {
            "min" : 0,
            "max" : 270,
            "step" : 90,
            "unit" : "°"
          }
        } ],
        "defaultValue" : 0
      }, {
        "key" : "locked",
        "description" : "Set the button lock status",
        "values" : [ {
          "value" : "false",
          "meaning" : "Manual room temperature selection enabled"
```

```
        }, {
          "value" : "true",
          "meaning" : "Manual room temperature selection disabled"
        } ],
        "defaultValue" : "false"
      }, {
        "key" : "measurement",
        "description" : "Measure feed + room temperature",
        "values" : [ {
          "value" : "off",
          "meaning" : "Disable measurement for energy saving"
        }, {
          "value" : "on",
          "meaning" : "Enable feed + room temperature measurement"
        } ],
        "defaultValue" : "on"
      }, {
        "key" : "temperatureSetPoint",
        "description" : "Temperature set point",
        "values" : [ {
          "range" : {
            "min" : 10,
            "max" : 30,
            "step" : 0.078,
            "unit" : "°C"
          }
        } ]
      }, {
        "key" : "valve",
        "description" : "Valve position",
        "values" : [ {
          "range" : {
            "min" : 0,
            "max" : 100,
            "step" : 1,
            "unit" : "%"
          }
        } ]
      }, {
        "key" : "wakeUpCycle",
        "description" : "Defines the cyclic wake-up time",
        "values" : [ {
          "meaning" : "Cyclic wake-up time, 30 secs - 25 min in 30 seconds steps",
          "range" : {
            "min" : 30,
            "max" : 1500,
            "step" : 30,
            "unit" : "s"
          }
        }, {
          "meaning" : "Cyclic wake-up time, 3-42 hours in 3 hours steps",
          "range" : {
            "min" : 10800,
            "max" : 151200,
            "step" : 10800,
            "unit" : "s"
          }
        }, {
          "value" : "10",
          "meaning" : "Minimal cyclic wake-up time = 10 seconds"
        } ],
        "defaultValue" : 600
      } ]
    } ]
  }
}
```


Figure 10: /profiles/A5-20-04 JSON response example

3.4. Devices Resource

3.4.1. GET /devices

This command returns a list of known devices for this EAG.

GET http://hostname:port/devices


```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "devices",
    "gateway" : "Interface",
    "timestamp" : "2001-01-01T01:22:04.802+0100"
  },
  "devices" : [ {
    "deviceId" : "FEFFFF60",
    "friendlyId" : "BathroomLightSwitch"
  }, {
    "deviceId" : "0014AAD8",
    "friendlyId" : "IlluminationSensor"
  }, {
    "deviceId" : "01843197",
    "friendlyId" : "OutdoorBathRoomTemperatureSensor"
  } ]
}
```


Figure 11: /devices/JSON response

3.4.2. GET /devices/states

This command returns the current state of all known devices.

GET http://hostname:port/devices/states


```
    {
      "header" : {
        "httpStatus" : 200,
        "content" : "states",
        "gateway" : "interface",
        "timestamp" : "2015-10-15T11:22:45.594+0200"
      },
    "states" : [ {

    "deviceId" : "00851E54",
    "friendlyId" : "piotek",
    "functions" : [ {
      "key" : "pirStatus",
      "value" : "off",
      "meaning" : "PIR off",
      "timestamp" : "2016-05-09T16:33:36.984+0200",
      "age" : 9533
    }, {
      "key" : "supplyVoltage",
      "value" : 2.18,
      "unit" : "V",
      "timestamp" : "2016-05-09T16:33:36.984+0200",
      "age" : 9533
    } ]
```

```
  }, {
    "deviceId" : "0181A5BC",
    "friendlyId" : "EnOceanTemperature",
    "functions" : [ {
      "key" : "temperature",
      "value" : 25.57,
      "unit" : "°C",
      "timestamp" : "2016-05-09T16:27:08.441+0200",
      "age" : 398076
    } ]
  }, {
    "deviceId" : "01844BB0",
    "friendlyId" : "contact",
    "functions" : [ {
      "key" : "contact",
      "value" : "open",
      "meaning" : "Open",
      "timestamp" : "2016-05-09T16:22:54.496+0200",
      "age" : 652021
    } ]
  }, {
    "deviceId" : "0180ADC2",
    "friendlyId" : "Z12aTempOutsideEast",
    "functions" : [ {
      "key" : "temperature",
      "value" : 26.67,
      "unit" : "°C",
      "timestamp" : "2016-05-09T16:30:18.140+0200",
      "age" : 208377
    } ]
  }, {
    "deviceId" : "0185570D",
    "friendlyId" : "VT5-1_Left",
    "physicalDevice" : "RoomPanel_FTR55D",
    "functions" : [ {
      "key" : "slideSwitch",
      "value" : "pos0DayOn",
      "meaning" : "Slide switch On/Day",
      "timestamp" : "2016-05-09T16:20:54.113+0200",
      "age" : 772404
    }, {
      "key" : "temperature",
      "value" : 24.31,
      "unit" : "°C",
      "timestamp" : "2016-05-09T16:20:54.113+0200",
      "age" : 772404
    }, {
      "key" : "temperatureSetPoint",
      "value" : 18.35,
      "unit" : "°C",
      "timestamp" : "2016-05-09T16:20:54.113+0200",
      "age" : 772404
    } ]
  }, {
    "deviceId" : "018F8927",
    "friendlyId" : "VT5-1_Right",
    "physicalDevice" : "TempSensor_FTF65S",
    "functions" : [ {
      "key" : "temperature",
      "value" : 23.53,
      "unit" : "°C",
      "timestamp" : "2016-05-09T16:28:40.853+0200",
      "age" : 305664
    } ]
  }, {
    "deviceId" : "018FF884",
    "friendlyId" : "SR06-LCD-2T",
    "functions" : [ {
      "key" : "setPoint",
```

```
      "value" : 0.00,
      "unit" : "°C",
      "timestamp" : "2016-05-09T10:21:37.043+0200",
      "age" : 22329474
    }, {
      "key" : "temperature",
      "value" : 22.90,
      "unit" : "°C",
      "timestamp" : "2016-05-09T10:21:37.043+0200",
      "age" : 22329475
    } ]
  }, {
    "deviceId" : "018720FE",
    "friendlyId" : "afrisoCo2",
    "functions" : [ {
      "key" : "co2",
      "value" : 470.59,
      "unit" : "ppm",
      "timestamp" : "2016-05-09T16:30:31.381+0200",
      "age" : 195137
    } ]
  }, {
    "deviceId" : "01870183",
    "friendlyId" : "myPlug",
    "functions" : [ {
      "key" : "energy",
      "value" : 1172.99993,
      "unit" : "Wh",
      "timestamp" : "2016-05-09T16:33:12.474+0200",
      "age" : 34044
    }, {
      "key" : "power",
      "value" : 0.00,
      "unit" : "W",
      "timestamp" : "2016-05-09T16:32:55.432+0200",
      "age" : 51086
    } ]
  } ]
}
```


Figure 12: /devices/state JSON response

3.4.3. GET /devices/stream?direction={*dir*}&delimited={*delimited*}&output={*output*}

This command gets the actual status of devices and informs the client of any changes via a stream of telegram objects. The stream uses the chunked transfer encoding of the HTTP[RFC2].

Parameter	Datatype	Valid values	Default	Functionality
direction	String	from | to | both	both	Defines which state changes are shown. From the device(from), send from the EAG to the device(to), or both
delimited	String	length | lengthBytes | lengthCharacters | newline |emptyLine	emptyLine	Modifes the delimiter between array entries see chapter 3.1.7.
output	String	formatted | singleLine	formatted	Modifes the formatting between received events see chapter 3.1.7.

Table 10: optional parameters of /device/stream

GET **http://hostname:port/devices/stream**


```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "states",
    "gateway" : "interface",
    "timestamp" : "2016-05-09T16:34:52.056+0200"
  },
  "states" : [ {
    "deviceId" : "00851E54",
    "friendlyId" : "piotek",
    "functions" : [ {
      "key" : "pirStatus",
      "value" : "off",
      "meaning" : "PIR off",
      "timestamp" : "2016-05-09T16:34:37.316+0200",
      "age" : 14740
    }, {
      "key" : "supplyVoltage",
      "value" : 2.18,
      "unit" : "V",
      "timestamp" : "2016-05-09T16:34:37.316+0200",
      "age" : 14740
    } ]
  }, {
    "deviceId" : "0181A5BC",
    "friendlyId" : "EnOceanTemperature",
    "functions" : [ {
      "key" : "temperature",
      "value" : 25.57,
      "unit" : "°C",
      "timestamp" : "2016-05-09T16:27:08.441+0200",
      "age" : 463615
    } ]
  }, {
    "deviceId" : "01844BB0",
    "friendlyId" : "contact",
    "functions" : [ {
      "key" : "contact",
      "value" : "open",
      "meaning" : "Open",
```

```
      "timestamp" : "2016-05-09T16:22:54.496+0200",
      "age" : 717561
    } ]
  }, {
    "deviceId" : "0180ADC2",
    "friendlyId" : "Z12aTempOutsideEast",
    "functions" : [ {
      "key" : "temperature",
      "value" : 26.67,
      "unit" : "°C",
      "timestamp" : "2016-05-09T16:30:18.140+0200",
      "age" : 273917
    } ]
  }, {
    "deviceId" : "0185570D",
    "friendlyId" : "VT5-1_Left",
    "physicalDevice" : "RoomPanel_FTR55D",
    "functions" : [ {
      "key" : "slideSwitch",
      "value" : "pos0DayOn",
      "meaning" : "Slide switch On/Day",
      "timestamp" : "2016-05-09T16:20:54.113+0200",
      "age" : 837944
    }, {
      "key" : "temperature",
      "value" : 24.31,
      "unit" : "°C",
      "timestamp" : "2016-05-09T16:20:54.113+0200",
      "age" : 837944
    }, {
      "key" : "temperatureSetPoint",
      "value" : 18.35,
      "unit" : "°C",
      "timestamp" : "2016-05-09T16:20:54.113+0200",
      "age" : 837944
    } ]
  }, {
    "deviceId" : "018F8927",
    "friendlyId" : "VT5-1_Right",
    "physicalDevice" : "TempSensor_FTF65S",
    "functions" : [ {
      "key" : "temperature",
      "value" : 23.53,
      "unit" : "°C",
      "timestamp" : "2016-05-09T16:28:40.853+0200",
      "age" : 371204
    } ]
  }, {
    "deviceId" : "018FF884",
    "friendlyId" : "SR06-LCD-2T",
    "functions" : [ {
      "key" : "setPoint",
      "value" : 0.00,
      "unit" : "°C",
      "timestamp" : "2016-05-09T10:21:37.043+0200",
      "age" : 22395014
    }, {
      "key" : "temperature",
      "value" : 22.90,
      "unit" : "°C",
      "timestamp" : "2016-05-09T10:21:37.043+0200",
      "age" : 22395014
    } ]
  }, {
    "deviceId" : "018720FE",
    "friendlyId" : "afrisoCo2",
    "functions" : [ {
      "key" : "co2",
      "value" : 470.59,
```

```
      "unit" : "ppm",
      "timestamp" : "2016-05-09T16:30:31.381+0200",
      "age" : 260677
    } ]
  }, {
    "deviceId" : "01870183",
    "friendlyId" : "myPlug",
    "functions" : [ {
      "key" : "energy",
      "value" : 1172.99993,
      "unit" : "Wh",
      "timestamp" : "2016-05-09T16:34:12.655+0200",
      "age" : 39403
    }, {
      "key" : "power",
      "value" : 0.00,
      "unit" : "W",
      "timestamp" : "2016-05-09T16:33:55.602+0200",
      "age" : 56456
    } ]
  } ]
}

{
  "header" : {
    "content" : "telegram",
    "gateway" : "DC-GW/EO-IP v0.99.0b",
    "timestamp" : "2016-05-09T16:35:06.498+0200"
  },
  "telegram" : {
    "deviceId" : "018720FE",
    "friendlyId" : "afrisoCo2",
    "timestamp" : "2016-05-09T16:35:06.498+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "co2",
      "value" : 431.37,
      "unit" : "ppm"
    } ],
    "telegramInfo" : {
      "data" : "00003708",
      "status" : "0",
      "dbm" : -68,
      "rorg" : "A5"
    }
  }
}

{
  "header" : {
    "content" : "telegram",
    "gateway" : "DC-GW/EO-IP v0.99.0b",
    "timestamp" : "2016-05-09T16:35:07.479+0200"
  },
  "telegram" : {
    "deviceId" : "00851E54",
    "friendlyId" : "piotek",
    "timestamp" : "2016-05-09T16:35:07.479+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "supplyVoltage",
      "value" : 2.18,
      "unit" : "V"
    }, {
      "key" : "pirStatus",
      "value" : "off",
      "meaning" : "PIR off"
    } ],
    "telegramInfo" : {
```

```
      "data" : "6D05050F",
      "status" : "0",
      "dbm" : -74,
      "rorg" : "A5"
    }
  }
}
```


Figure 13: /system/info JSON response

3.4.4. GET /devices/telegrams?direction={*direction*}

This command shows the last available telegrams of all devices. The number of telegrams kept in the history depends on the EAG implementer.

Parameter	Datatype	Valid values	Default	Functionality
direction	String	from | to | both	Both	Defines which state changes are shown. From the device(from), send from the EAG to the device(to), or both

Table 11: optional parameters of /devices/telegrams

GET **http://hostname:port/telegrams**


```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "telegrams",
    "gateway" : "DC-GW/EO-IP v0.99.0b",
    "timestamp" : "2016-05-09T16:37:33.609+0200"
  },
  "telegrams" : [ {
    "deviceId" : "0181A5BC",
    "friendlyId" : "EnOceanTemperature",
    "timestamp" : "2016-05-09T12:57:08.358+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "temperature",
      "value" : 25.10,
      "unit" : "°C"
    } ],
    "telegramInfo" : {
      "data" : "00005F08",
      "status" : "0",
      "dbm" : -80,
      "rorg" : "A5"
    }
  }, {
    "deviceId" : "01870183",
    "friendlyId" : "myPlug",
    "timestamp" : "2016-05-09T12:57:21.943+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "power",
      "value" : 0.00,
      "unit" : "W"
    } ],
    "telegramInfo" : {
```

```
      "data" : "076000000000",
      "status" : "0",
      "dbm" : -74,
      "rorg" : "D2"
    }
  }, {
    "deviceId" : "00851E54",
    "friendlyId" : "piotek",
    "timestamp" : "2016-05-09T12:57:27.145+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "supplyVoltage",
      "value" : 2.18,
      "unit" : "V"
    }, {
      "key" : "pirStatus",
      "value" : "off",
      "meaning" : "PIR off"
    } ],
    "telegramInfo" : {
      "data" : "6D05050F",
      "status" : "0",
      "dbm" : -73,
      "rorg" : "A5"
    }
  }, {
    "deviceId" : "01870183",
    "friendlyId" : "myPlug",
    "timestamp" : "2016-05-09T12:57:38.990+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "energy",
      "value" : 1172.99993,
      "unit" : "Wh"
    } ],
    "telegramInfo" : {
      "data" : "072000000495",
      "status" : "0",
      "dbm" : -74,
      "rorg" : "D2"
    }
  },
….
]
```


Figure 14: /system/telegrams JSON response

3.4.5. GET /devices/{deviceId}

This command returns information about a specific device.

{deviceId} is either the 4 byte hex ID or the friendlyID of the device.

GET **http://hostname:port/devices/VT5-1_Right**


```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "device",
    "gateway" : "interface",
    "timestamp" : "2016-05-09T13:13:40.876+0200"
  },
  "device" : {
    "deviceId" : "018F8927",
    "friendlyId" : "VT5-1_Right",
    "physicalDevice" : "TempSensor_FTF65S",
    "eeps" : [ {
      "eep" : "A5-02-05",
      "version" : 1.0,
      "direction" : "from"
    } ],
    "manufacturer" : "Eltako",
    "firstSeen" : "2016-01-15T09:21:48.447+0100",
    "lastSeen" : "2016-05-09T13:03:16.862+0200"
  }
}
```


Figure 15: /devices/{deviceId} JSON response

3.4.6. GET /devices/{deviceId}/profile

This command returns the EEP used by this device.

GET **http://hostname:port/devices/VT5-1_Left/profile**


```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "profile",
    "gateway" : "inteface",
    "timestamp" : "2016-05-09T16:41:08.961+0200"
  },
  "profile" : {
    "functionGroups" : [ {
      "direction" : "from",
      "functions" : [ {
        "key" : "slideSwitch",
        "description" : "Slide switch  0/I or Slide switch Day/Night",
        "values" : [ {
          "value" : "pos0DayOn",
          "meaning" : "Slide switch On/Day"
        }, {
          "value" : "pos1NightOff",
          "meaning" : "Slide switch Off/Night"
        } ]
      }, {
        "key" : "temperature",
        "description" : "Temperature (linear)",
        "values" : [ {
          "range" : {
            "min" : 0,
            "max" : 40,
            "step" : 0.157,
            "unit" : "°C"
          }
        } ]
      }, {
        "key" : "temperatureSetPoint",
        "description" : "Temperature Set point (linear)",
        "values" : [ {
          "range" : {
            "min" : 0,
            "max" : 40,
            "step" : 0.157,
            "unit" : "°C"
          }
        } ]
      } ]
    } ]
  }
}
```


Figure 16: /device/{deviceID}/profile JSON response

3.4.7. GET /devices/{deviceId}/state

This command gets the last state of a device. The state transmission depends on different settings, described in chapter 3.1.5.

GET **http://hostname:port/devices/0185570D/state**


```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "state",
    "gateway" : "interface",
    "timestamp" : "2016-05-09T16:45:28.737+0200"
  },
  "state" : {
    "deviceId" : "0185570D",
    "friendlyId" : "VT5-1_Left",
    "physicalDevice" : "RoomPanel_FTR55D",
    "functions" : [ {
      "key" : "slideSwitch",
      "value" : "pos0DayOn",
      "meaning" : "Slide switch On/Day",
      "timestamp" : "2016-05-09T16:37:57.513+0200",
      "age" : 451224
    }, {
      "key" : "temperature",
      "value" : 24.16,
      "unit" : "°C",
      "timestamp" : "2016-05-09T16:37:57.513+0200",
      "age" : 451225
    }, {
      "key" : "temperatureSetPoint",
      "value" : 18.35,
      "unit" : "°C",
      "timestamp" : "2016-05-09T16:37:57.513+0200",
      "age" : 451225
    } ]
  }
}
```


Figure 17: /devices/{deviceId}/state JSON response

3.4.8. PUT /devices/{deviceId}/state

This request puts/changes the state of a device. If not all keys are modified for one request the EAG shall use the default values, as defined in the profile.

PUT **http://hostname:port/devices/ Hora/state**

```
{
  "state" : {
    "functions" : [
    {
      "key" : "valve",
      "value" : 25
    },
    {
      "key" : "temperatureSetPoint",
      "value" : "23.5"
    },
    {
      "key" : "measurementStatus",
      "value" : "active"
    },
    {
      "key" : "wakeUpCycle",
      "value" : 1
    },
    {
      "key" : "displayOrientation",
      "value" : 0
    },
    {
      "key" : "buttonLockControl",
      "value" : "unlocked"
    },
    {
      "key" : "command",
      "value" : 0
    }
   ]
  }
}
```

Figure 18: /devices/{deviceid}/state outgoing JSON object

3.4.9. GET /devices/{deviceId}/stream?direction={*dir*}&delimited={*del*}&output={*out*}

This command gets the actual status of the device and updates the client of changes via a stream of telegram objects.

Parameter	Datatype	Valid values	Default	functionality
direction	String	from | to | both	Both	Defines which state changes are shown. From the device(from), send from the EAG to the device(to), or both
delimited	String	length | lengthBytes | lengthCharacters | newline |emptyLine	emptyLine	Modifes the delimiter between array entries see chapter 3.1.7.
output	String	formatted | singleLine	formatted	Modifes the formatting between received events see chapter 3.1.7.

Table 12: optional parameters of /device/{deviceid}/stream

GET http://hostname:port/devices/Hora/stream


```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "states",

    "timestamp" : "2015-10-15T09:33:40.760+0200"
  },
  "states" : [ {
    "deviceId" : "0193CC15",
    "friendlyId" : "Hora",
    "functions" : [ {
      "key" : "buttonLockStatus",
      "value" : "unlocked",
      "meaning" : "Unlocked",
      "timestamp" : "2015-10-15T09:29:57.771+0200",
      "age" : "222989"
    }, {
      "key" : "feedTemperature",
      "value" : "20.24",
      "unit" : "°C",
      "timestamp" : "2015-10-15T09:29:57.771+0200",
      "age" : "222989"
    }, {
      "key" : "roomTemperature",
      "value" : "21.02",
      "unit" : "°C",
      "timestamp" : "2015-10-15T09:29:57.771+0200",
      "age" : "222989"
    }, {
      "key" : "valve",
      "value" : "31",
      "unit" : "%",
      "timestamp" : "2015-10-15T09:29:57.771+0200",
      "age" : "222989"
    } ]
  }
}

{
  "header" : {
    "content" : "telegram",
    "timestamp" : "2015-10-15T09:34:34.845+0200"
  },
  "telegram" : {
    "deviceId" : "0193CC15",
    "friendlyId" : "Hora",
    "timestamp" : "2015-10-15T09:34:34.845+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "valve",
      "value" : "48",
      "unit" : "%"
    } ],
    "telegramInfo" : {
      "data" : "30018C08",
      "status" : "0",
      "dbm" : -67,
      "rorg" : "A5"
    }
  }
}
```


Figure 19: /devices/{deviceId}/stream JSON response

3.4.10. GET /devices/{deviceId}/telegrams?direction={*direction*}

This command shows the last available telegrams for the device. The number of telegrams kept in the history may differ between implementations.

Parameter	Datatype	Valid values	Default	Functionality
direction	String	from | to | both	both	Defines which state changes are shown. From the device(from), send from the EAG to the device(to), or both

GET http://hostname:port/0193CC15/telegrams


```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "telegrams",
    "gateway" : "DCGW v0.98.4",
    "timestamp" : "2015-07-16T08:50:39.165+0200"
  },
  "telegrams" : [ {
    "deviceId" : "0193CC15",
    "friendlyId" : "Room3_Heating",
    "physicalDevice" : "Room3_SmartDrive_MX",
    "timestamp" : "2015-07-16T08:12:15.976+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "valve",
      "value" : "0",
      "unit" : "%"
    }, {
      "key" : "temperatureSetPoint",
      "value" : "20.98",
      "unit" : "°C"
    }, {
      "key" : "roomTemperature",
      "value" : "26.16",
      "unit" : "°C"
    }, {
      "key" : "statusRequest",
      "value" : "0",
      "meaning" : "No change"
    }, {
      "key" : "buttonLockStatus",
      "value" : "unlocked",
      "meaning" : "Unlocked"
    }, {
      "key" : "measurementStatus",
      "value" : "active",
      "meaning" : "Measurement enabled"
    } ],
    "telegramInfo" : {
      "data" : "8CCE0A",
      "status" : "0",
      "dbm" : -60,
      "rorg" : "A5"
    }
  }, {
    "deviceId" : "0193CC15",
    "friendlyId" : "Room3_Heating",
```

```
    "physicalDevice" : "Room3_SmartDrive_MX",
    "timestamp" : "2015-07-16T08:18:19.034+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "feedTemperature",
      "value" : "25.88",
      "unit" : "°C"
    }, {
      "key" : "roomTemperature",
      "value" : "26.39",
      "unit" : "°C"
    } ],
    "telegramInfo" : {
      "data" : "19D108",
      "status" : "0",
      "dbm" : -60,
      "rorg" : "A5"
    }
  },
  {
    "deviceId" : "0193CC15",
    "friendlyId" : "Room3_Heating",
    "physicalDevice" : "Room3_SmartDrive_MX",
    "timestamp" : "2015-07-16T08:45:32.884+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "roomTemperature",
      "value" : "26.78",
      "unit" : "°C"
    } ],
    "telegramInfo" : {
      "data" : "19D608",
      "status" : "0",
      "dbm" : -61,
      "rorg" : "A5"
    }
  } ]
}
```


Figure 20: /devices/{deviceID}/telegrams JSON response