

# **STC-IoT Gateway**

## **USER MANUAL**

© 2018 Thermokon  
Thermokon



|  |           |
|--|-----------|
| <b>1. Overview</b>   | <b>7</b>  |
| 1.1 EnOcean API .....  | 9         |
| 1.2 Functionality/Purpose of the Gateway .....                     | 11        |
| <b>2. The Gateway</b>  | <b>13</b> |
| 2.1 Product .....  | 14        |
| 2.2 Quickstart Guide .....   | 15        |
| 2.3 Discovery of the Gateway .....                                 | 25        |
| 2.4 Manual .....   | 26        |
| 2.5 Functions and Profiles .....                                   | 42        |
| 2.6 Learn-In Procedures .....                                      | 51        |
| 2.7 SmartAck vs. Soft Smart Ack vs. 4BS Teach-in Variation 3 ..... | 53        |
| <b>3. The API</b>  | <b>55</b> |
| <b>3.1 API Overview</b> .....                                      | <b>56</b> |
| 3.1.1 API function calls .....                                     | 56        |
| 3.1.2 Last States .....  | 58        |
| 3.1.3 JSON vs. Simple String .....                                 | 59        |
| 3.1.4 Key Values pairs .....                                       | 60        |
| 3.1.4.1 From Gateway to Client .....                               | 61        |
| 3.1.4.2 From Client to Gateway .....                               | 61        |
| 3.1.4.3 JSON description .....                                     | 62        |
| 3.1.4.4 PDF description .....                                      | 68        |
| 3.1.4.5 HTML description .....                                     | 70        |
| <b>3.2 JSON API</b> .....  | <b>71</b> |
| 3.2.1 Security .....   | 72        |
| 3.2.1.1 HTTP Basic Authentication .....                            | 72        |
| 3.2.1.2 TLS Encryption .....                                       | 73        |
| 3.2.2 Communication design .....                                   | 73        |
| 3.2.2.1 Synchronous Communication with Keep-alive .....            | 74        |
| 3.2.2.2 Synchronous Communication with Streaming API .....         | 74        |
| 3.2.2.3 Asynchronous Communication .....                           | 75        |
| 3.2.3 Real Time Communication .....                                | 75        |
| 3.2.3.1 From Gateway to Client: Streaming API .....                | 76        |
| 3.2.3.1.1 Connecting to a streaming Endpoint .....                 | 76        |
| 3.2.3.1.2 Streaming API request parameters .....                   | 77        |
| 3.2.3.1.3 Streaming message types .....                            | 79        |
| 3.2.3.1.4 Processing streaming data .....                          | 80        |
| 3.2.3.2 From Client to Gateway: Persistent Connection .....        | 81        |
| 3.2.3.2.1 Http Keep Alive .....                                    | 81        |
| 3.2.3.3 Streaming REST filter .....                                | 82        |
| 3.2.4 REST Resources .....   | 83        |
| 3.2.4.1 JSON Resource endpoint .....                               | 83        |
| 3.2.4.2 Devices .....  | 85        |

|              |  |            |
|--------------|--|------------|
| 3.2.4.2.1    | GET .....                                  | 85         |
| 3.2.4.2.1.1  | GET /devices .....                         | 86         |
| 3.2.4.2.1.2  | GET /devices?newDevice=true .....          | 88         |
| 3.2.4.2.1.3  | GET /devices/states .....                  | 89         |
| 3.2.4.2.1.4  | GET /devices/stream .....                  | 94         |
| 3.2.4.2.1.5  | GET /devices/telegrams .....               | 136        |
| 3.2.4.2.1.6  | GET /devices/{deviceId} .....              | 145        |
| 3.2.4.2.1.7  | GET /devices/{deviceId}/profile .....      | 151        |
| 3.2.4.2.1.8  | GET /devices/{deviceId}/profile.html ..... | 158        |
| 3.2.4.2.1.9  | GET /devices/{deviceId}/profile.pdf .....  | 158        |
| 3.2.4.2.1.10 | GET /devices/{deviceId}/state .....        | 158        |
| 3.2.4.2.1.11 | GET /devices/{deviceId}/stream .....       | 163        |
| 3.2.4.2.1.12 | GET /devices/{deviceId}/telegrams .....    | 206        |
| 3.2.4.2.2    | PUT .....                                  | 216        |
| 3.2.4.2.2.1  | PUT /devices/{deviceId}/state .....        | 216        |
| 3.2.4.3      | Profiles .....                             | 220        |
| 3.2.4.3.1    | GET .....                                  | 220        |
| 3.2.4.3.1.1  | GET /profiles .....                        | 221        |
| 3.2.4.3.1.2  | GET /profiles/{eepld} .....                | 241        |
| 3.2.4.3.1.3  | GET /profiles/{eepld}.html .....           | 256        |
| 3.2.4.3.1.4  | GET /profiles/{eepld}.pdf .....            | 256        |
| 3.2.4.4      | System .....                               | 256        |
| 3.2.4.4.1    | Systeminfo .....                           | 256        |
| 3.2.5        | HTTP Response Codes .....                  | 259        |
| 3.2.6        | Gateway Response Codes .....               | 260        |
| <b>3.3</b>   | <b>JSON Adminstrator API .....</b>         | <b>264</b> |
| 3.3.1        | Understanding the admin API .....          | 265        |
| 3.3.1.1      | Learning in a new device .....             | 266        |
| 3.3.1.2      | Streaming Admin API vs Streaming API ..... | 267        |
| 3.3.1.3      | Streaming message types .....              | 268        |
| 3.3.1.3.1    | Devices .....                              | 269        |
| 3.3.1.3.2    | Device .....                               | 271        |
| 3.3.1.3.3    | Telegram .....                             | 272        |
| 3.3.2        | REST Resources .....                       | 273        |
| 3.3.2.1      | Devices .....                              | 273        |
| 3.3.2.1.1    | GET .....                                  | 274        |
| 3.3.2.1.1.1  | GET /devices .....                         | 274        |
| 3.3.2.1.1.2  | GET /devices/stream .....                  | 293        |
| 3.3.2.1.1.3  | GET /devices/{deviceId} .....              | 335        |
| 3.3.2.1.1.4  | GET /devices?newDevice=true .....          | 354        |
| 3.3.2.1.1.5  | GET /device/{deviceId}/profile .....       | 372        |
| 3.3.2.1.2    | PUT .....                                  | 386        |
| 3.3.2.1.2.1  | PUT /devices/{deviceId}/test .....         | 386        |
| 3.3.2.1.2.2  | PUT /devices/{deviceId}/learnIn .....      | 388        |
| 3.3.2.1.3    | POST .....                                 | 390        |
| 3.3.2.1.3.1  | POST /devices/{deviceId} .....             | 390        |
| 3.3.2.1.3.2  | Internet of Things Device .....            | 405        |
| 3.3.2.1.3.1  | IoT Label .....                            | 406        |
| 3.3.2.1.3.2  | POST /devices/ (iot) .....                 | 407        |

|              |   |            |
|--------------|---|------------|
| 3.3.2.1.3.3  | QR Code for existing Sensors .....              | 413        |
| 3.3.2.1.3.1  | Create an existing device .....                 | 415        |
| 3.3.2.1.4    | DELETE .....                                    | 418        |
| 3.3.2.1.4.1  | DELETE /devices/{deviceId} .....                | 418        |
| 3.3.2.2      | Profiles .....                                  | 420        |
| 3.3.2.2.1    | GET .....                                       | 420        |
| 3.3.2.2.1.1  | GET /profiles .....                             | 420        |
| 3.3.2.2.1.2  | GET /profiles/{eepld} .....                     | 426        |
| 3.3.2.3      | System .....                                    | 457        |
| 3.3.2.3.1    | GET .....                                       | 457        |
| 3.3.2.3.1.1  | GET /system/receiveMode .....                   | 457        |
| 3.3.2.3.2    | POST .....                                      | 458        |
| 3.3.2.3.2.1  | POST /system/receiveMode .....                  | 459        |
| 3.3.3        | Reference Client Administrator API .....        | 461        |
| <b>3.4</b>   | <b>Simple API .....</b>                         | <b>461</b> |
| 3.4.1        | Security .....                                  | 462        |
| 3.4.1.1      | Basic Authentication .....                      | 462        |
| 3.4.1.2      | TLS Encryption .....                            | 463        |
| 3.4.2        | Communication Design .....                      | 464        |
| 3.4.2.1      | String basics .....                             | 464        |
| 3.4.3        | From JSON profiles to Simple API Commands ..... | 465        |
| 3.4.4        | Messages from Gateway .....                     | 467        |
| 3.4.4.1      | state; .....                                    | 467        |
| 3.4.4.2      | telegram; .....                                 | 469        |
| 3.4.5        | Functions format .....                          | 471        |
| 3.4.5.1      | From Gateway to Client .....                    | 471        |
| 3.4.5.2      | From Client to Gateway .....                    | 472        |
| 3.4.6        | Commands to Gateway .....                       | 473        |
| 3.4.6.1      | date; .....                                     | 474        |
| 3.4.6.2      | devices; .....                                  | 474        |
| 3.4.6.3      | help; .....                                     | 476        |
| 3.4.6.4      | login; .....                                    | 477        |
| 3.4.6.5      | send; .....                                     | 478        |
| 3.4.6.6      | set; .....                                      | 479        |
| 3.4.6.7      | state; .....                                    | 480        |
| 3.4.6.8      | systeminfo; .....                               | 481        |
| 3.4.6.9      | version; .....                                  | 482        |
| <b>4.</b>    | <b>API Connectors/Client Side .....</b>         | <b>485</b> |
| <b>4.1</b>   | <b>Integration Guide .....</b>                  | <b>486</b> |
| <b>4.2</b>   | <b>Examples for sending commands .....</b>      | <b>492</b> |
| 4.2.1        | Permundo actuator PSC 234 .....                 | 492        |
| 4.2.1.1      | JSON API .....                                  | 492        |
| 4.2.1.2      | Simple API .....                                | 494        |
| <b>Index</b> |   | <b>497</b> |



# Overview

## 1 Overview

### Documentation of the Thermokon STC-IoT

- [EnOcean API](#)
- [Functionality/Purpose of the Gateway](#)

Version: V 0.99.21 - December 2016



Thermokon Sensortechnik GmbH



Platanenweg 1  
35756 Mittenaar

[www.thermokon.com](http://www.thermokon.com)



## 1.1 EnOcean API

---

### Motivation

A connectivity of the EnOcean standard to the TCP/IP world is one of the foundations for a successful integration of EnOcean in the emerging Internet of Things. One of the challenges in designing an IP Interface is to resolve the complexity of the EnOcean Equipment Profiles (EEP) in each case so that the presentation on the IP-side is simple and yet consistent in itself. In order to achieve this, it is important to process the content specified in the EnOcean profiles so that little to no knowledge of the EnOcean protocol is needed on the IP side to operate the respective devices.

For the representation of the "things" in the "Internet of Things" (IoT) the Representational State Transfer method (REST and RESTful) has proven to be useful over many similar implementations across and even within the large IoT alliances. That is why we propose such a RESTful API in JSON notation to represent the EnOcean functions over IP.

### General Guidelines

For a meaningful representation of the EnOcean components and functionalities in the IP world, certain requirements must be met. These requirements come out of limitations of the EnOcean specification, where compromises within the protocol have to be made due to energy harvesting, but also from the requirements of control systems, which are connected to a variety of systems over IP. The most important requirements are listed briefly:

- Consistent API

The IP - representation of EnOcean devices across all telegrams and functions must be consistent. This means that the syntax of all existing EEP (2.6.3. , as of September 2015) is displayed in the same way by the API. It should be noted in particular that the attributes, functions and values of the two channels of communication "from" and "to" should be identical. (eg " SetPoint " is always " SetPoint " and not "set point " , " Temp SetPoint " , " SET\_POINT " , etc. ) .

This consistency should also be applied to the transmitted units, which should always be of the same type. I.E. power should always be given out in Watt, never in mW., MW, KW, etc ...

- Persistent representation of the states of all EnOcean devices

The design of the EnOcean protocol doesn't necessarily foresee to ask devices for their current state, as opposed to other bus systems such as KNX. Therefore, an IP representation should keep the state of all devices and deliver it on request. Especially as complex EnOcean profiles hold many different states, individual state storage is an important task.

- Atomic addressing of device functions

The current EnOcean specification requires the transmission of full telegrams between EnOcean participants. With the transition to IP, it should be possible for the IP side to be able to address individual values and functions of a device directly. Thus, assembling and validating of full EnOcean EEPs is a task of the IP Interface. For example a change in a ramp time of an actuator must not force the IP participant to create the entire EnOcean telegram of the actuator.

- Dependency resolution in the EEP  
Complex EnOcean Equipment Profiles often have dependencies in themselves, which must be resolved during the transition to the IP world. Related profile functions which are spread over several databytes and/or telegrams have to be aggregated and delivered on one central position. Databytes with ambiguous meaning for the IP client have to be separated in individual functions. (For example, "If position A = 0, then unit watts; if Position A = 1, then Unit = liters / hour) .
- **Programmable API**  
The IP Interface should be able to deliver the syntax of the communication of all EnOcean devices (all EEP) on request to the IP participant. Thus, it meets the requirement of a "friendly" presentation of the possibilities of EnOcean devices and facilitates its implementation.

## 1.2 Functionality/Purpose of the Gateway

---

### Overall Design

The design goal of the Gateway is to provide an interface to the IP-world, which translates all EnOcean devices into a meaningful, clear and simple description which then can be used by IP-controllers or software to connect EnOcean devices.

The interface provides logic in the sense, that it facilitates the use of EnOcean devices by providing i.e. last states or consistency in the wording of the values and so on. The translation is on EnOcean Equipment Profiles (EEP) level and exposes all functions of all devices through translation of the individual telegrams.

There is no application logic in the gateway that provides "If this than that" functionality. This should be handled by software on a higher level.



# The Gateway

## 2 The Gateway

### Overview

- [The Product](#)
- [Quickstart Guide](#)
- [Manual](#)
- [Learn-in procedures](#)
- [Functions and Profiles](#)

### 2.1 Product

---

#### Overview

The STC-IoT is the first intelligent TCP/IP gateway that is optimized for the market of system integrators.

The design of the commands and the API is focused on a quick understanding and an easy implementation for various control systems. The system integrator can choose between two representations out of the API:

- Simple API: line by line and CSV (comma separated values) based commands over TCP/IP
- REST API: a more powerful JSON representation over HTTP protocol.

The software runs on a compact, robust and energy efficient hardware, equipped with a EnOcean TCM 310 chipset.



The following documentation is provided for the use of the gateway:

- [Quickstart guide](#) A starter when opening the box
- [Manual](#) User manual
- [Learn-in procedures](#) Examples on how to learn in devices
- [Function and profiles](#) All supported functions and profiles

## 2.2 Quickstart Guide

---

# QUICKSTART

## Thermokon STC-IoT





## **Contents**

1. Introduction
2. Functionality
3. Package Contents
4. Requirements
5. Ports
6. Mounting
  - a. Wall mounting
  - b. DIN rail mounting
  - c. Standalone
7. Installation
  - a. Physical Connections
  - b. Connecting to the web interface
  - c. Login
8. Troubleshooting
9. Technical Specification

## 1. Introduction

It is highly recommended to carefully read this quick start guide before using the gateway. Should the gateway be provided to third parties, make sure to also provide this document. For more information about the gateway, related software, as well as the latest documents, visit our website at [www.thermokon.com](http://www.thermokon.com)

### SYMBOLS:



WARNING! Important information regarding danger or risk.



NOTICE. The section contains additional important information.

### HEALTH HAZARDS:



Use the device only for its intended purpose and role. The device is for indoor use only. Avoid exposing the device to humidity, dirt or dust. Avoid direct exposure to sunlight and other heat sources.

Avoid placing the device in metal enclosures. Such placement will affect the gateways ability to communicate with devices. Should the device be placed in an enclosed metal cabinet, make sure to place the antenna outside of the cabinet.

Do not open the device! In event of failure, please contact the Thermokon support.



FAQ's and downloads can be found at: [www.thermokon.com](http://www.thermokon.com)  
In case of technical problems, please contact our support:  
[support@thermokon.de](mailto:support@thermokon.de)



## 2. Functionality

This device is designed to integrate "EnOcean certified components" into an IP-based home control system. Its primary function is to act as a communication bridge between the EnOcean wireless world and IP networks. EnOcean sensors and actuators connected to the gateway can be controlled and accessed by using various interface commands. Registration and administration of the EnOcean components is done via the device's Web interface. A system commission is possible even with limited technical knowledge of EnOcean devices.

## 3. Package Contents

A. 1x Gateway



B. 1x Antenna



## 4. Requirements

- stable and well configured network environment (with a DHCP server/router)
- CAT5 or higher quality network cable for connecting the gateway to a network
- current generation web browser for accessing the web interface (preferably Google Chrome or Mozilla Firefox)

## 5. Ports



- LAN [RJ45] with activity and LEDs
- 24V AC/DC Power Supply
- Antenna 868 MHz

## 6. Mounting



When choosing a mounting location for the device, be careful not to drill near existing electrical switches, sockets or known electrical cable positions.

- a. DIN rail mounting
- b. Standalone

## 7. Installation

### a. Physical Connections

- Screw the antenna cable (B) into the appropriate port on the gateway (A).
- Connect the Gateway (A) with a network cable [Patch Cable RJ45 connector] to your network environment or router.
- Connect the power adapter supplied [C] to the gateway and plug it into a socket.

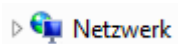
### b. Connecting to the web interface

- Start your computer
- Run your Web-Browser and enter [stc-iot] or [stc-iot.local] in the address bar.



If a connection to the web interface cannot be established, check the network cable connection and make sure your computer is connected to the same network. Alternatively, you can look up the IP address of the gateway from the network device list on your router or DHCP server.

- In addition to the Web-Browser, the device can be accessed via Windows Explorer (Windows Button+E)



- Open Networks

*See navigation panel (section on the left)*

- Double Click on the network device

### c. Login




- Enter your previously configured password (default password: admin)




## 8. Troubleshooting

- If the device is not responding, disconnect the power supply for 20 seconds, then plug the device back in.
- If the gateway is still not responding, please check all cable connections to the device. Verify that the network activity LEDs display normal operation.
- If EnOcean devices are not working reliably, try moving the gateway to another location or eliminate sources of potential RF interference.
- For more information, please visit: [www.thermokon.com](http://www.thermokon.com)
- If you encounter technical problems, please contact our support at Thermokon
- [support@thermokon.de](mailto:support@thermokon.de)

### 9. Technical Specifications

| Technical Specifications  |   |
|---|---|
| Device model name:  | Thermokon STC-IoT   |
| Power supply  | 15..24 V DC / 24V AC (+-10%)  |
| Dimensions (HxWxD):   | 60 x 95 x 107 mm  |
| Electric consumption (max):   | 3,0 W   |
| Connections:  | 1x LAN(RJ45), 1x Antenna connector  |
| RF band:  | 868MHz  |
| Range:  | Without obstacles up to 30m   |
| Chipset:  | TCM 310 / EnOcean Chipset – bidirectional   |
| Features  |   |
| Setup-GUI   | Web-Interface   |
| Import/Export of Configuration  | Yes   |
| Database  | MySQL (out of the Box)  |
| API (dual)  | Simple String, Restful JSON based Commands  |
| We reserve the right to make changes to the technical specifications                |   |
| Symbols   |   |
|  | The conformity of the product with the applicable EC directives.  |
| <b>RoHS</b>   | Tested in accordance with EU directive 2011/65 / EU - "Restriction of certain Hazardous Substances" - restriction of individual hazardous substances.   |
|  | WEEE Directive (Waste Electrical and Electronic Equipment)<br>According to the European Directive 2002/96 / EC and 2012/19 / EU<br>Disposal of the device is not in household waste but at an appropriate electronic waste collection point.<br>WEEE-Reg.-Nr. DE 83788620 |
|  | Device is to be used for the designated purpose only.<br>Device is to only be used with the provided power supply.  |
| <b>Usage</b>  |   |



|   |   |
|---|---|
|   | The device is only to be used in the designated areas of application:<br>Application: Interior; Storage and Use: Dry  |
|  | Tested in accordance with EMC Directive EN 61000<br>EN 61000-6-1 (Immunity to interferences Residential, ...) (Immunity for industrial environments)<br>EN 61000-6-3 (interference emission living area, ...)<br>EN 61000-6-4 (Interference emission industrial environments) |

## 2.3 Discovery of the Gateway

### Overview

There are several possibilities to discover the gateway once it has been powered up:

- UPNP
- Bonjour
- DHCP Server Entries

### Description

#### UPNP

Discovering the Gateway via UPNP is easy if you have windows as operating system and a network environment where UPNP broadcasts are allowed.

Just go to the windows network in the explorer and try to locate the gateway under the group "other devices".

The entry consists of the name followed by the IP Address. A double click will open the browser and take you to the login screen.

If you right click on the properties, you get additional information.

**Note:** In many industrial environments, UPNP broadcasts can be blocked by switches or routers. Please consult the local administrator in this case.

#### Bonjour

To discover the Gateway via Bonjour you need an MacOS or iOS operating system and a Hardware environment where bonjour broadcasts are allowed.

Just go to your browser and type the default name "STC-IoT" followed by the extension ".local". So "STC-IoT.local" should take you directly to the login screen (without quotations).

**Note:** In many industrial environments, Bonjour broadcasts can be blocked by switches or routers. Please consult the local administrator in this case.

### DHCP Server entries

If you are able to look at the tables of the local DHCP server, you should find the gateway easily with a name of STC-IoT.<local domain>

## 2.4 Manual

---

# IP EnOcean Gateway

## Thermokon STC-IoT

# Manual

### Contents

---

[1. General information](#)

[2. The Webinterface](#)

[2.1 Menu point "Admin"](#)

[2.2 Menu point "EnOcean"](#)

[2.3 Menu point "Internet of Things"](#)



## 1. General information

English edition 10/2017 Documentation © 2017 Thermokon Sensortechnik GmbH, Germany

All rights reserved. This manual may not be reproduced in part in any form or duplicated using electronic, mechanical or chemical processes or processed without the written permission of the publisher.

It is possible that this manual has printing technology defects or typographical errors. However, the specifications of this manual are regularly reviewed and corrections are made in the next issue. For technical or printing errors and their consequences Thermokon Sensortechnik GmbH assumes no liability.

Before using the Gateway, please read these instructions carefully. If you pass the product on to third parties, give them this manual, too.

More information, associated software, as well as the latest version of the manual can be found on our website [www.thermokon.com](http://www.thermokon.com)

This handbook should be seen as an extension of the Quick Start guide "Quick Start" which Thermokon already attaches to your product. It describes in detail, both the properties of the gateway, the menu structure, as well as the processes that are necessary for start-up and subsequent operation.

For installation instructions and basic information about the physical start-up, please look at our "Quick Start" !



Technical notes [FAQs] and downloads are available on our homepage: [www.thermokon.de](http://www.thermokon.de)

If you have technical problems or comments, please directly contact support from Thermokon Sensortechnik GmbH.

E-mail: [support@thermokon.com](mailto:support@thermokon.com)

Hazard statements:



- Do not use for purposes other than the intended use.
- Use only indoors and avoid external influences such as moisture, dirt/dust and sunlight or other heat load.
- Avoid repositories enclosed by metal, as these affect the transmission power. If a mount in DIN rail is to be made,

precautions should be taken to mount the antenna outside the control cabinet.

- Do not open the device. In case of failure, please contact the Thermokon support team.

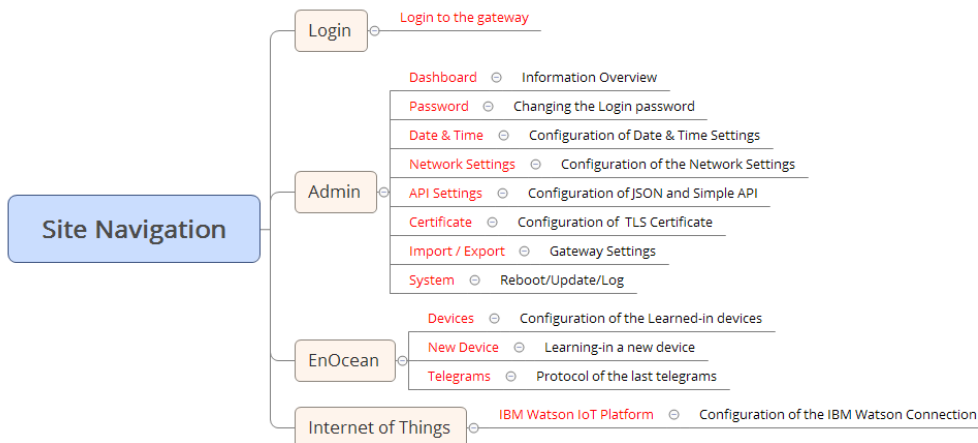
## 2. The webinterface

The web interface is primarily used to:

- commission and configure the gateway
- Register and configure the EnOcean components
- Monitor the transmitted states (telegrams)
- Analyze errors in the system

### The menu structure

For easy navigation, the complete menu structure of the gateway is shown here. The manual is based on this order and explains the individual areas .



## Menu item "Login" - registration at the gateway.

Discovering the gateway is described [here](#)



If a physical connection is established, you should see the login page of the gateway.

You have to enter a password. The default password is "admin". Please change it afterwards.

## Menu item “Admin“

All administrative settings and values which are fundamentally important for the gateway ? can be configured here.

### System Info - General Info System

After a successful login the system information page is shown.



As the name implies, you can view basic information here regarding the current state of your gateway.

**Version:** actual software release of the gateway

- *Do you have the latest firmware? – see section: Firmware*

**State & Date:** actual time & clock settings

- *Possibilities for adjusting can be found under the menu item "Date & Time"*

**Uptime:** Uptime since last reboot

**Load:** Load of the Gateway. As the hardware consists of a multiprocessor, there can be a load of above 100%.

**LAN Interface:** The actual network configuration of the gateway

- *Possibilities for adjusting can be found under the menu item "Ethernet"*

**Port Settings:** Current port setting for the two API versions with password

- *Possibilities for adjusting can be found under the menu item „API Port Settings“*
- *Notes on the API can be found in the "API Port Settings" menu or in the corresponding API descriptions in the download section of the Gateway website.*

**EnOcean Chip:** Information on the current status of the EnOcean chipset.

**Base ID:** Base ID of the gateway

**Chip ID:** Serial number of the EnOcean chipset

**Filter Mode:** When enabling the filter mode, only telegrams of learned-in devices are transmitted though the gateway on the API and on the Telegram menu. Default is off. This is used to limit the output of the gateway only to the desired devices.

- *Possibilities for adjusting can be found under the menu item „EnOcean Chip“*

**Number of devices:** Number of currently learned-in devices.

### Password – Changing the password

In order to protect the gateway from unauthorized access it is strongly recommended to change the factory-set password [factory setting: admin]. **Make sure that your password is not too complex. It is not possible at this stage of development of the gateway to reset the password.**



## **Date & Time – Configuration of date & time**

In this menu the appropriate time settings of the gateway can be made. The gateway provides the ability to sync via the Internet service NTP. To do this, enter your favorite NTP server. Alternatively, the time can also be entered manually. However, this can lead to variations in the time after a certain period of operation.

## **Ethernet – Network setting of the gateway**

In this menu you can adjust the Gateway IP settings according to your environment. By default the gateway is in DHCP mode (Dynamic Host Configuration Protocol), in which the gateway gets its IP-settings automatically transmitted from the server/router. If you prefer manual adjustment you can change the mode from DHCP to Static and enter your desired values.

*ServiceIP: Besides the above-mentioned possibilities, the gateway also provides a ServiceIP. This IP address cannot be changed and is for the purpose of support. This IP address is: 192.168.66.69.*

## **EnOcean Chip**

Each gateway includes an EnOcean communication chip with an individual chip ID. This chip ID is similar to a serial number and unchangeable.

API Settings – Port configuration of Simple API & JSON API



This menu allows the ports and passwords for both APIs to be set individually. There are several ways you can check the output of the gateway of each API:



#### *Simple String API*

Using the freeware tool "PuTTY" and the configuration of the Gateway API, it is easy to connect to the Simple API and look at the output.

Start the software, define the IP as required and choose „telnet“ as connection type. After setting the port to the corresponding Simple API port (default 9090), you can start the connection.

In the window shown you can see the connection startig with the actual version of the gateway. Press return and login with the desired user. (Default password: user)

*login; password=user*

Now you can see all the incoming and outgoing telegrams – one command per line.

#### *Restful JSON API*

The default port for the JSON API is <IP of gateway: 8080>. Here the gateway shows all the features that it offers with an easy to understand interface. The full range of functions is displayed and the syntax of the calls may be tested at any time extensively.

To use the JSON API, point your browser to the address <IP of gateway: 8080> and log in with as a user with the appropriate password. You now have reached the Help page of the JSON API. The coding of the JSON API is "UTF-8".

### Menu Item "Import/Export" – Backup and restore of configurations



- **Export of the Gateway configuration**  
The complete configuration and settings of the gateway can be saved in a configuration file "dcgw.conf". Press the "Export" button and select the destination folder according to your browser.  
The following information is saved:
  - Network settings
  - EnOcean Settings (API settings)
  - Device Database (Sensors and actuators)

- **Import of the Gateway configuration**

To load a configuration file into the gateway, please drag the configuration file (e.g. stciot.conf) to the field "Upload" (drag'n'drop). Three possible sections show up for you to choose what part to import.

## Menu Item System – How to update the firmware



The Thermokon Gateway will be improved over time with new profiles and new functionalities.

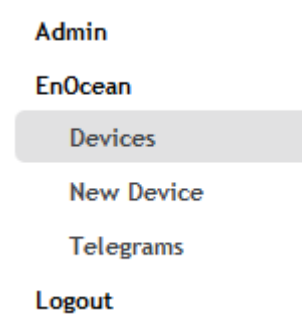
These improvements (new firmware) will be available on the homepage [www.thermokon.de](http://www.thermokon.de) in the support section and will have to be uploaded in the firmware menu of the gateway.

As described in the menu "Import/Export" please drag and drop the new firmware to the field "Upload"

The gateway will check the file before starting an update process. Nothing will happen if the file is not suitable.

## Menu item “EnOcean”

This menu contains management options for connected EnOcean devices. The options include learning-in (pairing) of new devices, editing features and communication settings for existing devices, and deleting sensors and actuators from device database.



## Menu Item "Devices" – Overview of registered EnOcean devices

Upon selecting the Devices menu, a table of all registered devices is displayed. The following managing options are available:



Action



**EDIT – Editing properties of a learned-in device** – edit device name and description, switch between API versions and select which telegrams are to be accepted and translated by the gateway. Changes of Device ID and profile version are not possible, as these values are set during the Learn-in procedure. In order to change Profile or Device type, the device needs to be deleted and learned-in again.



**DELETE – Remove device from database** – Deletes a device from the gateway’s database. Use if device is no longer used or if fresh learn-procedure is needed.

Last Seen ↻

Indicates time of the last valid telegram reception from this device.

### Device Id

Unique identification number assigned to device – 8 character hexadecimal address (range: 00000000 – FFFFFFFF) assigned by manufacturer to each produced device. Hardware based and not changeable.

### Friendly Id and Name

User definable fields, assignable to each device. Friendly Id and name have to be unique. Assign meaningful and easily recognised names to devices in order to simplify managing and grouping of devices.

### Friendly Id

Name of EnOcean product

### Name

Description of device and usage details

Example: Friendly ID - Thermokon XYZ  
Name - Light switch bedroom

### Last State

Last received and recognized device state. e.g. Contact open / Contact closed, measure of temperature and humidity.



Use the arrow button to sort data in ascending or descending order for selected values

---

Admin

EnOcean

Devices

New Device

Telegrams

Logout

## Menu Item "New Device" – Registering new EnOcean devices



Learning-in of new devices can vary in complexity depending on manufacturer and device type. For example, to learn-in a rocker button, a normal button press is enough to complete the learn-in procedure. For other devices, learn-in mode can be triggered by a magnet, pressing the special button or some other way of learning-in, as described by device's manual.

Two basic learn-in methods are available:

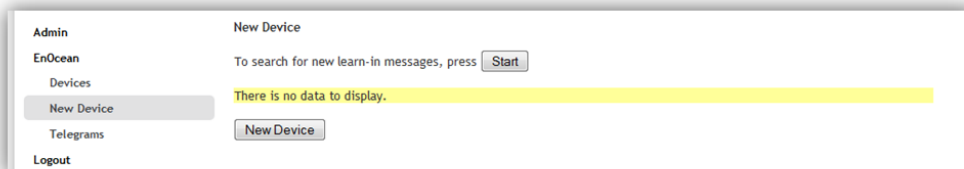
1. detect device automatically
2. add device manually

For the user, this offers both learning-in of well known existing devices and unknown or experimental/prototype devices. As long as communication parameters and protocol details are known, almost all devices can be learned-in and added into the gateway database.

### *Learn-in via automatic device detection*

Press the "Start" button to set gateway into "search mode" in order for the gateway to automatically recognize new device. The gateway will search for new devices for 60 seconds.

*In the following example, a rocker switch module PTM 215 (4 push buttons, 2 rockers) is being learned-in:*



EnOcean -> New Device -> press "Start" – Newly discovered devices will be shown upon detecting.

Gateway now recognizes and identifies new device by its unique Device Id and EnOcean Profile range (F6-??-??). Explanation and further information about EnOcean profiles can be found at [https://www.enocean-alliance.org/de/enOcean\\_standard/](https://www.enocean-alliance.org/de/enOcean_standard/).

It is important to note that new a device is recognized and displayed, however it is still not added into the gateway's database.



Press "EDIT" – Continue the learn-in procedure by selecting the "EDIT" button.



Under the tab “General”, device specific parameters and fields must be filled-in before a device can be saved to the gateway’s SQL database.

Tabs “Communication” and “Test” will be explained later in the manual, as they are not critical for the completion of a learn-in procedure.

Depending on the device, more or less additional information must be entered before proceeding. The following parameters must be entered:

- **Device Id:** Unique device identifier received during the learn-in procedure. Can often be found printed on the device itself.
- **Friendly Id:** Contains description of device’s function.  
*In our example “Rocker switch”*
- **Name of physical device:** Device name  
*In our example, “Light switch Living room”*
- **Location:** Enter the location of the device
- **Manufacturer:** Enter the manufacturer’s name, if known
- **Learn-in Procedure:** Shows the learn-in method (device specific, automatically recognized)
  - o receivedRps – Device has been detected and recognized automatically
  - o manuallyConfigured – Device is added manually
- **EnOcean Equipment Profile** – The correct device profile must be selected.  
*In our example: F6-02-01: Rocker Switch, 2 Rocker, Light and Blind Control*
- **EEP Variation:** Used in special cases when the manufacturer does not comply with the standard EEP rules.
- **API Version:** Currently only version 1.0 is available. As new API features are introduced and implemented, older versions will stay available in order to retain compatibility.
- **Transmission modes:**

Every EnOcean device sends and receives data as described by the device’s EnOcean Equipment Profile. Reported values and states can be filtered by selecting or deselecting checkboxes. When a value checkbox is selected, the gateway will report the value to connected TCP/IP clients. When not selected, the gateway will not translate this particular type of message. It is worth noting that devices can use multiple EEP profiles, therefore they have to be learned in twice or more times. The following transmission mode options are available:

- **On Connect**  
Last known value will be reported to a TCP/IP client upon establishing connection. Immediately after a client connects, all known EnOcean values will be transmitted. Together with the last known state, age of data (in milliseconds) and UTC timestamp will be reported. After the initial data stream has been transmitted, the gateway will start reporting incoming data as it is received.

**Note:** If none of the devices in the database have this attribute enabled, translation of events will start immediately after the TCP/IP connection has been established.

**Default Setting:** On

- **On Event**  
Incoming EnOcean values are translated and sent to a connected TCP/IP client as soon as they are received. This checkbox is helpful when filtering which incoming values should be processed. Devices often lack sensors for predefined EEP values, e.g. only humidity sensor in device that uses humidity & temperature profile. In this case, the transmission of the temperature value can be disabled.

**Default Setting:** On

- **On Duplicate**  
The gateway will transmit received values to a TCP/IP client, even if device values are unchanged in comparison to the last reported value. This happens when devices report values in a cyclic pattern. Heating controllers will often report values once every 10 min, and then drop back to deep standby mode. Disabling this option will eliminate unnecessary API traffic.

In many cases the result of the EnOcean Energy Harvesting scheme is the cyclic transmission of data. In the majority of cases, it is enough to transmit a value only after the value has been changed. Alternatively, the gateway can be asked at any time to report the last known value. An exception might happen when dealing with safety related functions, as lack of transmitted information might falsely indicate a malfunction or defect. If this is the case, the TransmitOnDuplicate setting can be enabled manually.

**Default Setting:** Off

After all settings have been set, there are two saving options to choose from:

1. **Apply** – Settings are saved and EnOcean device is added to gateway's database. Configuration page stays open.
2. **OK** – Settings are saved and EnOcean device is added to gateway's database. Configuration page is closed and web browser is redirected back to "Devices" menu.

New device is now registered on gateway and visible in the list of devices. Subsequent changes are possible via the "Edit" button.

#### *Adding a device manually*

As already mentioned, the complexity of the learn-in procedure depends greatly on the device that is being added. Some devices are not compatible with the automatic learn-in process. Therefore, they need to be added manually. During the learn-in of a simple button (see "Learn-in via automatic device detection"), the only direction of communication i.e. telegram transmission, is from device to gateway. Some devices cannot complete the learn-in procedure without receiving a confirmation telegram from the gateway, a so called "welcome message". Thus, before learning-in, the gateway must be correctly configured in order to



transmit the correct “welcome” telegram. That requires some detailed information about your device to be entered during the learn-in procedure.

- To start the manual learn-in procedure, navigate to EnOcean section, New Device menu. Under the button which starts automatic device learn-in, there is a “New device” button, which leads to the manual learn-in menu [*EnOcean->New Device->New Device*].



The device’s details and parameters must be entered here. In contrast to the automatic learn-in procedure, all device data, including the Device ID, must be manually specified.

After entering the correct device profile and all required data, press “Apply” in order to save the device into the gateway’s database.

**Admin**

**EnOcean**

Devices

New Device

Telegrams

**Logout**

### Menu Item "Telegrams" – Overview of telegram traffic

This page contains an overview of all transmitted and received messages between the gateway and the devices currently learned-in (devices in database). User can switch data view between the original code (*raw*) and analyzed/processed representation of same code (*parsed*).



### Menu Item "Internet of Things"

Under this menu point, all available Internet of Things connectors are shown and can be enabled/disabled.

### Logout

Ends current management session.

## Legend



ATTENTION! Important information, crucial to device's functionality.



NOTE. Paragraph contains additional or extended information.

## 2.5 Functions and Profiles

---

Every profile listed here is implemented and can be used. The profiles were implemented and often made easier on IP side, so integrators may have a better understanding of their meaning. Please do not rely on the used names in the official specification but check against the JSON API instead.

After you set up and run the STC-IoT successfully, you might check our website for system updates under [www.thermokon.com](http://www.thermokon.com) in Download-Center.

The firmware update file has a file ending of “.cua” and needs to be selected during the update process on the STC-IoT Web Interface. If you need help for firmware update please have a look to our documentations.

If you are trying to use a device/EEP which is not supported in the current version, please let us know so we will prioritize this and include it in the next firmware release.

Please report any issue you encounter, or tell us which missing functionality you would expect in the future.

### EnOcean Functions

*Supported EnOcean Functionality*

- ESP3, ERP 3
- Manufacturer modified Profiles
- 4BS Teach-in Variation 3
- Universal Teach In
- Smart Ack
- Soft Smart-Ack

*Not yet Supported EnOcean Functionality (as of Version 1.0)*

- EnOcean Security
- Remote Management
- Remote Commissioning

## EnOcean Equipment Profiles

All supported EnOcean Equipment Profiles (as of 09.12.2015) are shown below. STC-IoT is able to distinguish between genuine EEPs and manufacturer modified profiles. Therefore a profile may reveal a device model and manufacturer in one of the variation column in the table below.

You also can find the latest list of supported EEPs, whenever you make a HTTP request to:

`http://<IP-Address of the gateway>:8080/profiles`

| ee title                             | Variation1 | Variation2 | Variation3 |
|--------------------------------------|------------|------------|------------|
| p                                    |            |            |            |
| A5- Temperature Sensors, Temperature |            |            |            |
| 02- Sensor Range -40°C to 0°C        |            |            |            |
| 01                                   |            |            |            |
| A5- Temperature Sensors, Temperature |            |            |            |
| 02- Sensor Range -30°C to +10°C      |            |            |            |
| 02                                   |            |            |            |
| A5- Temperature Sensors, Temperature |            |            |            |
| 02- Sensor Range -20°C to +20°C      |            |            |            |
| 03                                   |            |            |            |
| A5- Temperature Sensors, Temperature |            |            |            |
| 02- Sensor Range -10°C to +30°C      |            |            |            |
| 04                                   |            |            |            |
| A5- Temperature Sensors, Temperature |            |            |            |
| 02- Sensor Range 0°C to +40°C        |            |            |            |
| 05                                   |            |            |            |
| A5- Temperature Sensors, Temperature |            |            |            |
| 02- Sensor Range +10°C to +50°C      |            |            |            |
| 06                                   |            |            |            |

- A5- Temperature Sensors, Temperature  
02- Sensor Range +20°C to +60°C  
07
- A5- Temperature Sensors, Temperature  
02- Sensor Range +30°C to +70°C  
08
- A5- Temperature Sensors, Temperature  
02- Sensor Range +40°C to +80°C  
09
- A5- Temperature Sensors, Temperature  
02- Sensor Range +50°C to +90°C  
0A
- A5- Temperature Sensors, Temperature  
02- Sensor Range +60°C to +100°C  
0B
- A5- Temperature Sensors, Temperature  
02- Sensor Range -60°C to +20°C  
10
- A5- Temperature Sensors, Temperature  
02- Sensor Range -50°C to +30°C  
11
- A5- Temperature Sensors, Temperature  
02- Sensor Range -40°C to +40°C  
12
- A5- Temperature Sensors, Temperature  
02- Sensor Range -30°C to +50°C  
13
- A5- Temperature Sensors, Temperature  
02- Sensor Range -20°C to +60°C  
14
- A5- Temperature Sensors, Temperature  
02- Sensor Range -10°C to +70°C  
15
- A5- Temperature Sensors, Temperature  
02- Sensor Range 0°C to +80°C  
16
- A5- Temperature Sensors, Temperature  
02- Sensor Range +10°C to +90°C  
17
- A5- Temperature Sensors, Temperature  
02- Sensor Range +20°C to +100°C  
18
- A5- Temperature Sensors, Temperature  
02- Sensor Range +30°C to +110°C  
19
- A5- Temperature Sensors, Temperature  
02- Sensor Range +40°C to +120°C  
1A
- A5- Temperature Sensors, Temperature  
02- Sensor Range +50°C to +130°C  
1B

- A5- Temperature Sensors, 10 Bit
- 02- Temperature Sensor Range -10°C to 20 +41.2°C
- A5- Temperature Sensors, 10 Bit
- 02- Temperature Sensor Range -40°C to 30 +62.3°C
- A5- Temperature and Humidity Sensor, alphaEOS\_SEN
- 04- Range 0°C to +40°C and 0% to 100% SETF-H
- 01
- A5- Temperature and Humidity Sensor, Eltako\_FAFT60
- 04- Range -20°C to +60°C and 0% to 100% ,FIFT65S,FBH6
- 02 5TFB
- A5- Temperature and Humidity Sensor,
- 04- Range -20°C to +60°C 10bit-
- 03 measurement and 0% to 100%
- A5- Barometric Sensor, Range 500 to
- 05- 1150 hPa
- 01
- A5- Light Sensor, Range 300lx to 60.000lx Eltako\_FAH60,
- 06- FAH60B,FAH65
- 01 S,FIH65S
- A5- Light Sensor, Range 0lx to 1.020lx Eltako\_FIH65B
- 06-
- 02
- A5- Light Sensor, 10-bit measurement (1-
- 06- Lux resolution) with range 0lx to
- 03 1000lx
- A5- Occupancy Sensor, Occupancy with
- 07- Supply voltage monitor
- 01
- A5- Occupancy Sensor, Occupancy with
- 07- Supply voltage monitor
- 02
- A5- Occupancy Sensor, Occupancy with
- 07- Supply voltage monitor and 10-bit
- 03 illumination measurement
- A5- Light, Temperature and Occupancy Eltako\_FABH6
- 08- Sensor, Range 0lx to 510lx, 0°C to 5S,FBH65B,FB
- 01 +51°C and Occupancy Button H65S,FBH65TF
- B
- A5- Light, Temperature and Occupancy
- 08- Sensor, Range 0lx to 1020lx, 0°C to
- 02 +51°C and Occupancy Button
- A5- Light, Temperature and Occupancy
- 08- Sensor, Range 0lx to 1530lx, -30°C to
- 03 +50°C and Occupancy Button
- A5- Gas Sensor, CO-Sensor 0 ppm to 1020
- 09- ppm
- 02
- A5- Gas Sensor, CO2 Sensor
- 09-

- 04
- A5- Gas Sensor, VOC Sensor
- 09-
- 05
- A5- Gas Sensor, Radon
- 09-
- 06
- A5- Gas Sensor, Particles
- 09-
- 07
- A5- Gas Sensor, Pure CO2 Sensor
- 09-
- 08
- A5- Gas Sensor, Pure CO2 Sensor with
- 09- Power Failure Detection
- 09
- A5- Gas Sensor, Hydrogen Gas Sensor
- 09-
- 0A
- A5- Room Operating Panel, Temperature
- 10- Sensor, Set Point, Fan Speed and
- 01 Occupancy Control
- A5- Room Operating Panel, Temperature
- 10- Sensor, Set Point, Fan Speed and
- 02 Day/Night Control
- A5- Room Operating Panel, Temperature Afriso\_RoomT Thermokon\_
- 10- Sensor, Set Point Control                      emperatureSe SR06LCD
- 03    nsorFT
- A5- Room Operating Panel, Temperature
- 10- Sensor, Set Point and Fan Speed
- 04 Control
- A5- Room Operating Panel, Temperature
- 10- Sensor, Set Point and Occupancy
- 05 Control
- A5- Room Operating Panel, Temperature Eltako\_FTR55D
- 10- Sensor, Set Point and Day/Night
- 06 Control
- A5- Room Operating Panel, Temperature
- 10- Sensor, Fan Speed Control
- 07
- A5- Room Operating Panel, Temperature
- 10- Sensor, Fan Speed and Occupancy
- 08 Control
- A5- Room Operating Panel, Temperature
- 10- Sensor, Fan Speed and Day/Night
- 09 Control
- A5- Room Operating Panel, Temperature
- 10- Sensor, Set Point Adjust and Single
- 0A Input Contact
- A5- Room Operating Panel, Temperature
- 10- Sensor and Single Input Contact

0B

A5- Room Operating Panel, Temperature  
10- Sensor and Occupancy Control

0C

A5- Room Operating Panel, Temperature  
10- Sensor and Day/Night Control

0D

A5- Room Operating Panel, Temperature  
10- and Humidity Sensor, Set Point and  
10 Occupancy Control

A5- Room Operating Panel, Temperature  
10- and Humidity Sensor, Set Point and  
11 Day/Night Control

A5- Room Operating Panel, Temperature Afriso\_RoomT  
10- and Humidity Sensor and Set Point emperatureSe  
12 nsorFTF

A5- Room Operating Panel, Temperature  
10- and Humidity Sensor, Occupancy  
13 Control

A5- Room Operating Panel, Temperature  
10- and Humidity Sensor, Day/Night  
14 Control

A5- Room Operating Panel, 10 Bit  
10- Temperature Sensor, 6 bit Set Point  
15 Control

A5- Room Operating Panel, 10 Bit  
10- Temperature Sensor, 6 bit Set Point  
16 Control;Occupancy Control

A5- Room Operating Panel, 10 Bit  
10- Temperature Sensor, Occupancy  
17 Control

A5- Room Operating Panel, Illumination,  
10- Temperature Set Point, Temperature  
18 Sensor, Fan Speed and Occupancy  
Control

A5- Room Operating Panel, Humidity,  
10- Temperature Set Point, Temperature  
19 Sensor, Fan Speed and Occupancy  
Control

A5- Room Operating Panel, Supply  
10- voltage monitor, Temperature Set  
1A Point, Temperature Sensor, Fan  
Speed and Occupancy Control

A5- Room Operating Panel, Supply  
10- Voltage Monitor, Illumination,  
1B Temperature Sensor, Fan Speed and  
Occupancy Control

A5- Room Operating Panel, Illumination,  
10- Illumination Set Point, Temperature  
1C Sensor, Fan Speed and Occupancy  
Control

- A5- Room Operating Panel, Humidity,  
10- Humidity Set Point, Temperature
- 1D Sensor, Fan Speed and Occupancy  
Control
- A5- Room Operating Panel, Humidity,  
10- Humidity Set Point, Temperature
- 1E Sensor, Fan Speed and Occupancy  
Control
- A5- Room Operating Panel, Temperature
- 10- Sensor, Set Point, Fan Speed,  
1F Occupancy and Unoccupancy Control
- A5- Room Operating Panel, Temperature  
10- and Set Point with Special Heating
- 20 States
- A5- Room Operating Panel,  
10- Temperature, Humidity and Set  
21 Point with Special Heating States
- A5- Controller Status, Lighting Controller  
11-  
01
- A5- Controller Status, Blind Status  
11-  
03
- A5- Automated Meter Reading (AMR),  
12- Counter  
00
- A5- Automated Meter Reading (AMR), Eltako\_FSS12,FPressac\_CTCl  
12- Electricity WZ12,FWZ61 amp  
01
- A5- Automated Meter Reading (AMR),  
12- Gas  
02
- A5- Automated Meter Reading (AMR),  
12- Water  
03
- A5- Environmental Applications, Sun  
13- Intensity  
01
- A5- Environmental Applications, Sun  
13- Intensity  
02
- A5- Environmental Applications, Date  
13- Exchange  
03
- A5- Environmental Applications, Time  
13- and Day Exchange  
04
- A5- Environmental Applications,  
13- Direction Exchange  
05



A5- Environmental Applications,  
 13- Geographic Position Exchange  
 06  
 A5- Environmental Applications, Sun  
 13- position and radiation  
 10  
 A5- Multi-Func Sensor, Single Input  
 14- Contact (Window/Door), Supply  
 01 voltage monitor  
 A5- Multi-Func Sensor, Single Input  
 14- Contact (Window/Door), Supply  
 02 voltage monitor and Illumination  
 A5- Multi-Func Sensor, Single Input  
 14- Contact (Window/Door), Supply  
 03 voltage monitor and Vibration  
 A5- Multi-Func Sensor, Single Input  
 14- Contact (Window/Door), Supply  
 04 voltage monitor, Vibration and  
 Illumination  
 A5- Multi-Func Sensor, Vibration/Tilt,  
 14- Supply voltage monitor  
 05  
 A5- Multi-Func Sensor, Vibration/Tilt,  
 14- Illumination and Supply voltage  
 06 monitor  
 A5- HVAC Components, Battery Powered  
 20- Actuator  
 01  
 A5- HVAC Components, Basic Actuator  
 20-  
 02  
 A5- HVAC Components, Line powered  
 20- Actuator  
 03  
 A5- HVAC Components, Heating Radiator  
 20- Valve Actuating Drive with Feed and  
 04 Room Temperature Measurement,  
 Local Set Point Control and Display  
 A5- Digital Input, Single Input Contact,  
 30- Battery Monitor  
 01  
 A5- Digital Input, Single Input Contact  
 30-  
 02  
 A5- Digital Input, 4 Digital Inputs, Wake  
 30- and Temperature  
 03  
 A5- Digital Input, 3 Digital Inputs, 1  
 30- Digital Input 8 Bits  
 04

- A5- Central Command, Gateway
- 38-
- 08
- D2- Electronic switches and dimmers
- 01- with Energy Measurement and Local
- 00 Control
- D2- Electronic switches and dimmers
- 01- with Energy Measurement and Local
- 01 Control
- D2- Electronic switches and dimmers
- 01- with Energy Measurement and Local
- 02 Control
- D2- Electronic switches and dimmers
- 01- with Energy Measurement and Local
- 03 Control
- D2- Electronic switches and dimmers
- 01- with Energy Measurement and Local
- 04 Control
- D2- Electronic switches and dimmers
- 01- with Energy Measurement and Local
- 05 Control
- D2- Electronic switches and dimmers
- 01- with Energy Measurement and Local
- 06 Control
- D2- Electronic switches and dimmers
- 01- with Energy Measurement and Local
- 07 Control
- D2- Electronic switches and dimmers
- 01- with Energy Measurement and Local
- 08 Control
- D2- Electronic switches and dimmers Permundo\_ps
- 01- with Energy Measurement and Local c234
- 09 Control
- D2- Electronic switches and dimmers Nodon\_Smart
- 01- with Energy Measurement and Local PlugASP-2-1-
- 0A Control 10
- D2- Electronic switches and dimmers
- 01- with Energy Measurement and Local
- 0B Control
- D2- Electronic switches and dimmers
- 01- with Energy Measurement and Local
- 10 Control
- D2- Electronic switches and dimmers
- 01- with Energy Measurement and Local
- 11 Control
- D2- Blinds Control for Position and Angle
- 05-
- 00
- D2- Multisensor Window Handle, Alarm,
- 06- Position Sensor, Vacation Mode,
- 01 Optional Sensors

|  |            |                          |
|--|------------|--------------------------|
| D2- Standard Valve, Valve Control        |            |                          |
| A0-01                                    |            |                          |
| D5- Contacts and Switches, Single Input  |            |                          |
| 00- Contact                              |            |                          |
| 01                                       |            |                          |
| F6- Rocker Switch, 2 Rocker, Light and   | Eltako_FRW | VariousMan               |
| 02- Blind Control - Application Style 1  |            | VariousManuf             |
| 01                                       |            | ufacturers_Gacturers_Smo |
|  |            | eneralSwitchkeDetector   |
|  |            | Actuator                 |
| F6- Rocker Switch, 2 Rocker, Light and   | Eltako_FRW | VariousMan               |
| 02- Blind Control - Application Style 10 |            | ufacturers_G             |
| 02                                       |            | eneralSwitch             |
|  |            | Actuator                 |
| F6- Rocker Switch, 2 Rocker, Light and   | Eltako_FRW | VariousMan               |
| 02- Blind Control - Application Style 11 |            | ufacturers_G             |
| 03                                       |            | eneralSwitch             |
|  |            | Actuator                 |
| F6- Rocker Switch, 4 Rocker, Light and   |            |                          |
| 03- Blind Control - Application Style 1  |            |                          |
| 01                                       |            |                          |
| F6- Detectors, Liquid Leakage Sensor     |            |                          |
| 05- (mechanic harvester)                 |            |                          |
| 01                                       |            |                          |
| F6- Mechanical Handle, Window Handle     |            |                          |
| 10-00                                    |            |                          |

## 2.6 Learn-In Procedures

---

### Learn-In procedures

The gateway supports all types of Learn-In, though it can be confusing with the different procedures. Learning in a device into the Gateway is a two step procedure.

#### First Step:

The Device and the Gateway get to know each other. This is done by sending Learn-in telegrams from and to the two parties. In a later version of the gateway, there will be more options here like the ability to use Remote Management/Remote Commissioning or by scanning a QR-Code from an App.

The devices appear in the Web Interface under the menu "New Devices", have internally the state "operable=false" and are not yet in the internal database. Once the device passes the second step, it will be saved in the database.

### Second Step:

The user has to fill out at least the FriendlyName via the Web interface; devices based on RORG F6 (e.g. Window Handles or Liquid detectors) require additional information about the Profile (F6-XX-XX) or even the DeviceId in case of a manual Learn-In. After providing the necessary information, the device is stored in the database by pressing OK or Apply. It is now "operable=true" and will from this moment on work over the APIs.

For an overview, the following methods are supported:

- Learn-In telegram initiated by the device - Pairing finished  
This is the most common learn-in procedure. You put the gateway in learn-in mode and initiate a learn-in telegram to be sent out from the device. In this case, the following learn-ins are available:
  - receivedRps (F6)
  - received1Bs (D5)
  - received4Bs (A5)
  - 4BS Teach-in Variation 3
  - Universal Teach-In (UTE)
  - Smart-Acknowledgment (Smart-Ack)
- Learn-In telegram initiated by the device - Pairing has to be completed manually  
This is the learn-in procedure for bidirectional devices that do not use neither UTE nor Smart-Ack nor 4BS Teach-in Variation 3. After having received the Learn-In from the device, you have to manually send out a learn-in to the device. This can be done via the tab "communication" in the Web-Interface. This is as for now the case for:
  - receivedRps (F6)
  - received4Bs (A5)
- Complete manual Learn-In - Pairing has to be completed manually  
This is the least best option to learn-in devices. It is basically the same as option Nr. 2, but you do not receive a Learn-In telegram from the device. Instead you enter the deviceId, EEP and the FriendlyName by hand in the appropriate fields and continue with the manual pairing from step 2. This is as for now the case for:

- receivedRps (F6)

For examples on how to Learn-In different devices please refer to

[Learn-in procedure on the website www.thermokon.de](http://www.thermokon.de)

## 2.7 SmartAck vs. Soft Smart Ack vs. 4BS Teach-in Variation 3

---

### Smart Ack

Smart Ack is a functionality in the EnOcean protocol that allows energy harvesting devices to fall asleep (stop listening on the radio) to preserve energy. Smart Ack turns the conventional request/response dialog (the gateway requests something and the device responds) around and allows the device to wake up at any time, sends a request to the gateway which then answers within a defined short time period before the device falls asleep again.

This functionality is negotiated during the Learn-in and can be seen, whenever a device has the Learn-in procedure "Smart-Ack" when editing a device in the Web Interface.

Smart Ack is handled by the EnOcean Chip (thus in hardware) and from the point of view of the gateway, it receives a command from the API, forwards it to the EnOcean chip which then takes care of delivering the message whenever the device gives notice.

### Soft Smart Ack

There are devices, which behave like Smart Ack Devices without using the Smart Ack Specification thus without using the functionality provided by the hardware.

These devices require an answer from the gateway after having sent out a telegram within a certain (mostly 1.000 ms) amount of time. This application layer functionality is provided by the gateway in the way that:

- A command that comes in over the API is answered with a http response code 202 - The telegram will be delivered after the device wakes up and sends something. (Device uses Soft Smart Ack)"

Example Answer:

```
{
  "header" : {
    "httpStatus" : 202,
    "code" : 1002,
    "message" : "will send telegram later",
    "gateway" : "STC-IoT v0.99.0c",
    "timestamp" : "2016-05-27T17:55:27.529+0200"
```

```
}  
}
```

- The telegram is stored by the gateway
- If another (second, third,...) command arrives over the API while a message is still on hold, the gateway will replace the waiting telegram with the last received command (last in -> out)
- When the next telegram from the device arrives, the stored telegram is immediately sent out. The sending out of the telegram is reported over the APIs in the moment of sending.

Known devices requiring Soft Smart Ack:

- Thermokon SAB05 / SAB+
- Kieback&Peter, MD10-FTL-HE, MD15-FTL-HE
- Micropelt, iTRV
- Soda GmbH, Recent Window Handle

### 4BS Teach-in Variation 3

4BS Teach-in Variation 3 can be seen as equivalent to the functionality of UTE for the Learn-In process. The device sends out a 4BS Learn-in which is received by the gateway and awaits a 4BS Learn-In back within a certain (mostly 1.000 ms) amount of time.

This application layer functionality is provided by the gateway in the way that when the Gateway receives a 4BS Teach-In for bidirectional profiles, it assumes that it is the Teach-in Variation 3 and replies immediately with an equivalent Teach-In back, thus resulting in fully learned-in devices.

Profiles requiring 4BS Teach-in Variation 3:

- A5-20-XX                      HVAC Components
- A5-38-08                      Central Command Gateway

# The API

## 3 The API

### Overview

This is the main chapter of the documentation. It's all about the API.

- [API Overview](#)
- [JSON API](#)
- [JSON Administrator API](#)
- [Simple API](#)

## 3.1 API Overview

---

### Overview

- [API Function Calls](#)
- [Last States](#)
- [JSON vs. Simple String](#)
- [RESTFUL Key-Value pairs](#)

### 3.1.1 API function calls

#### API functions calls

The API is the central Interface for the IP clients to connect to EnOcean devices. When designing the API, one should choose an implementation that is based on common modern standards which offers a solid basis for future EnOcean devices as well as for the transition to the Internet of Things.

A RESTful specification of the IP Interface enables many IP clients on the REST API to operate in parallel, each of which may have set his individual preferences with the Interface.

The connection between IP - Client and the IP-Interface is established via HTTP requests which are divided into five main functional groups:

- [Profile functions](#)

Profile functions allow a query of the Interface regarding profiles (EEP) and their functions (representation) without corresponding devices available or taught in.
- [Device functions](#)



Device functions are used to get an overview of the taught-in devices of the Interface. This includes the list of devices, their current states and the query for each device, which functions it enables and how to address them. These device functions are similar to the profile functions, except that they relate to real devices and their interaction.

The device functions also include functions that allow you to manage, teach-in and delete the devices into the Interface. These functions must be especially secured, because they reflect a base functionality.

- Transmission functions

The transmission functions are the main features for the regular operation of the Interface.

#### [PUT](#)

With the PUT function, commands are transmitted to the Interface which should be executed in the EnOcean world.

#### [GET](#)

The GET function enables the client to be notified of changes on the EnOcean side. In order to transfer events of EnOcean devices in real-time to the connected IP clients, a permanent connection must be kept open between the Interface and the respective client. This is achieved in the REST API via a streaming functionality where the gateway as well as the client do not close their connection on a certain HTTP-Endpoint. ([Streaming, chunked transfer encoding](#))

- [System functions](#)

The system functions provide direct control of the Interface. These include querying the system information and the switching of the EnOcean chip to learn-in or normal mode either.
- Help functions

The Interface should show a clear and comprehensive overview of all available functions on the REST API at root level. The codepage to be used is be „UTF-8“

### 3.1.2 Last States

#### Last States

This API function is used regulate the output of the API according to device states when using streaming API. A state is a set of key values describing the last known state of a device. For each key, API streaming output can be influenced by three different attributes (TransmitOnConnect, TransmitOnEvent or TransmitOnDuplicate). Influencing means, whether a state or telegram will be transmitted or not when connecting or streaming.

- TransmitOnConnect

If this attribute is set, API streaming output transmits this key-value pair among others when a client connects to the streaming API. This information is used to retrieve a database of values of every single learned-in EnOcean device to initialize device values in third party systems, while concurrently start streaming device telegrams. Each value contains an age (a value in milliseconds) as well as an UTC timestamp. After delivering the “states”, the Interface atomically starts with transferring events from the EnOcean radio.

**Default for all devices:** On

- TransmitOnEvent

If this attribute is set, the value is being transmitted via API streaming output. Activate this attribute for every key that won't be used in your system. For example, a device uses an EEP (e.g. a5-04-01) that forsee the transmission of temperature and humidity, but your system doesn't require temperature information, or the device has no measuring unit included. This value of “Temp 0 degree” can be suppressed on API streaming output.

**Default for all devices:** On

- TransmitOnDuplicate

If this attribute is set, then the Interface even transmits received values of devices, whose value that has not been changed since last transmission. For example, a window sensor periodically transmits every 10 minutes its "closed" state. If TransmitOnDuplicate is not set, then the Interface suppresses the output of the API. If set, the Interface transmits this key-value, although there is no change in the information.

Since in many cases, the cyclical sending of equal value is a tribute to the EnOcean protocol in terms of energy harvesting, it is sufficient to transmit in most cases a value only when it changes. Alternatively, the Interface can be asked at any time for a value of a device via the API. An exception may exist on mapping of security-related functions, in which it can be seen as a malfunction or a defect when a value is not repeated.

**Default for all devices:** Off

### 3.1.3 JSON vs. Simple String

#### JSON vs. Simple String

The decision whether you use the JSON API or the Simple API depends on the Client (Controller) you are using and on the programming environment you have.

Both APIs provide the same functionality once the devices are learned in and you use the gateway in normal working mode. However, The JSON API is more powerful but also more complex when it comes to asking the gateway about profiles or connecting directly to a stream of one dedicated device. For a best practice approach to the design of the connection, please refer to the chapter "[Client programming](#)".

Comparison between the JSON and the Simple String API

|                            | JSON API     | Simple String API                                      |
|----------------------------|--------------|--|
| <b>Application</b>         | HTTP/HTTPS   | TCP Socket Connection                                  |
| <b>Security</b>            | TLS 1.2      | TLS 1.2  |
| <b>Content Description</b> | JSON Strings | CSV Strings with delimiter "Carriage Return Line Feed" |

|                          | JSON API  | Simple String API  |
|--------------------------|---|--|
| <b>Encoding</b>          | UTF-8   | ISO-8859-1:1998 (Latin-1, West Europe)   |
| <b>Connection Design</b> | Two parallel permanent connections; (receive/send)  | one full duplex connection   |
| <b>Functionality</b>     | <ul style="list-style-type: none"> <li>• Working Commands</li> <li>• Retrieve Profile Information</li> </ul>  | <ul style="list-style-type: none"> <li>• Working Commands</li> </ul>   |
| <b>Target Clients</b>    | IP Clients with full HTTP Stack and "JSON" libraries <ul style="list-style-type: none"> <li>• Windows</li> <li>• Unix</li> <li>• MacOS</li> <li>• etc.</li> </ul> | Industrial Controllers i.e. with TCP/IP Stack <ul style="list-style-type: none"> <li>• Siemens</li> <li>• Crestron</li> <li>• AMX</li> <li>• WAGO</li> <li>• Beckhoff</li> <li>• Loxone</li> <li>• etc...</li> </ul> |

Both APIs do not have any limit in the number of parallel connections and are up and running at the same time.

### 3.1.4 Key Values pairs

#### Overview

A profile consists of a set of key value pairs to transport/deliver a state. The gateway includes profiles in a clean way with detailed description for each key value pair. The information is given in 3 different representations that can be used by users or machines either. Users usually take PDF or HTML representation to understand a profile and get a overview about the Key-Value pairs that are used to control the devices. Machines can use JSON output that is represented in a machine-parsable text.

- [JSON Output](#)
- [PDF Output](#)
- [HTML Output](#)

3.1.4.1 From Gateway to Client

**Key/Value Pairs used in the Communication "from" and "both"**

These are properties used by the gateway to specify a profile using the API.

**Description**

| String  | Meaning                         | Optional | Type  | Comment                         |
|---------|---------------------------------|----------|---|---------------------------------|
| key     | The name of the function / data | No       | string  |                                 |
| value   | Corresponding value of the key  | No       | string / float  |                                 |
| meaning | Meaning of the value            | Yes      | string  |                                 |
| range   | Defining the range of values    | Yes      | Object<br>property:<br><br>min : integ<br>max : integ<br>step: float<br>unit: strin | property: unit is not mandatory |

3.1.4.2 From Client to Gateway

**Key/Value Pairs used in the Communication "to" and "both"**

These are the properties a client need to set in order to command devices (change state) over the API.

**Description**

| String | Meaning                  | Optional | Type   | Comment |
|--------|--------------------------|----------|--------|---------|
| key    | The name of the function | No       | string |         |

|       |                                |    |                 |  |
|-------|--------------------------------|----|-----------------|--|
| value | Corresponding value of the key | No | float or string |  |
|-------|--------------------------------|----|-----------------|--|

When a key has a "default" value set, ([visible in JSON profile description, pdf-view](#) or [html-view](#)) then the key/value pair can be left out of the telegram. The gateway will fill the missing Key/Value pairs with their indicated default values.

### 3.1.4.3 JSON description

#### PDF-description of a profiles Key-Value pairs.

Sending a GET /profiles/{eepid} request at the gateway will return a JSON description, where all key-value pairs are listed for the full communication with the device.

Example: [http://{hostname:api\\_port}/profiles/D2-01-09](http://{hostname:api_port}/profiles/D2-01-09) will return a JSON as follows:

```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "profile",
    "gateway" : "STC-IoT v0.99.0",
    "timestamp" : "2016-04-29T15:00:04.571+0200"
  },
  "profile" : {
    "eep" : "D2-01-09",
    "title" : "Electronic switches and dimmers with Energy Measurement and Local Control",
    "functionGroups" : [ {
      "title" : "Actuator Set Output",
      "direction" : "to",
      "functions" : [ {
        "key" : "dimValue",
        "values" : [ {
          "meaning" : "Output value 0% to 100%",
          "range" : {
            "min" : 0,
            "max" : 100,
            "step" : 1,
            "unit" : "%"
          }
        }
      ]
    }
  ]
}, {
  "key" : "rampingMode",
  "values" : [ {
```

```

        "value" : "rampingTime1",
        "meaning" : "Dim to new output value using rampingTime1"
    }, {
        "value" : "rampingTime2",
        "meaning" : "Dim to new output value using rampingTime2"
    }, {
        "value" : "rampingTime3",
        "meaning" : "Dim to new output value using rampingTime3"
    }, {
        "value" : "stop",
        "meaning" : "Stop dimming"
    }, {
        "value" : "switch",
        "meaning" : "Switch to new output value"
    } ],
    "defaultValue" : "switch"
} ]
}, {
    "title" : "Configure Actuator",
    "direction" : "to",
    "functions" : [ {
        "key" : "defaultState",
        "values" : [ {
            "value" : "off",
            "meaning" : "Default state: 0% or OFF"
        }, {
            "value" : "on",
            "meaning" : "Default state: 100% or ON"
        }, {
            "value" : "previousState",
            "meaning" : "Default state: remember previous state"
        } ],
        "defaultValue" : "previousState"
    }, {
        "key" : "overcurrentSwitchOffReset",
        "values" : [ {
            "value" : "false",
            "meaning" : "Reset over current shut down: not active"
        }, {
            "value" : "true",
            "meaning" : "Reset over current shut down: trigger signal"
        } ],
        "defaultValue" : "false"
    }, {
        "key" : "rampingTime1",
        "values" : [ {
            "meaning" : "Dim timer 1",
            "range" : {
                "min" : 0.5,
                "max" : 7.5,
                "step" : 0.5,
                "unit" : "s"
            }
        } ],
        "defaultValue" : 1
    }, {
        "key" : "rampingTime2",

```

```

    "values" : [ {
      "meaning" : "Dim timer 2",
      "range" : {
        "min" : 0.5,
        "max" : 7.5,
        "step" : 0.5,
        "unit" : "s"
      }
    } ],
    "defaultValue" : 4
  }, {
    "key" : "rampingTime3",
    "values" : [ {
      "meaning" : "Dim timer 3",
      "range" : {
        "min" : 0.5,
        "max" : 7.5,
        "step" : 0.5,
        "unit" : "s"
      }
    } ],
    "defaultValue" : 7.5
  }, {
    "key" : "taughtInDevices",
    "values" : [ {
      "value" : "off",
      "meaning" : "Disable taught-in devices (with different EEPROM)"
    }, {
      "value" : "on",
      "meaning" : "Enable taught-in devices (with different EEPROM)"
    } ],
    "defaultValue" : "on"
  } ]
}, {
  "title" : "Actuator Status Response",
  "direction" : "from",
  "functions" : [ {
    "key" : "dimValue",
    "values" : [ {
      "meaning" : "Output value 0% to 100%",
      "range" : {
        "min" : 0,
        "max" : 100,
        "step" : 1,
        "unit" : "%"
      }
    } ],
    {
      "value" : "invalid",
      "meaning" : "Output value not valid / not applicable"
    } ]
  }, {
    "key" : "errorLevel",
    "values" : [ {
      "value" : "failure",
      "meaning" : "Error level 2: hardware failure"
    } ],
    {
      "value" : "noError",

```



```

        "meaning" : "Error level 0: hardware OK"
    }, {
        "value" : "notSupported",
        "meaning" : "Error level not supported"
    }, {
        "value" : "warning",
        "meaning" : "Error level 1: hardware warning"
    } ]
}, {
    "key" : "localControl",
    "values" : [ {
        "value" : "off",
        "meaning" : "Local control disabled / not supported"
    }, {
        "value" : "on",
        "meaning" : "Local control enabled"
    } ]
}, {
    "key" : "overcurrentSwitchOff",
    "values" : [ {
        "value" : "false",
        "meaning" : "Over current switch off: ready / not supported"
    }, {
        "value" : "true",
        "meaning" : "Over current switch off: executed"
    } ]
} ]
}, {
    "title" : "Actuator Set Measurement",
    "direction" : "to",
    "functions" : [ {
        "key" : "energyDelta",
        "description" : "Delta of energy to be reported",
        "values" : [ {
            "range" : {
                "min" : 0,
                "max" : 4095000,
                "step" : 0.000278,
                "unit" : "Wh"
            }
        } ]
    } ],
    "key" : "maxTimeBetweenReports",
    "description" : "Measurement Response messages",
    "values" : [ {
        "range" : {
            "min" : 10,
            "max" : 2550,
            "step" : 10,
            "unit" : "s"
        }
    } ],
    "defaultValue" : 60
}, {
    "key" : "minTimeBetweenReports",
    "description" : "Measurement Response messages",
    "values" : [ {

```

```

        "range" : {
            "min" : 0,
            "max" : 255,
            "step" : 1,
            "unit" : "s"
        }
    } ],
    "defaultValue" : 10
}, {
    "key" : "reportMeasurement",
    "values" : [ {
        "value" : "queryAndAuto",
        "meaning" : "Report measurement: query / auto reporting"
    }, {
        "value" : "queryOnly",
        "meaning" : "Report measurement: query only"
    } ],
    "defaultValue" : "queryAndAuto"
}, {
    "key" : "resetMeasurement",
    "values" : [ {
        "value" : "false",
        "meaning" : "Reset measurement: not active"
    }, {
        "value" : "true",
        "meaning" : "Reset measurement: trigger signal"
    } ],
    "defaultValue" : "false"
} ]
}, {
    "title" : "Actuator Set Measurement",
    "direction" : "to",
    "functions" : [ {
        "key" : "maxTimeBetweenReports",
        "description" : "Measurement Response messages",
        "values" : [ {
            "range" : {
                "min" : 10,
                "max" : 2550,
                "step" : 10,
                "unit" : "s"
            }
        } ],
        "defaultValue" : 60
    }, {
        "key" : "minTimeBetweenReports",
        "description" : "Measurement Response messages",
        "values" : [ {
            "range" : {
                "min" : 0,
                "max" : 255,
                "step" : 1,
                "unit" : "s"
            }
        } ],
        "defaultValue" : 10
    }, {

```

```

    "key" : "powerDelta",
    "description" : "Delta of power to be reported",
    "values" : [ {
      "range" : {
        "min" : 0,
        "max" : 4095000,
        "step" : 1,
        "unit" : "W"
      }
    } ]
  }, {
    "key" : "reportMeasurement",
    "values" : [ {
      "value" : "queryAndAuto",
      "meaning" : "Report measurement: query / auto reporting"
    }, {
      "value" : "queryOnly",
      "meaning" : "Report measurement: query only"
    } ],
    "defaultValue" : "queryAndAuto"
  }, {
    "key" : "resetMeasurement",
    "values" : [ {
      "value" : "false",
      "meaning" : "Reset measurement: not active"
    }, {
      "value" : "true",
      "meaning" : "Reset measurement: trigger signal"
    } ],
    "defaultValue" : "false"
  } ]
}, {
  "title" : "Actuator Query",
  "direction" : "to",
  "functions" : [ {
    "key" : "query",
    "values" : [ {
      "value" : "energy",
      "meaning" : "Query energy"
    }, {
      "value" : "power",
      "meaning" : "Query power"
    }, {
      "value" : "status",
      "meaning" : "Query status"
    } ]
  } ]
}, {
  "title" : "Actuator Measurement Response",
  "direction" : "from",
  "functions" : [ {
    "key" : "energy",
    "description" : "Cumulative electricity value from meter",
    "values" : [ {
      "range" : {
        "min" : 0,
        "max" : 4294967295000,

```

```
        "step" : 0.000278,  
        "unit" : "Wh"  
      }  
    } ]  
  }, {  
    "key" : "power",  
    "description" : "Current power value from meter",  
    "values" : [ {  
      "range" : {  
        "min" : 0,  
        "max" : 4294967295000,  
        "step" : 1,  
        "unit" : "W"  
      }  
    } ]  
  } ]  
} ]  
}
```

#### 3.1.4.4 PDF description

##### PDF-description of a profiles Key-Value pairs.

Appending ".pdf" to the URI will return a pdf, where all key-value pairs are listed for the full communication with the device.

Example: [http://{hostname:api\\_port}/profiles/D2-01-09.pdf](http://{hostname:api_port}/profiles/D2-01-09.pdf) will return a pdf as follows:

## D2-01-09: Electronic switches and dimmers with Energy Measurement and Local Control

## Direction "to": Actuator Set Output

| Key         | Value                 | Meaning                                    |
|-------------|-----------------------|--|
| dimValue    | 0 ... 100 [%], step 1 | Output value 0% to 100%                    |
| rampingMode | rampingTime1          | Dim to new output value using rampingTime1 |
|             | rampingTime2          | Dim to new output value using rampingTime2 |
|             | rampingTime3          | Dim to new output value using rampingTime3 |
|             | stop                  | Stop dimming                               |
|             | switch (default)      | Switch to new output value                 |

## Direction "to": Configure Actuator

| Key                       | Value                                   | Meaning   |
|---------------------------|---|---|
| defaultState              | off                                     | Default state: 0% or OFF                          |
|                           | on                                      | Default state: 100% or ON                         |
|                           | previousState (default)                 | Default state: remember previous state            |
| overcurrentSwitchOffReset | false (default)                         | Reset over current shut down: not active          |
|                           | true                                    | Reset over current shut down: trigger signal      |
| rampingTime1              | 0.5 ... 7.5 [s], step 0.5 (default 1)   | Dim timer 1                                       |
| rampingTime2              | 0.5 ... 7.5 [s], step 0.5 (default 4)   | Dim timer 2                                       |
| rampingTime3              | 0.5 ... 7.5 [s], step 0.5 (default 7.5) | Dim timer 3                                       |
| taughtInDevices           | off                                     | Disable taught-in devices (with different EEPROM) |
|                           | on (default)                            | Enable taught-in devices (with different EEPROM)  |

## Direction "from": Actuator Status Response

| Key                  | Value                 | Meaning  |
|----------------------|-----------------------|--|
| dimValue             | 0 ... 100 [%], step 1 | Output value 0% to 100%                        |
|                      | invalid               | Output value not valid / not applicable        |
| errorLevel           | failure               | Error level 2: hardware failure                |
|                      | noError               | Error level 0: hardware OK                     |
|                      | notSupported          | Error level not supported                      |
|                      | warning               | Error level 1: hardware warning                |
| localControl         | off                   | Local control disabled / not supported         |
|                      | on                    | Local control enabled                          |
| overcurrentSwitchOff | false                 | Over current switch off: ready / not supported |
|                      | true                  | Over current switch off: executed              |

**D2-01-09: Electronic switches and dimmers with Energy Measurement and Local Control****Direction "to": Actuator Set Measurement**

| Key                   | Value                                 | Meaning  |
|-----------------------|---------------------------------------|--|
| energyDelta           | 0 ... 4095000 [Wh], step 0.000278     | Delta of energy to be reported   |
| maxTimeBetweenReports | 10 ... 2550 [s], step 10 (default 60) | Measurement Response messages  |
| minTimeBetweenReports | 0 ... 255 [s], step 1 (default 10)    | Measurement Response messages  |
| reportMeasurement     | queryAndAuto (default)<br>queryOnly   | Report measurement: query / auto reporting<br>Report measurement: query only |
| resetMeasurement      | false (default)<br>true               | Reset measurement: not active<br>Reset measurement: trigger signal           |

**Direction "to": Actuator Set Measurement**

| Key                   | Value                                 | Meaning  |
|-----------------------|---------------------------------------|--|
| maxTimeBetweenReports | 10 ... 2550 [s], step 10 (default 60) | Measurement Response messages  |
| minTimeBetweenReports | 0 ... 255 [s], step 1 (default 10)    | Measurement Response messages  |
| powerDelta            | 0 ... 4095000 [W], step 1             | Delta of power to be reported  |
| reportMeasurement     | queryAndAuto (default)<br>queryOnly   | Report measurement: query / auto reporting<br>Report measurement: query only |
| resetMeasurement      | false (default)<br>true               | Reset measurement: not active<br>Reset measurement: trigger signal           |

**Direction "to": Actuator Query**

| Key   | Value                     | Meaning                                     |
|-------|---------------------------|---|
| query | energy<br>power<br>status | Query energy<br>Query power<br>Query status |

**Direction "from": Actuator Measurement Response**

| Key    | Value                                      | Meaning                                 |
|--------|--|---|
| energy | 0 ... 4294967295000 [Wh],<br>step 0.000278 | Cumulative electricity value from meter |
| power  | 0 ... 4294967295000 [W], step 1            | Current power value from meter          |

**3.1.4.5 HTML description****HTML-description of a profiles Key-Value pairs.**

Appending ".html" to the URI will return a html page, where all key-value pairs are listed for the full communication with the device.

Example: [http://{hostname:api\\_port}/profiles/D2-01-09.html](http://{hostname:api_port}/profiles/D2-01-09.html) will return a html file as follows:

**D2-01-09: Electronic switches and dimmers with Energy Measurement and Local Control**

EnOcean over IP

**Direction "to": Actuator Set Output**

| Key         | Value                 | Meaning                                    |
|-------------|-----------------------|--|
| dimValue    | 0 ... 100 [%], step 1 | Output value 0% to 100%                    |
| rampingMode | rampingTime1          | Dim to new output value using rampingTime1 |
|             | rampingTime2          | Dim to new output value using rampingTime2 |
|             | rampingTime3          | Dim to new output value using rampingTime3 |
|             | stop                  | Stop dimming                               |
|             | switch (default)      | Switch to new output value                 |

**Direction "to": Configure Actuator**

| Key                       | Value                                   | Meaning  |
|---------------------------|---|--|
| defaultState              | off                                     | Default state: 0% or OFF                       |
|                           | on                                      | Default state: 100% or ON                      |
|                           | previousState (default)                 | Default state: remember previous state         |
| overcurrentSwitchOffReset | false (default)                         | Reset over current shut down: not active       |
|                           | true                                    | Reset over current shut down: trigger signal   |
| rampingTime1              | 0.5 ... 7.5 [s], step 0.5 (default 1)   | Dim timer 1                                    |
| rampingTime2              | 0.5 ... 7.5 [s], step 0.5 (default 4)   | Dim timer 2                                    |
| rampingTime3              | 0.5 ... 7.5 [s], step 0.5 (default 7.5) | Dim timer 3                                    |
| taughtInDevices           | off                                     | Disable taught-in devices (with different EEP) |
|                           | on (default)                            | Enable taught-in devices (with different EEP)  |

**Direction "from": Actuator Status Response**

| Key        | Value                 | Meaning                                 |
|------------|-----------------------|---|
| dimValue   | 0 ... 100 [%], step 1 | Output value 0% to 100%                 |
|            | invalid               | Output value not valid / not applicable |
| errorLevel | failure               | Error level 2: hardware failure         |

## 3.2 JSON API

### Overview

The JSON API is the main API for talking to the gateway

- Security
- Communication design

- Real Time Communication
- REST Resources
- HTTP Response Codes
- Application Response Codes

## Codeset

The codeset used by the JSON API is: UTF-8

### 3.2.1 Security

## Authentication and Encryption

The JSON API has two mechanisms to secure the IP communication.

- [HTTP Basic Authentication](#)
- [TLS Encryption](#)

The EnOcean Radio Security is not yet integrated due to the fact that there are very few devices on the market, that support this feature at the moment. We will implement it in an upcoming release update.

#### 3.2.1.1 HTTP Basic Authentication

## JSON API password or JSON API port

To access the JSON API, the client must authenticate itself via [Basic access authentication](#).

Default credentials: Username: "user" password: "user".

Default password and port can be changed via the webinterface.

Menu item "Admin" / API Port Settings.

Admin

- System Info
- Password
- Date & Time
- Network Settings
- EnOcean Chip
- API Port Settings**
- Certificate
- Import / Export
- Firmware
- EnOcean
- Logout

API Port Settings

Simple API Port:

Simple API Password:  [generate a new one](#)

Simple API Encryption:  Off  On

JSON API Port:

JSON API Password for login "user":  [generate a new one](#)

JSON API Encryption:  Off  On

*Changes will result in lost of last states*



### 3.2.1.2 TLS Encryption

#### Certificate

The gateway provides a self signed certificate for the TLS encryption.

The certificate consists of:

- default name of the gateway "STC-IoT" - CN=STC-IoT
- name of the organization - O=TK
- and the Country Name - C=DE

Example for Thermokon



These values are generated out of your environment and will vary.

In case you change the name of the gateway (network settings), you also need to generate a new certificate.

#### Activation of the encryption

The encryption for both APIs can be activated or deactivated via the webinterface.

Menu item Admin / API Port Settings

### 3.2.2 Communication design

#### Communication design of the JSON API

There are three different communication interactions with the gateway.

- [Synchronous Communication](#)
- [Synchronous Communication with Streaming API](#)
- [Asynchronous Communication](#)

### 3.2.2.1 Synchronous Communication with Keep-alive

#### Synchronous requests

For client requests which do not involve telegrams being sent out, the gateway will directly return a full JSON answer in the body of the message.

This is valid for

- GET
  - [/systeminfo](#)
  - [/profiles](#)
  - [/profiles/{eepld}?variation={variationId}](#)
  - [/devices](#)
  - [/devices/states](#)
  - [/devices/telegrams?direction=from|to|both](#)
  - [/devices/{deviceId}](#)
  - [/devices/{deviceId}/profile](#)
  - [/devices/{deviceId}/state](#)
  - [/devices/{deviceId}/telegrams?direction=from|to|both](#)



### 3.2.2.2 Synchronous Communication with Streaming API

#### Synchronous requests with streaming API

For client requests to the streaming API, a never ending stream of events represented by individual JSON objects is established.

After establishing the connection, the gateway will

1. send out the last states of all devices, which have the flag [transmitOnConnect](#) set.
2. start immediately with the stream of events as they occur
3. *Gateway will never close this connection*

This is valid for

- GET
  - [/devices/stream?direction=from|to|both](#)
  - [/devices/{deviceId}/stream?direction=from|to|both](#)



### 3.2.2.3 Asynchronous Communication

#### Asynchronous Requests

The handling of incoming and outgoing requests is by design asynchronous for all calls which are related to EnOcean telegrams being sent out by the gateway.

This is due to the EnOcean Communication protocol where some devices don't confirm the reception of a telegram and other devices are not always listening to the radio (i.e. they sleep for a certain period of time). So the time between sending out a request to a EnOcean device and receiving an answer is not predictable.

This means, that when a client sends an action request to the gateway because it wants to change the state of a device (PUT /devices/{deviceid}/state), the gateway answers in general with OK which means that the request was transmitted correctly from the client to the gateway and the gateway was able to send out the command over the air. The answer from the device will not come over the same channel (or HTTP Response) but will be actively sent to all clients connected to streaming API.

This is valid for

- PUT
  - [/devices/{deviceid}/state](#)



### 3.2.3 Real Time Communication

#### Overview

Real time connection to the gateway is the most important part to get instant notification of changes in the EnOcean World.

#### GET

GET /devices/stream is the HTTP URI to get notifications of events that occur on EnOcean radio. A [streaming connection](#) will be established.

#### PUT

PUT /devices/{id}/state is the HTTP URI to [trigger an action](#) on the EnOcean side. A normal HTTP Request is being made.

### 3.2.3.1 From Gateway to Client: Streaming API

#### Overview

The Streaming API give developers low latency access to the Gateways stream of information. Once connected, the gateway will start pushing messages indicating telegrams or other notifications that have been occurred. No need for the client to constantly poll REST resources.

#### Differences between Streaming and REST

Connecting to the streaming API requires keeping a persistent HTTP connection open. In many cases this involves thinking about your application differently than if you were interacting with the REST API.

You normally start the streaming connection in a separate process/thread, because this function is actively responsible to receive asynchronous messages and forward this to your main thread accordingly.

#### 3.2.3.1.1 Connecting to a streaming Endpoint

##### Overview

Establishing a connection to the streaming APIs means long polling a certain HTTP resource (/devices/stream), and parsing the response incrementally. Conceptually, you can think of it as downloading an infinitely long text file over HTTP.

##### Connecting

To connect to the streaming API, form a HTTP request and consume the resulting stream for as long as is practical. The Gateway will keep the connection open indefinitely, barring server-side error, excessive client-side lag or network hiccups.

The method to form an HTTP request and parse the response will be different for every language or framework, so consult the documentation for the HTTP library you are using.

Some HTTP client libraries only return the response body after the connection has been closed by the server. These clients will not work for accessing the Streaming API. You must use an HTTP client that will return response data incrementally. Most robust HTTP client libraries will provide this functionality. The Apache HttpClient will handle this use case, for example.

## Disconnections

The Gateway will never close a connection.

## Reconnecting

Once an established connection drops, attempt to reconnect immediately. If the reconnect fails, slow down your reconnect attempts according to the type of error experienced:

Back off linearly for TCP/IP level network errors. These problems are generally temporary and tend to clear quickly. Increase the delay in reconnects by 250ms each attempt, up to 16 seconds.

Back off exponentially for HTTP errors for which reconnecting would be appropriate. Start with a 5 second wait, doubling each attempt, up to 320 seconds.

### 3.2.3.1.2 Streaming API request parameters

## GET /devices/stream

The streaming API has several parameter that influence which information and how the information is transported to the client.

### direction:

- from
- to
- both (default parameter)

### from

This parameter lets the gateway send you all messages over the stream that are coming from the devices. Messages that have been sent to the devices will not show up.

### to

This parameter lets the gateway send you all messages over the stream that are being sent to the devices. Messages that are coming from the devices will not show up. Normally, these messages are the ones that have been issued by yourself if only one client is connected. When more than one client is connected to the gateway, then this stream shows the messages that have been sent to the devices from all controllers - including the ones which are coming from your client.

### both

This parameter lets the gateway send you all messages over the stream that are coming from or going to the devices. These are all events.

### delimited:

- length
- lengthBytes
- lenghtCharacters
- newLine (default parameter)
- emptyLine

#### 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.

#### lenghtCharacters

Every JSON Object is announced by its length in Characters followed by a CR/LF. (The codepage of stream is in UTF-8. Characters and bytes can differ)

#### newLine

Every JSON Object is separated from the next Object by a CR/LF. (Output can not be **formatted**)

#### emptyLine

Every JSON Object is separated from the next Object by a CR/LF/CR/LF.

### output:

- formatted
- singleLine

#### formatted

The JSON Object is formatted nicely, so that it can be read by humans easily. Has no effect if combined with **newLine**

#### singleLine

The JSON Object is given out on a single line.

## Examples

- GET `/devices/stream?direction=both&delimited=emptyLine&output=formatted` (default)
- GET `/devices/stream?direction=both&delimited=emptyLine&output=singleLine`
- GET `/devices/stream?direction=both&delimited=lengthCharacters&output=formatted`
- GET `/devices/stream?direction=to&delimited=lengthBytes&output=singleLine`

### 3.2.3.1.3 Streaming message types

#### Overview

There are for the moment (V 1.0) two types of messages that will be sent over the streaming API

#### States

When connecting to the streaming API, the gateway will deliver at first the states of all the devices that have the flag "TransmitOnConnect" activated. This flag can be found in the administration User Interface when editing the devices.

```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "states",
    "timestamp" : "2015-09-25T17:02:33.967+0200"
  },
  "states" : [ {
    "deviceId" : "01872C13",
    "friendlyId" : "SR04_CO2",
    "physicalDevice" : "SR04_CO2_A5-09-08",
    "functions" : [ {
      "key" : "co2",
      "value" : "643.14",
      "unit" : "ppm",
      "timestamp" : "2015-09-25T17:00:16.036+0200",
      "age" : "137932"
    } ]
  } ]
}, ]
}
```

#### Telegrams

After the states, the connection will be held open and depending on the direction parameter, the subsequent telegrams will be transmitted over the stream.

```
{
  "header" : {
    "content" : "telegram",
    "timestamp" : "2015-10-12T16:25:27.220+0200"
  }
}
```

```
},
"telegram" : {
  "deviceId" : "FEFC8ED5",
  "friendlyId" : "TESTIOFI",
  "timestamp" : "2015-10-12T16:25:27.220+0200",
  "direction" : "from",
  "functions" : [ {
    "key" : "buttonBI",
    "value" : "1",
    "valueKey" : "pressed",
    "meaning" : "Button pressed"
  } ],
  "telegramInfo" : {
    "data" : "50",
    "status" : "30",
    "dbm" : -88,
    "rorg" : "F6"
  }
}
}
```

### 3.2.3.1.4 Processing streaming data

#### Parsing responses

The body of a streaming API response consists of a series of newline-delimited messages, where “newline” is considered to be `\r\n` (in hex: 0x0D 0x0A) and “message” is a JSON encoded data structure or a blank line.

See [streaming message types](#) for information about the message formats you will receive from the streaming API.

#### JSON data

The individual messages streamed by this API are JSON encoded. Keep in mind that the attributes of a JSON-encoded text are unordered - do not rely on fields appearing in any given order. Also keep in mind that your JSON parser should tolerate unexpected or missing fields.

#### Transfer-Encoding: chunked

Streaming connections are encoded using [chunked transfer encoding](#), as indicated by the presence of a `Transfer-Encoding: chunked` HTTP header in the response. Because most HTTP libraries deal with chunked transfer encoding transparently, this document will assume that your code has access to the reassembled HTTP stream and does not have to deal with this encoding.



### 3.2.3.2 From Client to Gateway: Persistent Connection

#### Overview

When issuing a command to the gateway (GET or PUT), the normal behavior would be to establish a second connection (the first is the streaming connection where the events are being pushed from the gateway to the client) send out the request to the gateway and then close the connection again.

The next command (whenever it is) would again open a connection, transmit the command and close the connection again.

This approach leads to some overhead in the discussion between the client and the gateway because the HTTP handshake and the meta data have to be transmitted every time with every command.

Things get worse when the connection is secured (TLS) because the overhead of establishing a secure connection each time is by far bigger than the actual payload (the command) itself.

The approach is technically possible but costs a lot of time and thus latency! That means, there will be a delay between the client issuing a command and the EnOcean device to react, which is not appealing to a user or a system at all.

#### Persistent Connection

The solution is to use a persistent connection between the client and the gateway with the [HTTP Keep-alive function](#).

#### 3.2.3.2.1 Http Keep Alive

#### Overview

Starting with HTTP 1.1 the [Persistent Connection](#) has been introduced.

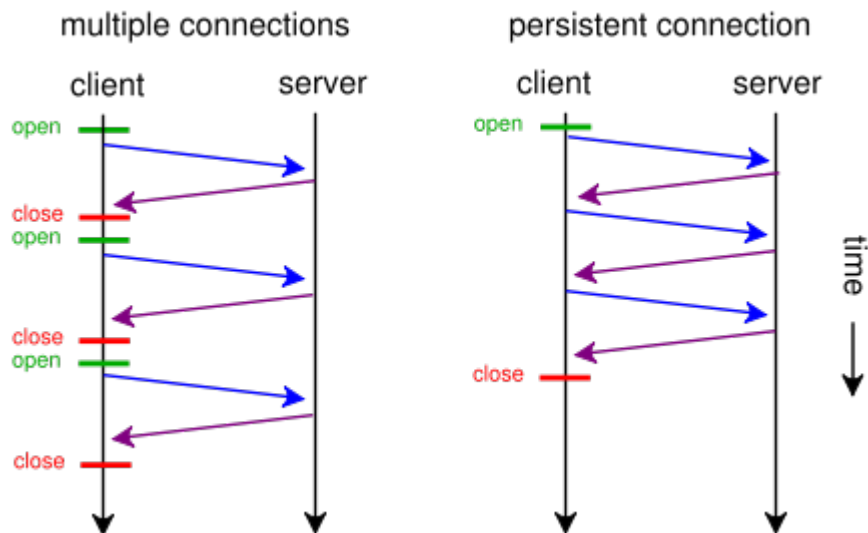
When a request from a client to a server has a ("Connection", "keep-alive") in the HTTP header, the server will not close the underlying TCP Connection for a certain period of time even though the request from the client has been completed. When the client issues another request within that time period, the TCP stack will normally take the already open connection to the server and use it to transmit the request. So there is no waste of time (latency!) for the client when sending commands to the gateway.

This may vary depending on the operating system and the programming language you are using. Please refer to the documentation of your underlying TCP-Stack to ensure you are able to use the Keep-alive function.

#### Gateway Settings

The gateway settings for keep-alive are:

- timeout= <NoLimit>. It means the gateway will keep the connection open.
- max= <NoLimit>. It means how many requests you can do over the same connection before the gateway will force a close of the connection.



### 3.2.3.3 Streaming REST filter

#### Overview

Opening only one Streaming connection from the gateway to the client and parsing the messages according to their device-id and the corresponding modules is best practice approach. You start a stream from the client with a GET <IP-Gateway>:8080/devices/stream that returns the states of the devices and will continue in send continuously events of all devices.

There are also other possibilities to use the REST Implementation but they are more often use when debugging or testing.

#### Stream per device

You can open a connection just to a single device and start a stream which delivers only events of this single device. The REST call would be GET <IP-Gateway>:8080/device/{deviceid}/stream. The gateway will then first deliver the state of the device (if applicable) and then continue with the events of only this device. This can be helpful if you have a big environment and need only a look at specific device and its events.

**From | to | Both**

You can filter the direction of events that the gateway receives or sends. By default, the gateway will transmit events with origin "from" EnOcean sensors/actuators as well as events which destination "to" EnOcean actuators. "To" events are client issued and would be communicated to any connected client.

**3.2.4 REST Resources**

**Overview of used Objects**

To handle the response messages easier, the REST API is split into 5 basic resources.

Each of these objects can be found in the response messages at least once again. It is intended as an aid and not to be confused with the actually available REST resource paths.

|   |  |
|---|--|
| 1 | <b>system</b>  |
|   | version of gateway with EnOcean base information   |
| 2 | <b>profile</b>   |
|   | EEP functionality / functions: Which information will send a specific profile or device and which states can be set? |
| 3 | <b>device</b>  |
|   | Information about known devices of the gateway   |
| 4 | <b>telegram</b>  |
|   | incoming and outgoing telegrams  |
| 5 | <b>state</b>   |
|   | saved states of devices  |

**3.2.4.1 JSON Resource endpoint**

**Overview**

A full listing of all available endpoint URI's with their used HTTP methods.

| Object class | REST path and query | resource |
|--------------|---------------------|----------|
| 1            | systemInfo          |          |

|     |  |            |
|-----|--|------------|
| GET | <b>/system/info</b>                                      | systemInfo |
|     | Get the version of gateway with EnOcean base information |            |

|   |                 |  |
|---|-----------------|--|
| 2 | <b>profiles</b> |  |
|---|-----------------|--|

|     |   |                     |
|-----|---|---------------------|
| GET | <b>/profiles</b>  | profiles (overview) |
|     | Overview of all supported EEP profiles and manufacturer variations  |                     |
| GET | <b>/profiles/{eepld} [?variation={manufacturer}_{product}]</b>  | profiles (detailed) |
|     | Get detailed information of a single EEP functionality. To query a specific variation of an EEP is not mandatory. |                     |
| GET | <b>/devices/{deviceId}/profile</b>  |                     |
|     | Get detailed information of device functionality / functions  |                     |

|   |                |  |
|---|----------------|--|
| 3 | <b>devices</b> |  |
|---|----------------|--|

|     |  |                    |
|-----|--|--------------------|
| GET | <b>/devices</b>                                    | devices (overview) |
|     | List of learned-in devices of the gateway          |                    |
| GET | <b>/devices/{deviceId}</b>                         | devices (detailed) |
|     | Get detailed information of a <b>single</b> device |                    |

|   |                  |  |
|---|------------------|--|
| 4 | <b>telegrams</b> |  |
|---|------------------|--|

|     |  |           |
|-----|--|-----------|
| GET | <b>/devices/telegrams?from to both</b>   | telegrams |
|     | Show last available / historic telegrams of <b>all</b> devices in receiving order, depending on the direction of transport     |           |
| GET | <b>/devices/{deviceId}/telegrams?direction=from to both</b>  |           |
|     | Show last available / historic telegrams of a <b>single</b> device in receiving order, depending on the direction of transport |           |

|   |               |  |
|---|---------------|--|
| 5 | <b>states</b> |  |
|---|---------------|--|

|     |   |        |
|-----|---|--------|
| GET | <b>/devices/states</b>  | states |
|     | Get last states of <b>all</b> learned-in devices.<br><b>notice</b> depending on transmit settings in webinterface |        |
| GET | <b>/devices/{deviceId}/state</b>  |        |
|     | Get last state of a <b>single</b> device.<br><b>notice</b> depending on transmit settings in webinterface         |        |

|       |  |  |
|-------|--|--|
| 4 + 5 | <b>telegram-streams (states + telegrams)</b> |  |
|-------|--|--|

|     |   |                    |
|-----|---|--------------------|
| GET | <b>/devices/stream?direction=from to both</b>   | states + telegrams |
|     | Get the actual state of <b>all</b> devices and continue directly with streaming of device events <b>all</b> devices, depending on the direction of transport              |                    |
| GET | <b>/devices/{deviceId}/stream?direction=from to both</b>  |                    |
|     | Get the actual state of a <b>single</b> device and continue directly with streaming of device events from a <b>single</b> device, depending on the direction of transport |                    |

|     |  |                |
|-----|--|----------------|
| 5   | Setting state                                    |                |
| PUT | <a href="#">/devices/{deviceId}/state</a>        | state (update) |
|     | With this resource, device state can be changed. |                |

### 3.2.4.2 Devices

#### Overview

The /devices resource is helpful, when you want to know the details about the devices, their profiles and also their states. You can also issue commands and get the devices to fulfill actions

#### GET

- [GET /devices](#)

#### PUT

- [PUT /devices](#)

#### 3.2.4.2.1 GET

#### Overview

The /devices resource is helpful, to get to know different details about the devices that are learned-in into the gateway.

#### All devices

- [GET /devices](#)  
Overview of known devices of the gateway
- [GET /devices/states](#)  
Returns last function states of known devices
- [GET /devices/stream](#)  
Streams all transmitted telegrams of all devices including states
- [GET /devices/telegrams](#)  
Show last available / historic telegrams of all devices in receiving order

### Single device

- [GET /devices/{deviceId}](#)  
Detailed information of a device
- [GET /devices/{deviceId}/profile](#)  
Detailed information of a device functionality / functions in JSON
- [GET /devices/{deviceId}/profile.html](#)  
Detailed information of device functionality / functions as HTML
- [GET /devices/{deviceId}/profile.pdf](#)  
Detailed information of device functionality / functions as PDF
- [GET /devices/{deviceId}/state](#)  
Returns last functions state of device
- [GET /devices/{deviceId}/stream](#)  
Stream all transmitted telegrams of device
- [GET /devices/{deviceId}/telegrams](#)  
Show last available / historic telegrams of device in receiving order

#### 3.2.4.2.1.1 GET /devices

### Devices overview

Overview of known devices of the gateway

#### Client to Gateway:

| Resource Path | HTTP method |
|---------------|-------------|
| /devices      | GET         |

no additional call parameters are necessary

#### Response Gateway to Client:

| parameter | datatype         | value / formatting | description |
|-----------|------------------|--------------------|-------------|
| header    | object           | {}                 |             |
| devices   | array of objects | [{}]               |             |
|           | deviceId         | string             | 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. Must be unique.  |
|  | <i>physicalDevice (optional)</i> | <i>string</i> |  | <i>User-assigned name to group actuators and sensors.</i>  |

**example:****Basic structure of gateway response:**

```

http://hostname:api_port/devices

{
  "header" : {
    "httpStatus" : 200,
    "content" : "devices",
    "gateway" : "STC-IoT v0.99.0",
    "timestamp" : "2016-05-03T16:56:43.399+0200"
  },
  "devices" : [ {
    "deviceId" : "0029CFC5",
    "friendlyId" : "RockerSwitchDBO",
    "physicalDevice" : "PTM200"
  }
]

```

```

    }, {
      "deviceId" : "0190A078",
      "friendlyId" : "SR_MDS_Solar"
    } ]
  }
}

```

### 3.2.4.2.1.2 GET /devices?newDevice=true

#### Overview

Returns information of newly learned-in devices, that are not yet in the database. These devices are shown in the WebInterface under "New Device". Each time the client issues the command, it will get all the devices that are listed as New Devices.

#### Client to Gateway:

| Resource Path           | HTTP method |
|-------------------------|-------------|
| /devices?newDevice=true | <b>GET</b>  |

#### Response Gateway to Client:

| parameter | datatype                | value / formatting | description |
|-----------|-------------------------|--------------------|-------------|
| header    | <b>object</b>           | {}                 |             |
| devices   | <b>array of objects</b> | [{}]               |             |



|  |          |        |                  |  |
|--|----------|--------|------------------|--|
|  | 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. |
|--|----------|--------|------------------|--|

**example:**

Basic structure of gateway response:

`http://hostname:api_port/devices?newDevice=true`

```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "devices",
    "gateway" : "STC-IoT v0.99.1",
    "timestamp" : "2016-05-11T16:33:24.997+0200"
  },
  "devices" : [ {
    "deviceId" : "018FBB0B"
  }, {
    "deviceId" : "0191047D"
  } ]
}
```

**3.2.4.2.1.3 GET /devices/states****Overview**

Get last saved states of known devices, depending on transmit settings in webinterface

**Last function states of all known devices**

**Client to Gateway:**

| Resource Path   | HTTP method |
|-----------------|-------------|
| /devices/states | <b>GET</b>  |

**response Gateway to Client:**

| parameter | datatype         | value / formatting | description      |   |
|-----------|------------------|--------------------|------------------|---|
| header    | object           | {}                 |                  |   |
| states    | array of objects | [{}]               |                  |   |
|           | deviceid         | string             | 4 byte hex value | Sender ID 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. Must be unique.     |
|  | <i>physicalDevice (optional)</i> | <i>string</i>    |               | <i>User-assigned name to group actuators and sensors.</i> |
|  | functions                        | array of objects | [{}]          |   |
|  | key                              | string           |               | Technical key of function .                               |
|  | <i>channel (optional)</i>        | <i>integer</i>   |               | <i>Channel if supported by this device.</i>               |
|  | value                            | string / integer |               | Value of the function .                                   |
|  | <i>unit (optional)</i>           | <i>string</i>    | <i>[unit]</i> | <i>Unit ISO.</i>  |
|  | <i>meaning (optional)</i>        | <i>string</i>    |               | <i>Description of value in English.</i>                   |

|  |           |         |                                      |  |
|--|-----------|---------|--------------------------------------|--|
|  | timestamp | string  | YYYY-mm-ddT<br>hh:mm:ss.sss+<br>hhmm | Timestamp function value was received in case of last state functions. |
|  | age       | integer |                                      | Age of function value in case of last state functions in milliseconds. |

### Example:

Here we see the actual state of our example device. As you can see, the gateway stored all function states received in different telegrams. Because of that, the age of the functions are varying.

Basic structure of gateway response:

```
http://hostname:api_port/devices/states

{
  "header" : {
    "httpStatus" : 200,
    "content" : "states",
    "gateway" : "STC-IoT v0.99.0",
    "timestamp" : "2016-05-04T09:54:48.651+0200"
  },
  "states" : [ {
    "deviceId" : "0190A078",
    "friendlyId" : "SR_MDS_Solar",
    "physicalDevice" : "sense-tfh",
    "functions" : [ {
      "key" : "humidity",
      "value" : 29.20,
      "unit" : "%",
      "timestamp" : "2016-05-04T09:53:55.365+0200",
```

```
    "age" : 53286
  }, {
    "key" : "illumination",
    "value" : 1411.76,
    "unit" : "lx",
    "timestamp" : "2016-05-04T09:53:55.365+0200",
    "age" : 53286
  }, {
    "key" : "temperature",
    "value" : 25.28,
    "unit" : "°C",
    "timestamp" : "2016-05-04T09:53:55.365+0200",
    "age" : 53286
  } ]
}, {
  "deviceId" : "01842329",
  "friendlyId" : "WeatherStation",
  "functions" : [ {
    "key" : "dawnSensor",
    "value" : 999.00,
    "unit" : "lx",
    "timestamp" : "2016-05-04T09:46:05.929+0200",
    "age" : 522722
  }, {
    "key" : "dayNight",
    "value" : "day",
    "meaning" : "Day",
    "timestamp" : "2016-05-04T09:46:05.929+0200",
    "age" : 522722
  }, {
    "key" : "hemisphere",
    "value" : "north",
    "meaning" : "North",
    "timestamp" : "2016-05-04T09:46:05.730+0200",
    "age" : 522921
  }, {
    "key" : "rainIndication",
    "value" : "noRain",
    "meaning" : "No rain",
    "timestamp" : "2016-05-04T09:46:05.929+0200",
    "age" : 522722
  }, {
    "key" : "sunEast",
    "value" : 1764.71,
    "unit" : "lx",
    "timestamp" : "2016-05-04T09:46:05.730+0200",
    "age" : 522921
  }, {
    "key" : "sunSouth",
    "value" : 1764.71,
    "unit" : "lx",
    "timestamp" : "2016-05-04T09:46:05.730+0200",
    "age" : 522921
  }, {
    "key" : "sunWest",
    "value" : 588.24,
    "unit" : "lx",
```

```

    "timestamp" : "2016-05-04T09:46:05.730+0200",
    "age" : 522922
  }, {
    "key" : "temperature",
    "value" : 24.94,
    "unit" : "°C",
    "timestamp" : "2016-05-04T09:46:05.929+0200",
    "age" : 522723
  }, {
    "key" : "windSpeed",
    "value" : 0.00,
    "unit" : "m/s",
    "timestamp" : "2016-05-04T09:46:05.929+0200",
    "age" : 522723
  } ]
}, {
  "deviceId" : "FFC5B701",
  "friendlyId" : "2ChannelActuator",
  "functions" : [ {
    "key" : "errorLevel",
    "value" : "notSupported",
    "meaning" : "Error level not supported",
    "timestamp" : "2016-05-04T09:41:35.672+0200",
    "age" : 792980
  }, {
    "key" : "localControl",
    "value" : "on",
    "meaning" : "Local control enabled",
    "timestamp" : "2016-05-04T09:41:35.672+0200",
    "age" : 792980
  }, {
    "key" : "overcurrentSwitchOff",
    "value" : "false",
    "meaning" : "Over current switch off: ready / not support",
    "timestamp" : "2016-05-04T09:41:35.672+0200",
    "age" : 792980
  }, {
    "key" : "switch",
    "value" : "on",
    "meaning" : "Output value ON",
    "timestamp" : "2016-05-04T09:41:35.672+0200",
    "age" : 792980
  } ]
} ]
}

```

### 3.2.4.2.1.4 GET /devices/stream

#### Overview

Get the actual state of all devices and continue directly with updates of the devices via a telegram stream.

Transmitted telegrams of all devices

Client to Gateway:

| Resource Path  | HTTP method |
|--|-------------|
| /devices/stream?direction={ <i>direction</i> }<br>&delimited={ <i>delimited</i> }<br>&output={ <i>output</i> } | <b>GET</b>  |

optional additional call parameters:

| parameter                               | valid values  | call option               | functionality   |
|---|---|---------------------------|---|
| <i>direction</i><br>( <i>optional</i> ) | from   to   both  | <b>default: both</b>      | direction of transport ( <i>from</i> device sent / <i>to</i> device sent / <i>both</i> directions ) |
| <i>delimited</i><br>( <i>optional</i> ) | length   lengthBytes   lengthCharacters   newLine   emptyLine | <b>default: emptyLine</b> | <a href="#">Format and delimiter options</a>  |
| <i>output</i><br>( <i>optional</i> )    | formatted   singleLine  | <b>default: formatted</b> | <a href="#">Format and delimiter options</a>  |

response Gateway to Client:

| parameter | datatype      | value / formatting | description |
|-----------|---------------|--------------------|-------------|
| header    | <b>object</b> | { }                |             |

| states |          | array of objects | {}           |  |
|--------|----------|------------------|--------------|--|
|        | deviceId | string           | xxxx<br>xxxx | S<br>e<br>n<br>d<br>e<br>r<br>I<br>D<br>o<br>f<br>E<br>n<br>o<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>,<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>m<br>o<br>d<br>u<br>l<br>e<br>s<br>u<br>s<br>u<br>a<br>l<br>l<br>y<br>s<br>e<br>n |



|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | d<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>s<br>w<br>i<br>t<br>h<br>t<br>h<br>e<br>i<br>r<br>u<br>n<br>i<br>q<br>u<br>e<br>3<br>2<br>-<br>b<br>i<br>t<br>C<br>h<br>i<br>p<br>I<br>D<br>·<br>I<br>n<br>t<br>h<br>e<br>o<br>t<br>h<br>e<br>r<br>c<br>a |
|--|--|--|--|--|

|  |  |  |  |   |
|--|--|--|--|---|
|  |  |  |  | s<br>e<br>,<br>t<br>h<br>e<br>s<br>e<br>a<br>r<br>e<br>t<br>h<br>e<br>B<br>a<br>s<br>e<br>I<br>D<br>o<br>f<br>t<br>h<br>e<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>. |
|--|--|--|--|---|

|  |            |        |  |   |
|--|------------|--------|--|---|
|  | friendlyId | string |  | User-assigned name of EnOcean device. Must be unique. |
|--|------------|--------|--|---|

|  |                                     |                     |      |  |
|--|-------------------------------------|---------------------|------|--|
|  |                                     |                     |      | U<br>s<br>e<br>r<br>-<br>a<br>s<br>s<br>i<br>g<br>n<br>e<br>d<br>n<br>a<br>m<br>e<br>t<br>o<br>g<br>r<br>o<br>u<br>p<br>a<br>c<br>t<br>u<br>a<br>t<br>o<br>r<br>s<br>a<br>n<br>d<br>s<br>e<br>n<br>s<br>o<br>r<br>s<br>. |
|  | <i>physicalDevice</i><br>(optional) | <i>string</i>       |      |  |
|  | functions                           | array of<br>objects | [{}] |  |

|  |     |        |  |   |
|--|-----|--------|--|---|
|  | key | string |  | T<br>e<br>c<br>h<br>n<br>i<br>c<br>a<br>l<br>k<br>e<br>y<br>o<br>f<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>. |
|--|-----|--------|--|---|

|  |   |                       |  |   |
|--|---|-----------------------|--|---|
|  | <p><i>channel</i><br/>(<i>optional</i>)</p> | <p><i>integer</i></p> |  | <p><i>C</i><br/><i>h</i><br/><i>a</i><br/><i>n</i><br/><i>n</i><br/><i>e</i><br/><i>l</i><br/><i>i</i><br/><i>f</i><br/><i>s</i><br/><i>u</i><br/><i>p</i><br/><i>p</i><br/><i>o</i><br/><i>r</i><br/><i>t</i><br/><i>e</i><br/><i>d</i><br/><i>b</i><br/><i>y</i><br/><i>t</i><br/><i>h</i><br/><i>i</i><br/><i>s</i><br/><i>d</i><br/><i>e</i><br/><i>v</i><br/><i>i</i><br/><i>c</i><br/><i>e</i><br/><i>.</i></p> |
|  | <p>value</p>                                | <p>float</p>          |  | <p><i>V</i><br/><i>a</i><br/><i>l</i><br/><i>u</i><br/><i>e</i><br/><i>o</i><br/><i>f</i><br/><i>t</i><br/><i>h</i><br/><i>e</i><br/><i>f</i><br/><i>u</i><br/><i>n</i><br/><i>c</i><br/><i>t</i><br/><i>i</i><br/><i>o</i><br/><i>n</i><br/><i>.</i></p>   |

|  |                                    |               |                    |  |
|--|------------------------------------|---------------|--------------------|--|
|  | <i>unit<br/>(optional)</i>         | <i>string</i> | <i>[unit<br/>]</i> | <i>U<br/>n<br/>i<br/>t<br/>I<br/>S<br/>O<br/>.</i>   |
|  | <i>meanin<br/>g<br/>(optional)</i> | <i>string</i> |                    | <i>D<br/>e<br/>s<br/>c<br/>r<br/>i<br/>p<br/>t<br/>i<br/>o<br/>n<br/>o<br/>f<br/>v<br/>a<br/>l<br/>u<br/>e<br/>i<br/>n<br/>E<br/>n<br/>g<br/>l<br/>i<br/>s<br/>h<br/>.</i> |

|  |                       |               |   |  |
|--|-----------------------|---------------|---|--|
|  | <p>timesta<br/>mp</p> | <p>string</p> | <p>yyyy<br/>-<br/>mm-<br/>ddT<br/>hh:m<br/>m:ss.<br/>sss+<br/>hhm<br/>m</p> | <p>T<br/>i<br/>m<br/>e<br/>s<br/>t<br/>a<br/>m<br/>p<br/>f<br/>u<br/>n<br/>c<br/>t<br/>i<br/>o<br/>n<br/>v<br/>a<br/>l<br/>u<br/>e<br/>w<br/>a<br/>s<br/>r<br/>e<br/>c<br/>e<br/>i<br/>v<br/>e<br/>d<br/>i<br/>n<br/>c<br/>a<br/>s<br/>e<br/>o<br/>f<br/>l<br/>a<br/>s<br/>t<br/>s<br/>t<br/>a<br/>t<br/>e</p> |
|--|-----------------------|---------------|---|--|



|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>s<br>. |
|--|--|--|--|--|

|  |     |         |  |   |
|--|-----|---------|--|---|
|  | age | integer |  | A<br>g<br>e<br>o<br>f<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>v<br>a<br>l<br>u<br>e<br>i<br>n<br>c<br>a<br>s<br>e<br>o<br>f<br>l<br>a<br>s<br>t<br>s<br>t<br>a<br>t<br>e<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>s<br>i<br>m<br>i<br>l<br>l |
|--|-----|---------|--|---|

|   |  |               |           |   |
|---|--|---------------|-----------|---|
|   |  |               |           | i<br>s<br>e<br>c<br>o<br>n<br>d<br>s<br>. |
| <i>followed by live telegram stream</i>   |  |               |           |   |
| <i>header</i>                             |  | <b>object</b> | <b>{}</b> |   |
| <i>telegram</i>                           |  | <b>object</b> | <b>{}</b> |   |
| <i>Case 1/2: telegram of known device</i> |  |               |           |   |

|  |          |        |              |   |
|--|----------|--------|--------------|---|
|  | deviceId | string | xxxx<br>xxxx | S<br>e<br>n<br>d<br>e<br>r<br>I<br>D<br>o<br>f<br>E<br>n<br>o<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>,<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>m<br>o<br>d<br>u<br>l<br>e<br>s<br>s<br>u<br>a<br>l<br>l<br>y<br>s<br>e<br>n<br>d<br>t |
|--|----------|--------|--------------|---|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | e<br>l<br>e<br>g<br>r<br>a<br>n<br>s<br>w<br>i<br>t<br>h<br>t<br>h<br>e<br>i<br>r<br>u<br>n<br>i<br>q<br>u<br>e<br>3<br>2<br>-<br>b<br>i<br>t<br>C<br>h<br>i<br>p<br>I<br>D<br>·<br>I<br>n<br>t<br>h<br>e<br>o<br>t<br>h<br>e<br>r<br>c<br>a<br>s<br>e |
|--|--|--|--|--|

|  |  |  |  |   |
|--|--|--|--|---|
|  |  |  |  | ,<br>t<br>h<br>e<br>s<br>e<br>a<br>r<br>e<br>t<br>h<br>e<br>B<br>a<br>s<br>e<br>I<br>D<br>o<br>f<br>t<br>h<br>e<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>. |
|--|--|--|--|---|

|  |            |        |  |   |
|--|------------|--------|--|---|
|  | friendlyId | string |  | User-assigned name of ENOcean device. Must be unique. |
|--|------------|--------|--|---|

|  |                                      |               |  |  |
|--|--------------------------------------|---------------|--|--|
|  | <i>physicalDevice<br/>(optional)</i> | <i>string</i> |  | <i>U<br/>s<br/>e<br/>r<br/>-<br/>a<br/>s<br/>s<br/>i<br/>g<br/>n<br/>e<br/>d<br/>n<br/>a<br/>m<br/>e<br/>t<br/>o<br/>g<br/>r<br/>o<br/>u<br/>p<br/>a<br/>c<br/>t<br/>u<br/>a<br/>t<br/>o<br/>r<br/>s<br/>a<br/>n<br/>d<br/>s<br/>e<br/>n<br/>s<br/>o<br/>r<br/>s<br/>.</i> |
|--|--------------------------------------|---------------|--|--|



|  |           |        |  |   |
|--|-----------|--------|--|---|
|  | timestamp | string | yyyy<br>-<br>mm-<br>ddT<br>hh:m<br>m:ss.<br>sss+<br>hhm<br>m | t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>s<br>s<br>i<br>o<br>n<br>t<br>i<br>m<br>e<br>i<br>n<br>U<br>T<br>C<br>f<br>o<br>r<br>m<br>a<br>t |
|  | direction | string | from<br> <br>to  | d<br>i<br>r<br>e<br>c<br>t<br>i<br>o<br>n<br>o<br>f<br>t<br>r<br>a<br>n   |

|  |           |                  |      |   |
|--|-----------|------------------|------|---|
|  |           |                  |      | support (from device sent / to device sent) |
|  | functions | array of objects | [{}] |   |

|  |     |        |  |   |
|--|-----|--------|--|---|
|  | key | string |  | T<br>e<br>c<br>h<br>n<br>i<br>c<br>a<br>l<br>k<br>e<br>y<br>o<br>f<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>. |
|--|-----|--------|--|---|

|  |   |                       |  |   |
|--|---|-----------------------|--|---|
|  | <p><i>channel</i><br/>(<i>optional</i>)</p> | <p><i>integer</i></p> |  | <p><i>C</i><br/><i>h</i><br/><i>a</i><br/><i>n</i><br/><i>n</i><br/><i>e</i><br/><i>l</i><br/><i>i</i><br/><i>f</i><br/><i>s</i><br/><i>u</i><br/><i>p</i><br/><i>p</i><br/><i>o</i><br/><i>r</i><br/><i>t</i><br/><i>e</i><br/><i>d</i><br/><i>b</i><br/><i>y</i><br/><i>t</i><br/><i>h</i><br/><i>i</i><br/><i>s</i><br/><i>d</i><br/><i>e</i><br/><i>v</i><br/><i>i</i><br/><i>c</i><br/><i>e</i><br/><i>.</i></p> |
|  | <p>value</p>                                | <p>float</p>          |  | <p><i>V</i><br/><i>a</i><br/><i>l</i><br/><i>u</i><br/><i>e</i><br/><i>o</i><br/><i>f</i><br/><i>t</i><br/><i>h</i><br/><i>e</i><br/><i>f</i><br/><i>u</i><br/><i>n</i><br/><i>c</i><br/><i>t</i><br/><i>i</i><br/><i>o</i><br/><i>n</i><br/><i>.</i></p>   |

|  |                                    |               |                    |  |
|--|------------------------------------|---------------|--------------------|--|
|  | <i>unit<br/>(optional)</i>         | <i>string</i> | <i>[unit<br/>]</i> | <i>U<br/>n<br/>i<br/>t<br/>I<br/>S<br/>O<br/>.</i>   |
|  | <i>meanin<br/>g<br/>(optional)</i> | <i>string</i> |                    | <i>D<br/>e<br/>s<br/>c<br/>r<br/>i<br/>p<br/>t<br/>i<br/>o<br/>n<br/>o<br/>f<br/>v<br/>a<br/>l<br/>u<br/>e<br/>i<br/>n<br/>E<br/>n<br/>g<br/>l<br/>i<br/>s<br/>h<br/>.</i> |
|  | telegramInfo                       | object        | {}                 |  |

|  |      |        |                               |   |
|--|------|--------|-------------------------------|---|
|  | data | string | x<br>byte<br>hex<br>valu<br>e | P<br>a<br>y<br>l<br>o<br>a<br>d<br>o<br>f<br>E<br>R<br>P<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>s<br>o<br>r<br>E<br>S<br>P<br>p<br>a<br>c<br>k<br>e<br>t<br>s |
|--|------|--------|-------------------------------|---|

|  |        |         |  |  |
|--|--------|---------|--|--|
|  | status | integer |  | i<br>d<br>e<br>n<br>t<br>i<br>f<br>i<br>e<br>s<br>i<br>f<br>t<br>h<br>e<br>s<br>u<br>b<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>i<br>s<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>t<br>t<br>e<br>d<br>f<br>r<br>o<br>m<br>a<br>r<br>e<br>p<br>e<br>a<br>t |
|--|--------|---------|--|--|

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

e  
r  
a  
n  
d  
t  
h  
e  
t  
y  
p  
e  
o  
f  
i  
n  
t  
e  
g  
r  
i  
t  
y  
c  
o  
n  
t  
r  
o  
l  
m  
e  
c  
h  
a  
n  
i  
s  
m  
u  
s  
e  
d  
(  
n  
o  
t  
e  
:  
n



|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

o  
t  
p  
r  
e  
s  
e  
n  
t  
i  
n  
a  
R  
P  
S  
t  
e  
l  
e  
g  
r  
a  
m  
)

|  |     |         |  |  |
|--|-----|---------|--|--|
|  | dbm | integer |  | s<br>i<br>g<br>n<br>a<br>l<br>s<br>t<br>r<br>e<br>n<br>g<br>t<br>h<br>o<br>f<br>r<br>e<br>c<br>e<br>i<br>v<br>e<br>d<br>/<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>t<br>t<br>e<br>d<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m |
|--|-----|---------|--|--|

|   |          |        |                               |  |
|---|----------|--------|-------------------------------|--|
|   | rorg     | string | 1<br>byte<br>hex<br>valu<br>e | i<br>d<br>e<br>n<br>t<br>i<br>f<br>i<br>e<br>r<br>f<br>o<br>r<br>s<br>u<br>b<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>t<br>y<br>p<br>e |
| <i>Case 2/2: telegram of unknown device (only if Filtermode is Off)</i> |          |        |                               |  |
|   | deviceId | string | 4<br>byte<br>hex<br>valu<br>e | S<br>e<br>n<br>d<br>e<br>r<br>I<br>D<br>o<br>f<br>E<br>n<br>o<br>c<br>e<br>a<br>n<br>d<br>e<br>v   |

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | i<br>c<br>e<br>,<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>m<br>o<br>d<br>u<br>l<br>e<br>s<br>u<br>s<br>u<br>a<br>l<br>l<br>y<br>s<br>e<br>n<br>d<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>s<br>w<br>i<br>t<br>h<br>t<br>h<br>e<br>i<br>r<br>u<br>n<br>i |
|--|--|--|--|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | q<br>u<br>e<br>3<br>2<br>-<br>b<br>i<br>t<br>C<br>h<br>i<br>p<br>I<br>D<br>.<br>I<br>n<br>t<br>h<br>e<br>o<br>t<br>h<br>e<br>r<br>c<br>a<br>s<br>e<br>,<br>t<br>h<br>e<br>s<br>e<br>a<br>r<br>e<br>t<br>h<br>e<br>B<br>a<br>s<br>e<br>I<br>D<br>o<br>f |
|--|--|--|--|--|

|  |  |  |  |   |
|--|--|--|--|---|
|  |  |  |  | t<br>h<br>e<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>. |
|--|--|--|--|---|

|  |           |        |  |   |
|--|-----------|--------|--|---|
|  | timestamp | string | yyyy-<br>mm-<br>ddT<br>hh:m<br>m:ss.<br>sss+<br>hhm<br>m | t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>s<br>s<br>i<br>o<br>n<br>t<br>i<br>m<br>e<br>i<br>n<br>U<br>T<br>C<br>f<br>o<br>r<br>m<br>a<br>t |
|--|-----------|--------|--|---|

|  |              |        |              |  |
|--|--------------|--------|--------------|--|
|  | direction    | string | from<br>  to | d<br>i<br>r<br>e<br>c<br>t<br>i<br>o<br>n<br>o<br>f<br>t<br>r<br>a<br>n<br>s<br>p<br>o<br>r<br>t<br>(<br>f<br>r<br>o<br>m<br>G<br>W<br>s<br>e<br>n<br>t<br>o<br>r<br>r<br>e<br>c<br>e<br>i<br>v<br>e<br>d<br>) |
|  | telegramInfo | object | {}           |  |



|  |      |        |                               |  |
|--|------|--------|-------------------------------|--|
|  | data | string | x<br>byte<br>hex<br>valu<br>e | P<br>a<br>y<br>l<br>o<br>a<br>d<br>o<br>f<br>E<br>R<br>P<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>s<br>o<br>r<br>E<br>S<br>P<br>p<br>a<br>c<br>k<br>e<br>t<br>s<br>. |
|--|------|--------|-------------------------------|--|

|  |        |         |  |  |
|--|--------|---------|--|--|
|  | status | integer |  | i<br>d<br>e<br>n<br>t<br>i<br>f<br>i<br>e<br>s<br>i<br>f<br>t<br>h<br>e<br>s<br>u<br>b<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>i<br>s<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>t<br>t<br>e<br>d<br>f<br>r<br>o<br>m<br>a<br>r<br>e<br>p<br>e<br>a<br>t |
|--|--------|---------|--|--|

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

e  
r  
a  
n  
d  
t  
h  
e  
t  
y  
p  
e  
o  
f  
i  
n  
t  
e  
g  
r  
i  
t  
y  
c  
o  
n  
t  
r  
o  
l  
m  
e  
c  
h  
a  
n  
i  
s  
m  
u  
s  
e  
d  
.  
(  
n  
o  
t  
e  
:

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

n  
o  
t  
p  
r  
e  
s  
e  
n  
t  
i  
n  
a  
R  
P  
S  
t  
e  
l  
e  
g  
r  
a  
n  
)

|  |     |         |  |  |
|--|-----|---------|--|--|
|  | dbm | integer |  | s<br>i<br>g<br>n<br>a<br>l<br>s<br>t<br>r<br>e<br>n<br>g<br>t<br>h<br>o<br>f<br>r<br>e<br>c<br>e<br>i<br>v<br>e<br>d<br>/<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>t<br>t<br>e<br>d<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m |
|--|-----|---------|--|--|

|  |      |        |                               |  |
|--|------|--------|-------------------------------|--|
|  | rorg | string | 1<br>byte<br>hex<br>valu<br>e | i<br>d<br>e<br>n<br>t<br>i<br>f<br>i<br>e<br>r<br>f<br>o<br>r<br>s<br>u<br>b<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>t<br>y<br>p<br>e |
|--|------|--------|-------------------------------|--|

### Example:

In that example we received the state of `multiFuncAlphaEos` and telegrams from an unknown device and a `myRocker` switch.

Basic structure of gateway response:

```
http://hostname:api_port/devices/stream

{
  "header" : {
    "httpStatus" : 200,
    "content" : "states",
    "timestamp" : "2016-05-04T10:00:02.570+0200"
  },
  "states" : [ {
    "deviceId" : "0190A078",
    "friendlyId" : "SR_MDS_Solar",
    "physicalDevice" : "sense-tf-h",
    "functions" : [ {
      "key" : "humidity",
      "value" : 28.80,
```

```

        "unit" : "%",
        "timestamp" : "2016-05-04T09:59:35.758+0200",
        "age" : 26866
    }, {
        "key" : "illumination",
        "value" : 1411.76,
        "unit" : "lx",
        "timestamp" : "2016-05-04T09:59:35.758+0200",
        "age" : 26867
    }, {
        "key" : "temperature",
        "value" : 25.12,
        "unit" : "°C",
        "timestamp" : "2016-05-04T09:59:35.758+0200",
        "age" : 26867
    } ]
} ]
}

{
  "header" : {
    "content" : "telegram",
    "timestamp" : "2016-05-04T10:00:04.758+0200"
  },
  "telegram" : {
    "deviceId" : "0187D64B",
    "timestamp" : "2016-05-04T10:00:04.758+0200",
    "direction" : "from",
    "telegramInfo" : {
      "data" : "00000E5D",
      "status" : "0",
      "dbm" : -77,
      "rorg" : "A5"
    }
  }
}

{
  "header" : {
    "content" : "telegram",
    "timestamp" : "2016-05-04T10:00:58.859+0200"
  },
  "telegram" : {
    "deviceId" : "0029CFC5",
    "friendlyId" : "myRocker",
    "timestamp" : "2016-05-04T10:00:58.859+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "buttonAI",
      "value" : "pressed",
      "meaning" : "Button pressed"
    } ],
    "telegramInfo" : {
      "data" : "10",
      "status" : "30",
      "dbm" : -71,
      "rorg" : "F6"
    }
  }
}

```

```

    }
  }
}
{
  "header" : {
    "content" : "telegram",
    "timestamp" : "2016-05-04T10:00:59.027+0200"
  },
  "telegram" : {
    "deviceId" : "0029CFC5",
    "friendlyId" : "myRocker",
    "timestamp" : "2016-05-04T10:00:59.027+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "buttonAI",
      "value" : "released",
      "meaning" : "Button released"
    } ],
    "telegramInfo" : {
      "data" : "00",
      "status" : "20",
      "dbm" : -71,
      "rorg" : "F6"
    }
  }
}
}
}

```

### 3.2.4.2.1.5 GET /devices/telegrams

#### Overview

Show the last 1000 available / historic telegrams of all devices in receiving order depending on the direction of transport (from device sent / to device sent / both directions).

#### Last transmitted telegrams of all known devices

##### Client to Gateway:

| Resource Path   | HTTP method |
|---|-------------|
| /devices/telegrams?<br>direction={ <i>direction</i> } | <b>GET</b>  |

##### optional additional call parameters:

| parameter | valid values | call option | functionality |
|-----------|--------------|-------------|---------------|
|-----------|--------------|-------------|---------------|



|                             |                  |                      |   |
|-----------------------------|------------------|----------------------|---|
| <i>direction (optional)</i> | from   to   both | <b>default: both</b> | direction of transport ( <i>from</i> device sent / <i>to</i> device sent / <i>both</i> directions ) |
|-----------------------------|------------------|----------------------|---|

**response Gateway to Client:**

| parameter                                 | datatype               | value / formatting | description |
|---|------------------------|--------------------|-------------|
| header                                    | <b>object</b>          | { }                |             |
| telegrams                                 | <b>array of object</b> | [{ }]              |             |
| <i>Case 1/2: telegram of known device</i> |                        |                    |             |

|  |            |        |                  |   |
|--|------------|--------|------------------|---|
|  | deviceId   | string | 4 byte hex value | Sender ID 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. Must be unique.   |

|  |                                     |                  |                                 |   |
|--|-------------------------------------|------------------|---------------------------------|---|
|  | <i>physicalDevice</i><br>(optional) | string           |                                 | User-assign ed name to group actuators and sensors.                       |
|  | timestamp                           | string           | yyyy-mm-ddT hh:mm:ss.sss + hhmm | telegram transmission time in UTC format                                  |
|  | direction                           | string           | from   to                       | direction of transport ( <i>from</i> device sent / <i>to</i> device sent) |
|  | functions                           | array of objects | [{}]                            |   |
|  | key                                 | string           |                                 | Technical key of function.  |
|  | <i>channel</i><br>(optional)        | <i>integer</i>   |                                 | <i>Channel if supported by this device.</i>                               |
|  | value                               | float            |                                 | Value of the function.  |

|  | <i>unit (optional)</i>    | <i>string</i> | <i>[unit]</i>    | <i>Unit ISO.</i>   |
|--|---------------------------|---------------|------------------|--|
|  | <i>meaning (optional)</i> | <i>string</i> |                  | <i>Description of value in English .</i>   |
|  | telegramInfo              | object        | {}               |  |
|  | data                      | string        | x byte hex value | Payloa<br>d of<br>ERP<br>telegra<br>ms or<br>ESP<br>packet<br>s  |
|  | status                    | intege<br>r   |                  | identif<br>ies if<br>the<br>subtel<br>egram<br>is<br>transm<br>itted<br>from a<br>repeat<br>er and<br>the<br>type of<br>integri<br>ty<br>control<br>mecha<br>nism<br>used<br>(note:<br>not<br>presen<br>t in a<br>RPS<br>telegra<br>m) |

|   |  |          |         |                  |   |
|---|--|----------|---------|------------------|---|
|   |  | dbm      | integer |                  | signal strength of received / transmitted telegram  |
|   |  | rorg     | string  | 1 byte hex value | identifier for subtelegram type   |
| <i>Case 2/2: telegram of unknown device (only if Filtermode is Off)</i> |  |          |         |                  |   |
|   |  | deviceid | string  | 4 byte hex value | Sender ID 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. |

|  |              |        |                                       |   |
|--|--------------|--------|---------------------------------------|---|
|  | timestamp    | string | yyyy-mm-ddT<br>hh:mm:ss.sss<br>+ hhmm | telegram transmission time in UTC format                                  |
|  | direction    | string | from                                  | direction of transport ( <i>from</i> device sent / <i>to</i> device sent) |
|  | telegramInfo | object | {}                                    |   |
|  | 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 RPS telegram) |
|  | dbm    | integer |                  | signal strength of received / transmitted telegram  |
|  | rorg   | string  | 1 byte hex value | identifier for subtelegram type   |

**Example:**

Three different telegrams of two devices has been received in history. One of them is not learned-in yet (deviceId = 0187D64B).

Please note that (depending on the settings in the web interface) you will see only functions, where the value has been changed.

#### Basic structure of gateway response:

```

http://hostname:api_port/devices/telegrams

{
  "header" : {
    "httpStatus" : 200,
    "content" : "telegrams",
    "gateway" : "STC-IoT v0.99.0",
    "timestamp" : "2016-05-04T10:22:05.162+0200"
  },
  "telegrams" : [ {
    "deviceId" : "0187D64B",
    "timestamp" : "2016-05-04T09:59:28.668+0200",
    "direction" : "from",
    "telegramInfo" : {
      "data" : "00000F5D",
      "status" : "0",
      "dbm" : -77,
      "rorg" : "A5"
    }
  }, {
    "deviceId" : "0187D64B",
    "timestamp" : "2016-05-04T09:59:33.819+0200",
    "direction" : "from",
    "telegramInfo" : {
      "data" : "0000105D",
      "status" : "0",
      "dbm" : -77,
      "rorg" : "A5"
    }
  }, {
    "deviceId" : "0190A078",
    "friendlyId" : "SR_MDS_Solar",
    "physicalDevice" : "Sense-tf-h",
    "timestamp" : "2016-05-04T09:59:35.758+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "illumination",
      "value" : 1411.76,
      "unit" : "lx"
    }, {
      "key" : "humidity",
      "value" : 28.80,
      "unit" : "%"
    }, {
      "key" : "temperature",
      "value" : 25.12,
      "unit" : "°C"
    } ],
    "telegramInfo" : {
      "data" : "0C489D0A",

```



```

    "status" : "0",
    "dbm" : -76,
    "rorg" : "A5"
  }
}
}

```

### 3.2.4.2.1.6 GET /devices/{deviceId}

#### Overview

Get detailed informations of a single device

#### Client to Gateway:

| Resource Path                | HTTP method |
|------------------------------|-------------|
| /devices/{ <i>deviceId</i> } | <b>GET</b>  |

#### necessary additional call parameters:

| parameter       | valid values          | call option | description  |
|-----------------|-----------------------|-------------|--|
| <i>deviceId</i> | deviceId   friendlyId | required    | 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. |

**Response Gateway to Client:**

| parameter |  | data type | value / formatting | description |
|-----------|--|-----------|--------------------|-------------|
| header    |  | object    | {}                 |             |
| device    |  | object    | {}                 |             |

|  |           |  |        |                  |   |
|--|-----------|--|--------|------------------|---|
|  | device id |  | string | 4 byte hex value | Sender ID of Ocean ean device, Ocean module usually send telegram with their unique 32-bit Chip ID. In the other case, th |
|--|-----------|--|--------|------------------|---|

|  |                            |  |            |  |  |
|--|----------------------------|--|------------|--|--|
|  |                            |  |            |  | es<br>e<br>ar<br>e<br>th<br>e<br>Ba<br>se<br>ID<br>of<br>th<br>e<br>En<br>Oc<br>ea<br>n<br>de<br>vic<br>e.                             |
|  | fri<br>en<br>dl<br>yl<br>d |  | str<br>ing |  | Us<br>er-<br>ass<br>ign<br>ed<br>na<br>m<br>e<br>of<br>En<br>Oc<br>ea<br>n<br>de<br>vic<br>e.<br>M<br>ust<br>be<br>un<br>iq<br>ue<br>. |

|  |                        |                                   |                     |                                   |  |
|--|------------------------|-----------------------------------|---------------------|-----------------------------------|--|
|  | <i>physical Device</i> |                                   | <i>string</i>       |                                   | <i>Us er- as sig ne d na me to gr ou p act ua tor s an d se ns ors .</i> |
|  | ee ps                  |                                   | arr ay of ob jec ts | [{}]                              |  |
|  |                        | ee p                              | str ing             |                                   | ee pld   |
|  |                        | <i>va ria tio n (o pti on al)</i> | <i>str ing</i>      | <i>manufacturer _ productname</i> | <i>M an uf act ur er sp eci fic va ria tio n</i>                         |
|  |                        | ve rsi on                         | str ing             |                                   | AP I Ve rsi on   |

|  |                     |           |        |                               |  |
|--|---------------------|-----------|--------|-------------------------------|--|
|  |                     | direction | string | from   to   both              | "from", "to" or "both"   |
|  | firstSeen           |           | string | yyyy-mm-ddT hh:mm:ss.sss+hhmm | timestamp on which the device has been detected for the first time |
|  | lastSeen (optional) |           | string | yyyy-mm-ddT hh:mm:ss.sss+hhmm | timestamp on which the   |



Get detailed informations of a device that uses one or more EEPs including vendor specific EEP variations.

**Client to Gateway:**

| Resource Path                        | HTTP method |
|--------------------------------------|-------------|
| /devices/{ <i>deviceId</i> }/profile | <b>GET</b>  |

**necessary additional call parameters:**

| parameter       | values                | call option | description  |
|-----------------|-----------------------|-------------|--|
| <i>deviceId</i> | deviceId   friendlyId | required    | 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. |

**response Gateway to Client:**



| parameter                 |                                | datatype                | value / formatting | description  |
|---------------------------|--------------------------------|-------------------------|--------------------|--|
| header                    |                                | <b>object</b>           | {}                 |  |
| profile                   |                                | <b>object</b>           | {}                 |  |
|                           | eep                            | string                  | <b>xx-xx-xx</b>    | EnOcean Equipment Profiles, definition of EnOcean radio telegram structure           |
|                           | title                          | string                  |                    | description of eep profile   |
|                           | functionGroups                 | array of objects        | [{}]               |  |
|                           | direction                      | string                  | from   to          | direction of transport ( <i>from</i> or <i>to</i> device)                            |
|                           | functions                      | array of objects        | [{}]               |  |
|                           | key                            | string                  |                    | Technical key of function.   |
|                           | <i>description (optional)</i>  | <i>string</i>           |                    | <i>description of function</i>   |
|                           | <i>defaultValue (optional)</i> | <i>string / integer</i> |                    | <i>Default value will be taken if no value has been set. Not present on any key.</i> |
|                           | values                         | array of objects        | [{}]               |  |
| <b>Case: value ranges</b> |                                |                         |                    |  |
|                           | <i>meaning (optional)</i>      | <i>string</i>           |                    | <i>meaning of range</i>  |
|                           | range                          | object                  | {}                 |  |
|                           | min                            | float                   |                    | Logical minimal value.   |

|                                 |                           |               |               |                               |
|---------------------------------|---------------------------|---------------|---------------|-------------------------------|
|                                 | max                       | float         |               | Logical maximal value.        |
|                                 | step                      | float         |               | Value stepping.               |
|                                 | <i>unit (optional)</i>    | <i>string</i> | <b>[unit]</b> | <i>Value unit in English.</i> |
| <b>Case: value enumerations</b> |                           |               |               |                               |
|                                 | value                     | float         |               | Value Constant                |
|                                 | <i>meaning (optional)</i> | <i>string</i> |               | <i>meaning of value</i>       |

**Example:**

The example shows device functionality of a device named Radiator. The result equals to one profile A5-20-01 in this case. If a device uses more than one EEP the functionGroups array would get longer.

## Basic structure of gateway response:

```

http://hostname:api_port/devices/Radiator/profile

{
  "header" : {
    "httpStatus" : 200,
    "content" : "profile",
    "gateway" : "STC-IoT v0.99.0",
    "timestamp" : "2016-05-04T14:27:16.280+0200"
  },
  "profile" : {
    "functionGroups" : [ {
      "direction" : "from",
      "functions" : [ {
        "key" : "actuatorObstructed",
        "description" : "Actuator obstructed",
        "values" : [ {
          "value" : "false",
          "meaning" : "Actuator is free"
        }, {
          "value" : "true",
          "meaning" : "Actuator is obstructed"
        } ]
      } ]
    }, {
      "key" : "batteryLow",

```

```
"description" : "Change battery next days",
"values" : [ {
  "value" : "false",
  "meaning" : "Battery does not need to be changed"
}, {
  "value" : "true",
  "meaning" : "Change battery in the next days"
} ]
}, {
  "key" : "contact",
  "description" : "Contact or cover",
  "values" : [ {
    "value" : "closed",
    "meaning" : "Contact or cover is closed"
  }, {
    "value" : "open",
    "meaning" : "Contact or cover is open"
  } ]
}, {
  "key" : "energyInput",
  "description" : "Energy input enabled",
  "values" : [ {
    "value" : "false",
    "meaning" : "Energy input is disabled"
  }, {
    "value" : "true",
    "meaning" : "Energy input is enabled"
  } ]
}, {
  "key" : "energyStorageCharged",
  "description" : "Energy storage sufficiently charged",
  "values" : [ {
    "value" : "false",
    "meaning" : "Energy storage needs to be charged"
  }, {
    "value" : "true",
    "meaning" : "Energy storage is sufficiently charge
  } ]
}, {
  "key" : "serviceMode",
  "description" : "Service On",
  "values" : [ {
    "value" : "off",
    "meaning" : "Service is off"
  }, {
    "value" : "on",
    "meaning" : "Service is on"
  } ]
}, {
  "key" : "temperature",
  "description" : "Temperature (linear)",
  "values" : [ {
    "range" : {
      "min" : 0,
      "max" : 40,
      "step" : 0.157,
      "unit" : "°C"
```

```

    }
  }, {
    "value" : "overRange",
    "meaning" : "Temperature sensor failure or out of
  } ]
}, {
  "key" : "valve",
  "description" : "Current valve position",
  "values" : [ {
    "range" : {
      "min" : 0,
      "max" : 100,
      "step" : 1,
      "unit" : "%"
    }
  } ]
}, {
  "key" : "window",
  "description" : "Detection, window open",
  "values" : [ {
    "value" : "closed",
    "meaning" : "Normal operation"
  }, {
    "value" : "open",
    "meaning" : "Open window detected"
  } ]
} ]
}, {
  "title" : "Control actuator via valve position",
  "direction" : "to",
  "functions" : [ {
    "key" : "setPointInverse",
    "description" : "Valve set point can be sent to the
  "values" : [ {
    "value" : "false",
    "meaning" : "Set point is normal"
  }, {
    "value" : "true",
    "meaning" : "Set point is inversed"
  } ],
  "defaultValue" : "false"
}, {
  "key" : "summerMode",
  "description" : "The radio communication between the
  "values" : [ {
    "value" : "false",
    "meaning" : "Summer mode is deactivated"
  }, {
    "value" : "true",
    "meaning" : "Summer mode is activated"
  } ],
  "defaultValue" : "false"
}, {
  "key" : "valve",
  "description" : "Valve position",
  "values" : [ {
    "range" : {

```

```

        "min" : 0,
        "max" : 100,
        "step" : 1,
        "unit" : "%"
    }
} ]
} ]
}, {
    "title" : "Control actuator via temperature",
    "direction" : "to",
    "functions" : [ {
        "key" : "summerMode",
        "description" : "The radio communication between the
        "values" : [ {
            "value" : "false",
            "meaning" : "Summer mode is deactivated"
        }, {
            "value" : "true",
            "meaning" : "Summer mode is activated"
        } ],
        "defaultValue" : "false"
    }, {
        "key" : "temperature",
        "description" : "Temperature from room control unit
        "values" : [ {
            "range" : {
                "min" : 0,
                "max" : 40,
                "step" : 0.157,
                "unit" : "°C"
            }
        } ]
    }, {
        "key" : "temperatureSetPoint",
        "description" : "Temperature set point",
        "values" : [ {
            "range" : {
                "min" : 0,
                "max" : 40,
                "step" : 0.157,
                "unit" : "°C"
            }
        } ]
    } ]
}, {
    "title" : "Service mode",
    "direction" : "to",
    "functions" : [ {
        "key" : "runInitSequence",
        "description" : "The limit switching measures the tr
        "values" : [ {
            "value" : "false",
            "meaning" : "Do not run init sequence"
        }, {
            "value" : "true",
            "meaning" : "Run init sequence"
        } ],
    } ],

```

```

    "defaultValue" : "false"
  }, {
    "key" : "runLiftSet",
    "description" : "Initialization, adjustment to the v
    "values" : [ {
      "value" : "false",
      "meaning" : "Done with initialization"
    }, {
      "value" : "true",
      "meaning" : "Initialization is running"
    } ],
    "defaultValue" : "false"
  }, {
    "key" : "valveMaintenance",
    "description" : "Operate valve in maintenance mode",
    "values" : [ {
      "value" : "close",
      "meaning" : "Close valve"
    }, {
      "value" : "noChange",
      "meaning" : "Do not move valve after init sequence
    }, {
      "value" : "open",
      "meaning" : "Open valve"
    } ],
    "defaultValue" : "noChange"
  } ]
} ]
}
}

```

#### 3.2.4.2.1.8 GET /devices/{deviceId}/profile.html

##### Overview

See [HTML description](#)

#### 3.2.4.2.1.9 GET /devices/{deviceId}/profile.pdf

##### Overview

See [PDF description](#)

#### 3.2.4.2.1.10 GET /devices/{deviceId}/state

##### Overview

Get last saved states of a device, depending on transmit settings in webinterface

**Last function states of a single device**

**Client to Gateway:**

| Resource Path                      | HTTP method |
|------------------------------------|-------------|
| /devices/{ <i>deviceId</i> }/state | <b>GET</b>  |

**necessary additional call parameters:**

| parameter       | valid values   | call option | functionality   |
|-----------------|--|-------------|---|
| <i>deviceId</i> | valid<br>deviceId or<br>friendlyId<br><br>4 byte hex<br>value or<br>friendlyId | required    | SenderId of<br>EnOcean<br>device,<br>EnOcean<br>modules<br>usually send<br>telegrams<br>with their<br>unique 32-bit<br>Chip ID. In<br>the other<br>case, these<br>are the Base<br>ID of the<br>EnOcean<br>device. |

**response Gateway to Client:**

| parameter | datatype                    | value /<br>formatt<br>ing | descript<br>ion |
|-----------|-----------------------------|---------------------------|-----------------|
| header    | <b>object</b>               | { }                       |                 |
| states    | <b>array of<br/>objects</b> | [{ }]                     |                 |

|  |                                  |                  |                  |  |
|--|----------------------------------|------------------|------------------|--|
|  | 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. Must be unique.  |
|  | <i>physicalDevice (optional)</i> | <i>string</i>    |                  | <i>User-assigned name to group actuators and sensors</i>   |
|  | functions                        | array of objects | [{}]             |  |



|  |                           |                |                               |   |
|--|---------------------------|----------------|-------------------------------|---|
|  | key                       | string         |                               | Technical key of function .   |
|  | <i>channel (optional)</i> | <i>integer</i> |                               | <i>Channel if supported by this device.</i>                             |
|  | value                     | float          |                               | Value of the function .   |
|  | <i>unit (optional)</i>    | <i>string</i>  | <b>[unit]</b>                 | <i>Unit ISO.</i>  |
|  | <i>meaning (optional)</i> | <i>string</i>  |                               | <i>Description of value in English.</i>                                 |
|  | timesta mp                | string         | yyyy-mm-ddT hh:mm:ss.sss+hhmm | Timestamp function value was received in case of last state functions.  |
|  | age                       | integer        |                               | Age of function value in case of last state functions in milliseconds . |

**Example:**

Here we see the actual state of our example device. As you can see, the gateway stored all function states received in different telegrams. Because of that, the age of the functions are varying.

#### Basic structure of gateway response:

```
http://hostname:api_port/devices/weatherstation/state

{
  "header" : {
    "httpStatus" : 200,
    "content" : "state",
    "gateway" : "STC-IoT v0.99.0",
    "timestamp" : "2016-05-04T15:15:04.082+0200"
  },
  "state" : {
    "deviceId" : "01842329",
    "friendlyId" : "WeatherStation",
    "functions" : [ {
      "key" : "dawnSensor",
      "value" : 999.00,
      "unit" : "lx",
      "timestamp" : "2016-05-04T15:08:38.381+0200",
      "age" : 385701
    }, {
      "key" : "dayNight",
      "value" : "day",
      "meaning" : "Day",
      "timestamp" : "2016-05-04T15:08:38.381+0200",
      "age" : 385701
    }, {
      "key" : "hemisphere",
      "value" : "north",
      "meaning" : "North",
      "timestamp" : "2016-05-04T15:08:38.185+0200",
      "age" : 385897
    }, {
      "key" : "rainIndication",
      "value" : "noRain",
      "meaning" : "No rain",
      "timestamp" : "2016-05-04T15:08:38.381+0200",
      "age" : 385701
    }, {
      "key" : "sunEast",
      "value" : 1764.71,
      "unit" : "lx",
      "timestamp" : "2016-05-04T15:08:38.185+0200",
      "age" : 385897
    }, {
      "key" : "sunSouth",
      "value" : 2941.18,
      "unit" : "lx",
      "timestamp" : "2016-05-04T15:08:38.185+0200",
      "age" : 385898
    }, {

```

```

    "key" : "sunWest",
    "value" : 588.24,
    "unit" : "lx",
    "timestamp" : "2016-05-04T15:08:38.185+0200",
    "age" : 385898
  }, {
    "key" : "temperature",
    "value" : 25.88,
    "unit" : "°C",
    "timestamp" : "2016-05-04T15:08:38.381+0200",
    "age" : 385702
  }, {
    "key" : "windSpeed",
    "value" : 0.00,
    "unit" : "m/s",
    "timestamp" : "2016-05-04T15:08:38.381+0200",
    "age" : 385702
  } ]
}

```

### 3.2.4.2.1.11 GET /devices/{deviceId}/stream

#### Overview

Get the actual state of one device and continue directly with updates of the device via a telegram stream.

#### Transmitted telegrams of a single device

##### Client to Gateway:

| Resource Path  | HTTP method |
|--|-------------|
| /devices/{ <i>deviceId</i> }/stream?direction={ <i>direction</i> }<br>&delimited={ <i>delimited</i> }&output={ <i>output</i> } | <b>GET</b>  |

##### necessary additional call parameters:

| parameter | valid values | call option | functionality |
|-----------|--------------|-------------|---------------|
|-----------|--------------|-------------|---------------|

|                             |   |                    |  |
|-----------------------------|---|--------------------|--|
| <i>deviceId</i>             | deviceId   friendlyId   | required           | 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. |
| <i>direction (optional)</i> | from   to   both  | default: both      | direction of transport (from device sent / to device sent / both directions )  |
| <i>delimited (optional)</i> | length   lengthBytes   lengthCharacters   newLine   emptyLine | default: emptyLine | <a href="#">Format and delimiter options</a>   |
| <i>output (optional)</i>    | formatted   singleLine  | default: formatted | <a href="#">Format and delimiter options</a>   |

**response Gateway to Client:**

| parameter |          | datatype         | value / formatting | description                                     |
|-----------|----------|------------------|--------------------|---|
| header    |          | object           | {}                 |   |
| states    |          | array of objects | [{}]               |   |
|           | deviceId | string           | xxxx<br>xxxx       | Sender ID of Enocce and device, Enocce and mode |

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | u<br>l<br>e<br>s<br>u<br>s<br>u<br>a<br>l<br>l<br>y<br>s<br>e<br>n<br>d<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>s<br>w<br>i<br>t<br>h<br>t<br>h<br>e<br>i<br>r<br>u<br>n<br>i<br>q<br>u<br>e<br>3<br>2<br>-<br>b<br>i<br>t<br>C<br>h<br>i<br>p<br>l |
|--|--|--|--|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | D<br>·<br>I<br>n<br>t<br>h<br>e<br>o<br>t<br>h<br>e<br>r<br>c<br>a<br>s<br>e<br>,<br>t<br>h<br>e<br>s<br>e<br>a<br>r<br>e<br>t<br>h<br>e<br>B<br>a<br>s<br>e<br>I<br>D<br>o<br>f<br>t<br>h<br>e<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i |
|--|--|--|--|--|

|  |            |        |  |  |
|--|------------|--------|--|--|
|  |            |        |  | c<br>e<br>·  |
|  | friendlyId | string |  | U<br>s<br>e<br>r<br>-<br>a<br>s<br>s<br>i<br>g<br>n<br>e<br>d<br>n<br>a<br>m<br>e<br>o<br>f<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>·<br>M<br>u<br>s<br>t<br>b<br>e<br>u<br>n<br>i<br>q<br>u<br>e<br>· |



|  |                                     |                     |      |  |
|--|-------------------------------------|---------------------|------|--|
|  |                                     |                     |      | U<br>s<br>e<br>r<br>-<br>a<br>s<br>s<br>i<br>g<br>n<br>e<br>d<br>n<br>a<br>m<br>e<br>t<br>o<br>g<br>r<br>o<br>u<br>p<br>a<br>c<br>t<br>u<br>a<br>t<br>o<br>r<br>s<br>a<br>n<br>d<br>s<br>e<br>n<br>s<br>o<br>r<br>s<br>. |
|  | <i>physicalDevice</i><br>(optional) | <i>string</i>       |      |  |
|  | functions                           | array of<br>objects | [{}] |  |

|  |     |        |  |   |
|--|-----|--------|--|---|
|  | key | string |  | T<br>e<br>c<br>h<br>n<br>i<br>c<br>a<br>l<br>k<br>e<br>y<br>o<br>f<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>. |
|--|-----|--------|--|---|

|  |                               |                |  |  |
|--|-------------------------------|----------------|--|--|
|  | <i>channel<br/>(optional)</i> | <i>integer</i> |  | <i>C<br/>h<br/>a<br/>n<br/>n<br/>e<br/>l<br/>i<br/>f<br/>s<br/>u<br/>p<br/>p<br/>o<br/>r<br/>t<br/>e<br/>d<br/>b<br/>y<br/>t<br/>h<br/>i<br/>s<br/>d<br/>e<br/>v<br/>i<br/>c<br/>e<br/>.</i> |
|  | value                         | float          |  | <i>V<br/>a<br/>l<br/>u<br/>e<br/>o<br/>f<br/>t<br/>h<br/>e<br/>f<br/>u<br/>n<br/>c<br/>t<br/>i<br/>o<br/>n<br/>.</i>   |

|  |                                    |               |                    |  |
|--|------------------------------------|---------------|--------------------|--|
|  | <i>unit<br/>(optional)</i>         | <i>string</i> | <i>[unit<br/>]</i> | <i>U<br/>n<br/>i<br/>t<br/>I<br/>S<br/>O<br/>.</i>   |
|  | <i>meanin<br/>g<br/>(optional)</i> | <i>string</i> |                    | <i>D<br/>e<br/>s<br/>c<br/>r<br/>i<br/>p<br/>t<br/>i<br/>o<br/>n<br/>o<br/>f<br/>v<br/>a<br/>l<br/>u<br/>e<br/>i<br/>n<br/>E<br/>n<br/>g<br/>l<br/>i<br/>s<br/>h<br/>.</i> |

|  |                       |               |   |  |
|--|-----------------------|---------------|---|--|
|  | <p>timesta<br/>mp</p> | <p>string</p> | <p>yyyy<br/>-<br/>mm-<br/>ddT<br/>hh:m<br/>m:ss.<br/>sss+<br/>hhm<br/>m</p> | <p>T<br/>i<br/>m<br/>e<br/>s<br/>t<br/>a<br/>m<br/>p<br/>f<br/>u<br/>n<br/>c<br/>t<br/>i<br/>o<br/>n<br/>v<br/>a<br/>l<br/>u<br/>e<br/>w<br/>a<br/>s<br/>r<br/>e<br/>c<br/>e<br/>i<br/>v<br/>e<br/>d<br/>i<br/>n<br/>c<br/>a<br/>s<br/>e<br/>o<br/>f<br/>l<br/>a<br/>s<br/>t<br/>s<br/>t<br/>a<br/>t<br/>e</p> |
|--|-----------------------|---------------|---|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>s<br>. |
|--|--|--|--|--|

|  |     |         |  |   |
|--|-----|---------|--|---|
|  | age | integer |  | A<br>g<br>e<br>o<br>f<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>v<br>a<br>l<br>u<br>e<br>i<br>n<br>c<br>a<br>s<br>e<br>o<br>f<br>l<br>a<br>s<br>t<br>s<br>t<br>a<br>t<br>e<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>s<br>i<br>m<br>i<br>l<br>l |
|--|-----|---------|--|---|

|   |  |               |           |   |
|---|--|---------------|-----------|---|
|   |  |               |           | i<br>s<br>e<br>c<br>o<br>n<br>d<br>s<br>. |
| <i>followed by live telegram stream</i>   |  |               |           |   |
| <i>header</i>                             |  | <b>object</b> | <b>{}</b> |   |
| <i>telegram</i>                           |  | <b>object</b> | <b>{}</b> |   |
| <i>Case 1/2: telegram of known device</i> |  |               |           |   |



|  |          |        |              |   |
|--|----------|--------|--------------|---|
|  | deviceId | string | xxxx<br>xxxx | S<br>e<br>n<br>d<br>e<br>r<br>I<br>D<br>o<br>f<br>E<br>n<br>o<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>,<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>m<br>o<br>d<br>u<br>l<br>e<br>s<br>s<br>u<br>a<br>l<br>l<br>y<br>s<br>e<br>n<br>d<br>t |
|--|----------|--------|--------------|---|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | e<br>l<br>e<br>g<br>r<br>a<br>n<br>s<br>w<br>i<br>t<br>h<br>t<br>h<br>e<br>i<br>r<br>u<br>n<br>i<br>q<br>u<br>e<br>3<br>2<br>-<br>b<br>i<br>t<br>C<br>h<br>i<br>p<br>I<br>D<br>·<br>I<br>n<br>t<br>h<br>e<br>o<br>t<br>h<br>e<br>r<br>c<br>a<br>s<br>e |
|--|--|--|--|--|

|  |  |  |  |   |
|--|--|--|--|---|
|  |  |  |  | ,<br>t<br>h<br>e<br>s<br>e<br>a<br>r<br>e<br>t<br>h<br>e<br>B<br>a<br>s<br>e<br>I<br>D<br>o<br>f<br>t<br>h<br>e<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>. |
|--|--|--|--|---|

|  |            |        |  |   |
|--|------------|--------|--|---|
|  | friendlyId | string |  | User-assigned name of EnOcean device. Must be unique. |
|--|------------|--------|--|---|

|  |                                      |               |  |  |
|--|--------------------------------------|---------------|--|--|
|  | <i>physicalDevice<br/>(optional)</i> | <i>string</i> |  | <i>U<br/>s<br/>e<br/>r<br/>-<br/>a<br/>s<br/>s<br/>i<br/>g<br/>n<br/>e<br/>d<br/>n<br/>a<br/>m<br/>e<br/>t<br/>o<br/>g<br/>r<br/>o<br/>u<br/>p<br/>a<br/>c<br/>t<br/>u<br/>a<br/>t<br/>o<br/>r<br/>s<br/>a<br/>n<br/>d<br/>s<br/>e<br/>n<br/>s<br/>o<br/>r<br/>s<br/>.</i> |
|--|--------------------------------------|---------------|--|--|

|  |           |        |  |   |
|--|-----------|--------|--|---|
|  | timestamp | string | yyyy<br>-<br>mm-<br>ddT<br>hh:m<br>m:ss.<br>sss+<br>hhm<br>m | t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>s<br>s<br>i<br>o<br>n<br>t<br>i<br>m<br>e<br>i<br>n<br>U<br>T<br>C<br>f<br>o<br>r<br>m<br>a<br>t |
|  | direction | string | from<br> <br>to  | d<br>i<br>r<br>e<br>c<br>t<br>i<br>o<br>n<br>o<br>f<br>t<br>r<br>a<br>n   |

|  |           |                  |      |  |
|--|-----------|------------------|------|--|
|  |           |                  |      | s<br>p<br>o<br>r<br>t<br>(<br>f<br>r<br>o<br>m<br>d<br>e<br>v<br>i<br>c<br>e<br>s<br>e<br>n<br>t<br>/<br>t<br>o<br>d<br>e<br>v<br>i<br>c<br>e<br>s<br>e<br>n<br>t<br>) |
|  | functions | array of objects | [{}] |  |

|  |     |        |  |   |
|--|-----|--------|--|---|
|  | key | string |  | T<br>e<br>c<br>h<br>n<br>i<br>c<br>a<br>l<br>k<br>e<br>y<br>o<br>f<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>. |
|--|-----|--------|--|---|



|  |                               |                |  |  |
|--|-------------------------------|----------------|--|--|
|  | <i>channel<br/>(optional)</i> | <i>integer</i> |  | <i>C<br/>h<br/>a<br/>n<br/>n<br/>e<br/>l<br/>i<br/>f<br/>s<br/>u<br/>p<br/>p<br/>o<br/>r<br/>t<br/>e<br/>d<br/>b<br/>y<br/>t<br/>h<br/>i<br/>s<br/>d<br/>e<br/>v<br/>i<br/>c<br/>e<br/>.</i> |
|  | <i>value</i>                  | <i>float</i>   |  | <i>V<br/>a<br/>l<br/>u<br/>e<br/>o<br/>f<br/>t<br/>h<br/>e<br/>f<br/>u<br/>n<br/>c<br/>t<br/>i<br/>o<br/>n<br/>.</i>   |

|  |                                    |               |                    |  |
|--|------------------------------------|---------------|--------------------|--|
|  | <i>unit<br/>(optional)</i>         | <i>string</i> | <i>[unit<br/>]</i> | <i>U<br/>n<br/>i<br/>t<br/>I<br/>S<br/>O<br/>.</i>   |
|  | <i>meanin<br/>g<br/>(optional)</i> | <i>string</i> |                    | <i>D<br/>e<br/>s<br/>c<br/>r<br/>i<br/>p<br/>t<br/>i<br/>o<br/>n<br/>o<br/>f<br/>v<br/>a<br/>l<br/>u<br/>e<br/>i<br/>n<br/>E<br/>n<br/>g<br/>l<br/>i<br/>s<br/>h<br/>.</i> |
|  | telegramInfo                       | object        | {}                 |  |

|  |      |        |                               |   |
|--|------|--------|-------------------------------|---|
|  | data | string | x<br>byte<br>hex<br>valu<br>e | P<br>a<br>y<br>l<br>o<br>a<br>d<br>o<br>f<br>E<br>R<br>P<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>s<br>o<br>r<br>E<br>S<br>P<br>p<br>a<br>c<br>k<br>e<br>t<br>s |
|--|------|--------|-------------------------------|---|

|  |        |         |  |  |
|--|--------|---------|--|--|
|  | status | integer |  | i<br>d<br>e<br>n<br>t<br>i<br>f<br>i<br>e<br>s<br>i<br>f<br>t<br>h<br>e<br>s<br>u<br>b<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>i<br>s<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>t<br>t<br>e<br>d<br>f<br>r<br>o<br>m<br>a<br>r<br>e<br>p<br>e<br>a<br>t |
|--|--------|---------|--|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | e<br>r<br>a<br>n<br>d<br>t<br>h<br>e<br>t<br>y<br>p<br>e<br>o<br>f<br>i<br>n<br>t<br>e<br>g<br>r<br>i<br>t<br>y<br>c<br>o<br>n<br>t<br>r<br>o<br>l<br>m<br>e<br>c<br>h<br>a<br>n<br>i<br>s<br>m<br>u<br>s<br>e<br>d<br>(<br>n<br>o<br>t<br>e<br>:<br>n |
|--|--|--|--|--|

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

o  
t  
p  
r  
e  
s  
e  
n  
t  
i  
n  
a  
R  
P  
S  
t  
e  
l  
e  
g  
r  
a  
m  
)

|  |     |         |  |  |
|--|-----|---------|--|--|
|  | dbm | integer |  | s<br>i<br>g<br>n<br>a<br>l<br>s<br>t<br>r<br>e<br>n<br>g<br>t<br>h<br>o<br>f<br>r<br>e<br>c<br>e<br>i<br>v<br>e<br>d<br>/<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>t<br>t<br>e<br>d<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m |
|--|-----|---------|--|--|

|   |          |        |                               |  |
|---|----------|--------|-------------------------------|--|
|   | rorg     | string | 1<br>byte<br>hex<br>valu<br>e | i<br>d<br>e<br>n<br>t<br>i<br>f<br>i<br>e<br>r<br>f<br>o<br>r<br>s<br>u<br>b<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>t<br>y<br>p<br>e |
| <i>Case 2/2: telegram of unknown device (only if Filtermode is Off)</i> |          |        |                               |  |
|   | deviceId | string | 4<br>byte<br>hex<br>valu<br>e | S<br>e<br>n<br>d<br>e<br>r<br>I<br>D<br>o<br>f<br>E<br>n<br>o<br>c<br>e<br>a<br>n<br>d<br>e<br>v   |



|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | i<br>c<br>e<br>,<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>m<br>o<br>d<br>u<br>l<br>e<br>s<br>u<br>s<br>u<br>a<br>l<br>l<br>y<br>s<br>e<br>n<br>d<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>s<br>w<br>i<br>t<br>h<br>t<br>h<br>e<br>i<br>r<br>u<br>n<br>i |
|--|--|--|--|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | q<br>u<br>e<br>3<br>2<br>-<br>b<br>i<br>t<br>C<br>h<br>i<br>p<br>I<br>D<br>.<br>I<br>n<br>t<br>h<br>e<br>o<br>t<br>h<br>e<br>r<br>c<br>a<br>s<br>e<br>,<br>t<br>h<br>e<br>s<br>e<br>a<br>r<br>e<br>t<br>h<br>e<br>B<br>a<br>s<br>e<br>I<br>D<br>o<br>f |
|--|--|--|--|--|

|  |  |  |  |   |
|--|--|--|--|---|
|  |  |  |  | t<br>h<br>e<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>. |
|--|--|--|--|---|

|  |           |        |  |   |
|--|-----------|--------|--|---|
|  | timestamp | string | yyyy<br>-<br>mm-<br>ddT<br>hh:m<br>m:ss.<br>sss+<br>hhm<br>m | t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>s<br>s<br>i<br>o<br>n<br>t<br>i<br>m<br>e<br>i<br>n<br>U<br>T<br>C<br>f<br>o<br>r<br>m<br>a<br>t |
|--|-----------|--------|--|---|

|  |              |        |              |  |
|--|--------------|--------|--------------|--|
|  | direction    | string | from<br>  to | d<br>i<br>r<br>e<br>c<br>t<br>i<br>o<br>n<br>o<br>f<br>t<br>r<br>a<br>n<br>s<br>p<br>o<br>r<br>t<br>(<br>f<br>r<br>o<br>m<br>G<br>W<br>s<br>e<br>n<br>t<br>o<br>r<br>r<br>e<br>c<br>e<br>i<br>v<br>e<br>d<br>) |
|  | telegramInfo | object | {}           |  |

|  |      |        |                               |  |
|--|------|--------|-------------------------------|--|
|  | data | string | x<br>byte<br>hex<br>valu<br>e | P<br>a<br>y<br>l<br>o<br>a<br>d<br>o<br>f<br>E<br>R<br>P<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>s<br>o<br>r<br>E<br>S<br>P<br>p<br>a<br>c<br>k<br>e<br>t<br>s<br>. |
|--|------|--------|-------------------------------|--|

|  |        |         |  |  |
|--|--------|---------|--|--|
|  | status | integer |  | i<br>d<br>e<br>n<br>t<br>i<br>f<br>i<br>e<br>s<br>i<br>f<br>t<br>h<br>e<br>s<br>u<br>b<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>i<br>s<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>t<br>t<br>e<br>d<br>f<br>r<br>o<br>m<br>a<br>r<br>e<br>p<br>e<br>a<br>t |
|--|--------|---------|--|--|

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

e  
r  
a  
n  
d  
t  
h  
e  
t  
y  
p  
e  
o  
f  
i  
n  
t  
e  
g  
r  
i  
t  
y  
c  
o  
n  
t  
r  
o  
l  
m  
e  
c  
h  
a  
n  
i  
s  
m  
u  
s  
e  
d  
.  
(  
n  
o  
t  
e  
:



|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

n  
o  
t  
p  
r  
e  
s  
e  
n  
t  
i  
n  
a  
R  
P  
S  
t  
e  
l  
e  
g  
r  
a  
m  
)

|  |     |         |  |  |
|--|-----|---------|--|--|
|  | dbm | integer |  | s<br>i<br>g<br>n<br>a<br>l<br>s<br>t<br>r<br>e<br>n<br>g<br>t<br>h<br>o<br>f<br>r<br>e<br>c<br>e<br>i<br>v<br>e<br>d<br>/<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>t<br>t<br>e<br>d<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m |
|--|-----|---------|--|--|

|  |      |        |                               |  |
|--|------|--------|-------------------------------|--|
|  | rorg | string | 1<br>byte<br>hex<br>valu<br>e | i<br>d<br>e<br>n<br>t<br>i<br>f<br>i<br>e<br>r<br>f<br>o<br>r<br>s<br>u<br>b<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>t<br>y<br>p<br>e |
|--|------|--------|-------------------------------|--|

**Example:**

In that example we received various telegrams from a weatherstation.

Basic structure of gateway response:

```
http://hostname:api_port/devices/weatherstation/stream

{
  "header" : {
    "httpStatus" : 200,
    "content" : "states",
    "timestamp" : "2016-05-04T15:24:05.453+0200"
  },
  "states" : [ {
    "deviceId" : "01842329",
    "friendlyId" : "WeatherStation",
    "functions" : [ {
      "key" : "dawnSensor",
```

```
    "value" : 999.00,  
    "unit" : "lx",  
    "timestamp" : "2016-05-04T15:18:38.909+0200",  
    "age" : 326544  
  }, {  
    "key" : "dayNight",  
    "value" : "day",  
    "meaning" : "Day",  
    "timestamp" : "2016-05-04T15:18:38.909+0200",  
    "age" : 326544  
  }, {  
    "key" : "hemisphere",  
    "value" : "north",  
    "meaning" : "North",  
    "timestamp" : "2016-05-04T15:18:38.709+0200",  
    "age" : 326744  
  }, {  
    "key" : "rainIndication",  
    "value" : "noRain",  
    "meaning" : "No rain",  
    "timestamp" : "2016-05-04T15:18:38.909+0200",  
    "age" : 326544  
  }, {  
    "key" : "sunEast",  
    "value" : 1764.71,  
    "unit" : "lx",  
    "timestamp" : "2016-05-04T15:18:38.709+0200",  
    "age" : 326744  
  }, {  
    "key" : "sunSouth",  
    "value" : 2941.18,  
    "unit" : "lx",  
    "timestamp" : "2016-05-04T15:18:38.709+0200",  
    "age" : 326744  
  }, {  
    "key" : "sunWest",  
    "value" : 588.24,  
    "unit" : "lx",  
    "timestamp" : "2016-05-04T15:18:38.709+0200",  
    "age" : 326744  
  }, {  
    "key" : "temperature",  
    "value" : 25.88,  
    "unit" : "°C",  
    "timestamp" : "2016-05-04T15:18:38.909+0200",  
    "age" : 326545  
  }, {  
    "key" : "windSpeed",  
    "value" : 0.00,  
    "unit" : "m/s",  
    "timestamp" : "2016-05-04T15:18:38.909+0200",  
    "age" : 326545  
  } ]  
} ]  
{  
  "header" : {
```

```

    "content" : "telegram",
    "timestamp" : "2016-05-04T15:24:44.653+0200"
  },
  "telegram" : {
    "deviceId" : "01842329",
    "friendlyId" : "WeatherStation",
    "timestamp" : "2016-05-04T15:24:44.653+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "sunWest",
      "value" : 0.00,
      "unit" : "lx"
    }, {
      "key" : "sunSouth",
      "value" : 588.24,
      "unit" : "lx"
    }, {
      "key" : "sunEast",
      "value" : 588.24,
      "unit" : "lx"
    }, {
      "key" : "hemisphere",
      "value" : "north",
      "meaning" : "North"
    } ],
    "telegramInfo" : {
      "data" : "00010128",
      "status" : "0",
      "dbm" : -82,
      "rorg" : "A5"
    }
  }
}
{
  "header" : {
    "content" : "telegram",
    "timestamp" : "2016-05-04T15:28:56.379+0200"
  },
  "telegram" : {
    "deviceId" : "01842329",
    "friendlyId" : "WeatherStation",
    "timestamp" : "2016-05-04T15:28:56.379+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "sunWest",
      "value" : 0.00,
      "unit" : "lx"
    }, {
      "key" : "sunSouth",
      "value" : 588.24,
      "unit" : "lx"
    }, {
      "key" : "sunEast",
      "value" : 588.24,
      "unit" : "lx"
    }, {
      "key" : "hemisphere",

```

```

        "value" : "north",
        "meaning" : "North"
    } ],
    "telegramInfo" : {
        "data" : "00010128",
        "status" : "0",
        "dbm" : -86,
        "rorg" : "A5"
    }
}
}
{
  "header" : {
    "content" : "telegram",
    "timestamp" : "2016-05-04T15:28:56.580+0200"
  },
  "telegram" : {
    "deviceId" : "01842329",
    "friendlyId" : "WeatherStation",
    "timestamp" : "2016-05-04T15:28:56.580+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "dawnSensor",
      "value" : 999.00,
      "unit" : "lx"
    }, {
      "key" : "temperature",
      "value" : 25.88,
      "unit" : "°C"
    }, {
      "key" : "windSpeed",
      "value" : 0.00,
      "unit" : "m/s"
    }, {
      "key" : "dayNight",
      "value" : "day",
      "meaning" : "Day"
    }, {
      "key" : "rainIndication",
      "value" : "noRain",
      "meaning" : "No rain"
    } ],
    "telegramInfo" : {
      "data" : "FF8C0018",
      "status" : "0",
      "dbm" : -86,
      "rorg" : "A5"
    }
  }
}
}

```

### 3.2.4.2.1.12 GET /devices/{deviceId}/telegrams

## Overview

Show the last 1000 available or historic telegrams of a single device in receiving order depending on the direction of transport (from device sent / to device sent / both directions ).

**Last transmitted telegrams of a single device**

**Client to Gateway:**

| Resource Path  | HTTP method |
|--|-------------|
| <code>/devices/{<i>deviceId</i>}/telegrams?direction={<i>direction</i>}</code> | <b>GET</b>  |

**necessary additional call parameters:**

| parameter                   | valid values          | call option   | functionality  |
|-----------------------------|-----------------------|---------------|--|
| <i>deviceId</i>             | deviceID   friendlyId | required      | 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. |
| <i>direction</i> (optional) | from   to   both      | default: both | direction of transport (from device sent / to device sent / both directions )  |

**response Gateway to Client:**

| parameter | datatype | value / formatting | description |
|-----------|----------|--------------------|-------------|
|-----------|----------|--------------------|-------------|

|   |          |                        |                  |   |
|---|----------|------------------------|------------------|---|
| header                                    |          | <b>object</b>          | { }              |   |
| telegrams                                 |          | <b>array of object</b> | [{ }]            |   |
| <i>Case 1/2: telegram of known device</i> |          |                        |                  |   |
|   | deviceid | string                 | 4 byte hex value | Sender ID 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-assign ed name of EnOcean device . Must be unique .                  |
|  | <i>physicalDevice (optional)</i> | <i>string</i>    |                                 | <i>User-assign ed name to group actuators and sensors.</i>                |
|  | timestamp                        | string           | yyyy-mm-ddT hh:mm:ss.sss + hhmm | telegram transmission time in UTC format                                  |
|  | direction                        | string           | from   to                       | direction of transport ( <i>from</i> device sent / <i>to</i> device sent) |
|  | functions                        | array of objects | [{}]                            |   |

|  |                           |                |                  |   |
|--|---------------------------|----------------|------------------|---|
|  | key                       | string         |                  | Technical key of function.                  |
|  | <i>channel (optional)</i> | <i>integer</i> |                  | <i>Channel if supported by this device.</i> |
|  | value                     | float          |                  | Value of the function.                      |
|  | <i>unit (optional)</i>    | <i>string</i>  | <i>[unit]</i>    | <i>Unit ISO.</i>                            |
|  | <i>meaning (optional)</i> | <i>string</i>  |                  | <i>Description of value in English.</i>     |
|  | telegramInfo              | object         | {}               |   |
|  | 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 RPS telegram) |
|  | dbm    | integer |                  | signal strength of received / transmitted telegram  |
|  | rorg   | string  | 1 byte hex value | identifier for subtelegram type   |
| Case 2/2: telegram of unknown device (only if Filtermode is Off) |        |         |                  |   |

|  |           |        |                                 |   |
|--|-----------|--------|---------------------------------|---|
|  | deviceid  | string | 4 byte hex value                | Sender ID 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. |
|  | timestamp | string | yyyy-mm-ddT hh:mm:ss.sss + hhmm | telegram transmission time in UTC format  |
|  | direction | string | from                            | direction of transport ( <i>from</i> device sent / <i>to</i> device sent)   |

|  | telegramInfo | object  | {}               |   |
|--|--------------|---------|------------------|---|
|  | 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 RPS telegram) |
|  | dbm          | integer |                  | signal strength of received / transmitted telegram  |

|  |      |        |                  |                                 |
|--|------|--------|------------------|---------------------------------|
|  | rorg | string | 1 byte hex value | identifier for subtelegram type |
|--|------|--------|------------------|---------------------------------|

**Example:**

As a example historic telegrams of the weatherstation are shown.

**Basic structure of gateway response:**

```
http://hostname:api_port/devices/weatherstation/telegrams
```

```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "telegrams",
    "gateway" : "STC-IoT v0.99.0",
    "timestamp" : "2016-05-04T16:01:12.790+0200"
  },
  "telegrams" : [ {
    "deviceId" : "01842329",
    "friendlyId" : "WeatherStation",
    "timestamp" : "2016-05-04T12:17:32.299+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "sunWest",
      "value" : 588.24,
      "unit" : "lx"
    }, {
      "key" : "sunSouth",
      "value" : 2941.18,
      "unit" : "lx"
    }, {
      "key" : "sunEast",
      "value" : 2941.18,
      "unit" : "lx"
    }, {
      "key" : "hemisphere",
      "value" : "north",
      "meaning" : "North"
    }
  ],
    "telegramInfo" : {
      "data" : "01050528",
      "status" : "0",
      "dbm" : -83,
      "rorg" : "A5"
    }
  }, {
    "deviceId" : "01842329",
    "friendlyId" : "WeatherStation",
    "timestamp" : "2016-05-04T12:17:32.505+0200",
    "direction" : "from",
```

```
"functions" : [ {
  "key" : "dawnSensor",
  "value" : 999.00,
  "unit" : "lx"
}, {
  "key" : "temperature",
  "value" : 24.94,
  "unit" : "°C"
}, {
  "key" : "windSpeed",
  "value" : 0.00,
  "unit" : "m/s"
}, {
  "key" : "dayNight",
  "value" : "day",
  "meaning" : "Day"
}, {
  "key" : "rainIndication",
  "value" : "noRain",
  "meaning" : "No rain"
} ],
"telegramInfo" : {
  "data" : "FF8A0018",
  "status" : "0",
  "dbm" : -83,
  "rorg" : "A5"
}
}, {
  "deviceId" : "01842329",
  "friendlyId" : "WeatherStation",
  "timestamp" : "2016-05-04T12:18:52.629+0200",
  "direction" : "from",
  "functions" : [ {
    "key" : "sunWest",
    "value" : 0.00,
    "unit" : "lx"
  }, {
    "key" : "sunSouth",
    "value" : 1764.71,
    "unit" : "lx"
  }, {
    "key" : "sunEast",
    "value" : 588.24,
    "unit" : "lx"
  }, {
    "key" : "hemisphere",
    "value" : "north",
    "meaning" : "North"
  } ],
  "telegramInfo" : {
    "data" : "00030128",
    "status" : "0",
    "dbm" : -82,
    "rorg" : "A5"
  }
}
}]
}
```

### 3.2.4.2.2 PUT

#### Overview

The /devices resource make EnOcean devices act.

#### Single device

- [PUT /devices/{deviceId}/state](#) Change the state of a device

#### 3.2.4.2.2.1 PUT /devices/{deviceId}/state

#### Overview

Change a state of a single device

Client to Gateway:

| URL                                | call method |
|------------------------------------|-------------|
| /devices/{ <i>deviceId</i> }/state | PUT         |

necessary additional call parameters:

| parameter       | values                | call option | functionality  |
|-----------------|-----------------------|-------------|--|
| <i>deviceId</i> | deviceId   friendlyId | required    | 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. |



+ additionally in json notation (Call Body):

| parameter |                           | datatype         | value / formatting | description                                 |
|-----------|---------------------------|------------------|--------------------|---|
| state     |                           | <b>object</b>    | <b>{}</b>          |   |
|           | functions                 | array of objects | <b>[{}]</b>        |   |
|           | key                       | string           |                    | Technical key of function .                 |
|           | <i>channel (optional)</i> | <i>integer</i>   |                    | <i>Channel if supported by this device.</i> |
|           | value                     | float            |                    | Value of the function .                     |

All functions of a functionGroup have to be set in one PUT-request. For many functions default values have been introduced. That means, if you don't set a function and a default value is available, the gateway will auto-complete the telegram with associated default values.

The function group described in the profile, must have the attribute *"direction": "to"* or *"direction": "both"*.

If the request is incomplete the gateways responses with error code 3010 and message "incomplete request - function(s) missing: XYZ,ABC".

If several function groups exist for a device, the gateway searches for the corresponding one depending on the functions send. If this fails, the gateway responses with error code 3011: " *no matching group found*".

**Response Gateway to Client:**

| parameter | datatype       |  |  |
|-----------|----------------|--|--|
| header    | message header |  |  |

|  |            |         |  |   |
|--|------------|---------|--|---|
|  | httpStatus | integer |  | HTTP state code (200 = OK), otherwise refer to eMSG |
|  | gateway    | string  |  | current firmware version                            |
|  | timestamp  | string  | yyyy-mm-ddT<br>hh:mm:ss<br>.sss+<br>hhmm | transmission time in UTC format                     |

**Response Gateway to Client (in case of error):**

| parameter |            | datatype       |  |   |
|-----------|------------|----------------|--|---|
| header    |            | message header |  |   |
|           | httpStatus | integer        | e.g. 400                                 | HTTP state code (200 = OK), otherwise refer to eMSG |
|           | code       | integer        | e.g. 3011                                | error number  |
|           | message    | string         | e.g. "no matching group found"           |   |
|           | gateway    | string         | e.g. STC-IoT v0.99.1                     | current firmware version                            |
|           | timestamp  | string         | yyyy-mm-ddT<br>hh:mm:ss<br>.sss+<br>hhmm | transmission time in UTC format                     |

**example:**

## Basic structure of outgoing message(s) based on d2-01-08:

"Actuator Set Output"

**PUT** [http://hostname:api\\_port/devices/switchActuator/state](http://hostname:api_port/devices/switchActuator/state)

```
{
  "state" : {
    "functions" : [
      {
        "key" : "switch",
        "value" : "on"
      }
    ]
  }
}
```

"Configure Actuator"

**PUT** [http://hostname:api\\_port/devices/switchActuator/state](http://hostname:api_port/devices/switchActuator/state)

```
{
  "state" : {
    "functions" : [
      {
        "key" : "defaultState",
        "value": "previousState",
      },
      {
        "key" : "localControl",
        "value": "off",
      },
      {
        "key" : "overcurrentSwitchOffMode",
        "value": "staticOff",
      },
      {
        "key" : "overcurrentSwitchOffReset",
        "value": "false",
      },
      {
        "key" : "taughtInDevices",
        "value": "on",
      },
      {
        "key" : "userInterfaceIndication",
        "value": "day",
      }
    ]
  }
}
```

Simplified "Configure Actuator" command due to default values. The same result we would get with a http-request:

```
"Configure Actuator"

PUT http://hostname:api\_port/devices/switchActuator/state

{
  "state" : {
    "functions" : [
      {
        "key" : "localControl",
        "value": "off",
      }
    ]
  }
}
```

### 3.2.4.3 Profiles

#### Overview

The /profiles resource is helpful, when you want to know all supported profiles of the gateway and to get a detailed JSON description of a supported profile.

#### All profiles

- [GET /profiles](#)

#### Single profile

- [GET /profiles/{eepId}](#)
- [GET /profiles/{eepId}.html](#)
- [GET /profiles/{eepId}.pdf](#)
- [GET /profiles/{eepId}?variation={Manufacturer} {Product}](#)

#### 3.2.4.3.1 GET

#### Overview

The /profiles resource is helpful, when you want to know the supported devices of the gateway and the detailed JSON description of a supported profile.

### All profiles

- [GET /profiles](#)

### Single profile

- [GET /profiles/{eepId}](#)
- [GET /profiles/{eepId}.html](#)
- [GET /profiles/{eepId}.pdf](#)

#### 3.2.4.3.1.1 GET /profiles

### Profile Overview

Lists overview of all supported EEP profiles and manufacturer variations

#### Client to Gateway:

| Resource Path | HTTP method |
|---------------|-------------|
| /profiles     | GET         |

no additional call parameters are necessary

#### Response Gateway to Client:

| parameter | datatype         | value / formatting | description |
|-----------|------------------|--------------------|-------------|
| header    | object           | {}                 |             |
| profiles  | array of objects | [{}]               |             |

|  |            |           |                  |                      |   |
|--|------------|-----------|------------------|----------------------|---|
|  | eep        |           | string           | xx-xx-xx             | EnOcean Equipment Profiles , definition of EnOcean radio telegram structure     |
|  | title      |           | string           |                      | short description of eep profile  |
|  | variations |           | array of objects | [{}]                 |   |
|  |            | variation | string           | Manufacturer_Product | Variation identifier. Mainly composed of Manufacturer and Product that uses it. |
|  |            | version   | float            |                      | version identifier  |

|  |  |           |        |             |   |
|--|--|-----------|--------|-------------|---|
|  |  | direction | string | from   both | direction of transport ( <i>from device sent / to device sent / both directions</i> ) |
|--|--|-----------|--------|-------------|---|

**example:**

## Basic structure of gateway response:

```

http://hostname:api_port/profiles

{
  "header" : {
    "httpStatus" : 200,
    "content" : "profiles",
    "gateway" : "STC-IoT v0.99.1",
    "timestamp" : "2016-05-09T15:41:13.487+0200"
  },
  "profiles" : [ {
    "eep" : "A5-02-01",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-02",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-03",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-04",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  } ]
}

```

```
    } ]
  }, {
    "eep" : "A5-02-05",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-06",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-07",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-08",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-09",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-0A",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-0B",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-10",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  } ]
```



```
}, {
  "eep" : "A5-02-11",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-02-12",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-02-13",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-02-14",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-02-15",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-02-16",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-02-17",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-02-18",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
```

```

    "eep" : "A5-02-19",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-1A",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-1B",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-20",
    "title" : "Temperature Sensors, 10 Bit Temperature Sensor",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-02-30",
    "title" : "Temperature Sensors, 10 Bit Temperature Sensor",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-04-01",
    "title" : "Temperature and Humidity Sensor, Range 0°C to 50°C",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    }, {
      "variation" : "to",
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-04-02",
    "title" : "Temperature and Humidity Sensor, Range -20°C to 50°C",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    }, {
      "variation" : "to",
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {

```

```

    "eep" : "A5-04-03",
    "title" : "Temperature and Humidity Sensor, Range -20°C
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-05-01",
    "title" : "Barometric Sensor, Range 500 to 1150 hPa",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-06-01",
    "title" : "Light Sensor, Range 300lx to 60.000lx",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ], {
      "variation" : "",
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-06-02",
    "title" : "Light Sensor, Range 0lx to 1.020lx",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ], {
      "variation" : "",
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-06-03",
    "title" : "Light Sensor, 10-bit measurement (1-Lux resol
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-07-01",
    "title" : "Occupancy Sensor, Occupancy with Supply volta
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-07-02",
    "title" : "Occupancy Sensor, Occupancy with Supply volta
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {

```

```
"eep" : "A5-07-03",
"title" : "Occupancy Sensor, Occupancy with Supply volta
"variations" : [ {
  "direction" : "from",
  "version" : 1.0
} ]
}, {
  "eep" : "A5-08-01",
  "title" : "Light, Temperature and Occupancy Sensor, Rang
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  }, {
    "variation" : "Eltako_FABH65S,FBH65B,FBH65S,FBH65TFB",
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-08-02",
  "title" : "Light, Temperature and Occupancy Sensor, Rang
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-08-03",
  "title" : "Light, Temperature and Occupancy Sensor, Rang
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-09-02",
  "title" : "Gas Sensor, CO-Sensor 0 ppm to 1020 ppm",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-09-04",
  "title" : "Gas Sensor, CO2 Sensor",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-09-05",
  "title" : "Gas Sensor, VOC Sensor",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-09-06",
  "title" : "Gas Sensor, Radon",
  "variations" : [ {
    "direction" : "from",
```

```

    "version" : 1.0
  } ]
}, {
  "eep" : "A5-09-07",
  "title" : "Gas Sensor, Particles",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-09-08",
  "title" : "Gas Sensor, Pure CO2 Sensor",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-09-09",
  "title" : "Gas Sensor, Pure CO2 Sensor with Power Failur
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-09-0A",
  "title" : "Gas Sensor, Hydrogen Gas Sensor",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-01",
  "title" : "Room Operating Panel, Temperature Sensor, Set
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-02",
  "title" : "Room Operating Panel, Temperature Sensor, Set
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-03",
  "title" : "Room Operating Panel, Temperature Sensor, Set
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ], {
    "variation" : "Afriso_RoomTemperatureSensorFT",
    "direction" : "from",
    "version" : 1.0
  }, {
    "variation" : "Thermokon_SR06LCD",
    "direction" : "from",

```

```
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-04",
  "title" : "Room Operating Panel, Temperature Sensor, Set
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-05",
  "title" : "Room Operating Panel, Temperature Sensor, Set
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-06",
  "title" : "Room Operating Panel, Temperature Sensor, Set
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  }, {
    "variation" : "",
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-07",
  "title" : "Room Operating Panel, Temperature Sensor, Fan
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-08",
  "title" : "Room Operating Panel, Temperature Sensor, Fan
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-09",
  "title" : "Room Operating Panel, Temperature Sensor, Fan
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-0A",
  "title" : "Room Operating Panel, Temperature Sensor, Set
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-0B",
```

```
    "title" : "Room Operating Panel, Temperature Sensor and
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-0C",
    "title" : "Room Operating Panel, Temperature Sensor and
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-0D",
    "title" : "Room Operating Panel, Temperature Sensor and
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-10",
    "title" : "Room Operating Panel, Temperature and Humidit
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-11",
    "title" : "Room Operating Panel, Temperature and Humidit
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-12",
    "title" : "Room Operating Panel, Temperature and Humidit
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    }, {
      "variation" : "Afriso_RoomTemperatureSensorFTF",
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-13",
    "title" : "Room Operating Panel, Temperature and Humidit
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-14",
    "title" : "Room Operating Panel, Temperature and Humidit
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
```

```
    } ]
  }, {
    "eep" : "A5-10-15",
    "title" : "Room Operating Panel, 10 Bit Temperature Sens
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-16",
    "title" : "Room Operating Panel, 10 Bit Temperature Sens
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-17",
    "title" : "Room Operating Panel, 10 Bit Temperature Sens
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-18",
    "title" : "Room Operating Panel, Illumination, Temperatu
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-19",
    "title" : "Room Operating Panel, Humidity, Temperature S
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-1A",
    "title" : "Room Operating Panel, Supply voltage monitor,
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-1B",
    "title" : "Room Operating Panel, Supply Voltage Monitor,
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-10-1C",
    "title" : "Room Operating Panel, Illumination, Illuminat
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  } ]
```



```
}, {
  "eep" : "A5-10-1D",
  "title" : "Room Operating Panel, Humidity, Humidity Set
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-1E",
  "title" : "Room Operating Panel, Humidity, Humidity Set
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-1F",
  "title" : "Room Operating Panel, Temperature Sensor, Set
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-20",
  "title" : "Room Operating Panel, Temperature and Set Poi
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-10-21",
  "title" : "Room Operating Panel, Temperature, Humidity a
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-11-01",
  "title" : "Controller Status, Lighting Controller",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-11-03",
  "title" : "Controller Status, Blind Status",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-12-00",
  "title" : "Automated Meter Reading (AMR), Counter",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
```

```
"eep" : "A5-12-01",
"title" : "Automated Meter Reading (AMR), Electricity",
"variations" : [ {
  "direction" : "from",
  "version" : 1.0
}, {
  "variation" : "Eltako_FSS12,FWZ12,FWZ61",
  "direction" : "from",
  "version" : 1.0
}, {
  "variation" : "Pressac_CTClamp",
  "direction" : "from",
  "version" : 1.0
} ]
}, {
  "eep" : "A5-12-02",
  "title" : "Automated Meter Reading (AMR), Gas",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-12-03",
  "title" : "Automated Meter Reading (AMR), Water",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-13-01",
  "title" : "Environmental Applications, Sun Intensity",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-13-02",
  "title" : "Environmental Applications, Sun Intensity",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-13-03",
  "title" : "Environmental Applications, Date Exchange",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-13-04",
  "title" : "Environmental Applications, Time and Day Exch
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
```

```
"eep" : "A5-13-05",
"title" : "Environmental Applications, Direction Exchange",
"variations" : [ {
  "direction" : "from",
  "version" : 1.0
} ]
}, {
  "eep" : "A5-13-06",
  "title" : "Environmental Applications, Geographic Positioning",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-13-10",
  "title" : "Environmental Applications, Sun position and direction",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-14-01",
  "title" : "Multi-Func Sensor, Single Input Contact (Wind Speed)",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-14-02",
  "title" : "Multi-Func Sensor, Single Input Contact (Wind Direction)",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-14-03",
  "title" : "Multi-Func Sensor, Single Input Contact (Wind Speed)",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-14-04",
  "title" : "Multi-Func Sensor, Single Input Contact (Wind Direction)",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-14-05",
  "title" : "Multi-Func Sensor, Vibration/Tilt, Supply voltage",
  "variations" : [ {
    "direction" : "from",
    "version" : 1.0
  } ]
}, {
  "eep" : "A5-14-06",
```

```

    "title" : "Multi-Func Sensor, Vibration/Tilt, Illuminati
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-20-01",
    "title" : "HVAC Components, Battery Powered Actuator",
    "variations" : [ {
      "direction" : "both",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-20-02",
    "title" : "HVAC Components, Basic Actuator",
    "variations" : [ {
      "direction" : "both",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-20-03",
    "title" : "HVAC Components, Line powered Actuator",
    "variations" : [ {
      "direction" : "both",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-20-04",
    "title" : "HVAC Components, Heating Radiator Valve Actua
    "variations" : [ {
      "direction" : "both",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-30-01",
    "title" : "Digital Input, Single Input Contact, Battery
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-30-02",
    "title" : "Digital Input, Single Input Contact",
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-30-03",
    "title" : "Digital Input, 4 Digital Inputs, Wake and Tem
    "variations" : [ {
      "direction" : "from",
      "version" : 1.0
    } ]
  }, {
    "eep" : "A5-30-04",
    "title" : "Digital Input, 3 Digital Inputs, 1 Digital In

```

```
"variations" : [ {
  "direction" : "from",
  "version" : 1.0
} ]
}, {
  "eep" : "A5-38-08",
  "title" : "Central Command, Gateway",
  "variations" : [ {
    "direction" : "both",
    "version" : 1.0
  } ]
}, {
  "eep" : "D2-01-00",
  "title" : "Electronic switches and dimmers with Energy M
  "variations" : [ {
    "direction" : "both",
    "version" : 1.0
  } ]
}, {
  "eep" : "D2-01-01",
  "title" : "Electronic switches and dimmers with Energy M
  "variations" : [ {
    "direction" : "both",
    "version" : 1.0
  } ]
}, {
  "eep" : "D2-01-02",
  "title" : "Electronic switches and dimmers with Energy M
  "variations" : [ {
    "direction" : "both",
    "version" : 1.0
  } ]
}, {
  "eep" : "D2-01-03",
  "title" : "Electronic switches and dimmers with Energy M
  "variations" : [ {
    "direction" : "both",
    "version" : 1.0
  } ]
}, {
  "eep" : "D2-01-04",
  "title" : "Electronic switches and dimmers with Energy M
  "variations" : [ {
    "direction" : "both",
    "version" : 1.0
  } ]
}, {
  "eep" : "D2-01-05",
  "title" : "Electronic switches and dimmers with Energy M
  "variations" : [ {
    "direction" : "both",
    "version" : 1.0
  } ]
}, {
  "eep" : "D2-01-06",
  "title" : "Electronic switches and dimmers with Energy M
  "variations" : [ {
```

```
        "direction" : "both",
        "version" : 1.0
    } ]
}, {
    "eep" : "D2-01-07",
    "title" : "Electronic switches and dimmers with Energy M
    "variations" : [ {
        "direction" : "both",
        "version" : 1.0
    } ]
}, {
    "eep" : "D2-01-08",
    "title" : "Electronic switches and dimmers with Energy M
    "variations" : [ {
        "direction" : "both",
        "version" : 1.0
    } ]
}, {
    "eep" : "D2-01-09",
    "title" : "Electronic switches and dimmers with Energy M
    "variations" : [ {
        "direction" : "both",
        "version" : 1.0
    }, {
        "variation" : "Permundo_psc234",
        "direction" : "both",
        "version" : 1.0
    } ]
}, {
    "eep" : "D2-01-0A",
    "title" : "Electronic switches and dimmers with Energy M
    "variations" : [ {
        "direction" : "both",
        "version" : 1.0
    }, {
        "variation" : "Nodon_SmartPlugASP-2-1-10",
        "direction" : "both",
        "version" : 1.0
    } ]
}, {
    "eep" : "D2-01-0B",
    "title" : "Electronic switches and dimmers with Energy M
    "variations" : [ {
        "direction" : "both",
        "version" : 1.0
    } ]
}, {
    "eep" : "D2-01-10",
    "title" : "Electronic switches and dimmers with Energy M
    "variations" : [ {
        "direction" : "both",
        "version" : 1.0
    } ]
}, {
    "eep" : "D2-01-11",
    "title" : "Electronic switches and dimmers with Energy M
    "variations" : [ {
```

```

        "direction" : "both",
        "version" : 1.0
    } ]
}, {
    "eep" : "D2-05-00",
    "title" : "Blinds Control for Position and Angle",
    "variations" : [ {
        "direction" : "both",
        "version" : 1.0
    } ]
}, {
    "eep" : "D2-06-01",
    "title" : "Multisensor Window Handle, Alarm, Position Se
    "variations" : [ {
        "direction" : "both",
        "version" : 1.0
    } ]
}, {
    "eep" : "D2-A0-01",
    "title" : "Standard Valve, Valve Control",
    "variations" : [ {
        "direction" : "both",
        "version" : 1.0
    } ]
}, {
    "eep" : "D5-00-01",
    "title" : "Contacts and Switches, Single Input Contact",
    "variations" : [ {
        "direction" : "from",
        "version" : 1.0
    } ]
}, {
    "eep" : "F6-02-01",
    "title" : "Rocker Switch, 2 Rocker, Light and Blind Cont
    "variations" : [ {
        "direction" : "from",
        "version" : 1.0
    }, {
        "variation" : "Eltako_FRW",
        "direction" : "from",
        "version" : 1.0
    }, {
        "variation" : "VariousManufacturers_GeneralSwitchActua
        "direction" : "both",
        "version" : 1.0
    }, {
        "variation" : "VariousManufacturers_SmokeDetector",
        "direction" : "from",
        "version" : 1.0
    } ]
}, {
    "eep" : "F6-02-02",
    "title" : "Rocker Switch, 2 Rocker, Light and Blind Cont
    "variations" : [ {
        "direction" : "from",
        "version" : 1.0
    }, {

```

```

        "variation" : "Eltako_FRW",
        "direction" : "from",
        "version" : 1.0
    }, {
        "variation" : "VariousManufacturers_GeneralSwitchActua
        "direction" : "both",
        "version" : 1.0
    }, {
        "variation" : "VariousManufacturers_SmokeDetector",
        "direction" : "from",
        "version" : 1.0
    } ]
}, {
    "eep" : "F6-02-03",
    "title" : "Rocker Switch, 2 Rocker, Light and Blind Cont
    "variations" : [ {
        "direction" : "from",
        "version" : 1.0
    }, {
        "variation" : "Eltako_FRW",
        "direction" : "from",
        "version" : 1.0
    }, {
        "variation" : "VariousManufacturers_GeneralSwitchActua
        "direction" : "both",
        "version" : 1.0
    }, {
        "variation" : "VariousManufacturers_SmokeDetector",
        "direction" : "from",
        "version" : 1.0
    } ]
}, {
    "eep" : "F6-03-01",
    "title" : "Rocker Switch, 4 Rocker, Light and Blind Cont
    "variations" : [ {
        "direction" : "from",
        "version" : 1.0
    } ]
}, {
    "eep" : "F6-05-01",
    "title" : "Detectors, Liquid Leakage Sensor (mechanic ha
    "variations" : [ {
        "direction" : "from",
        "version" : 1.0
    } ]
}, {
    "eep" : "F6-10-00",
    "title" : "Mechanical Handle, Window Handle",
    "variations" : [ {
        "direction" : "from",
        "version" : 1.0
    } ]
} ]
}

```



3.2.4.3.1.2 GET /profiles/{eepld}

**Profile detail**

Get detailed informations of a standard or a vendor specific EEP variation functionality / functions

**Client to Gateway:**

| Resource Path  | HTTP method |
|--|-------------|
| /profiles/{ <i>eepld</i> }&variation= <i>variationId</i> | <b>GET</b>  |

**Necessary and optional additional call parameters:**

| parameter                   | valid values             | description   |
|-----------------------------|--------------------------|---|
| eepld                       | valid EEP                | EnOcean Equipment Profile, definition of EnOcean radio telegram structure           |
| <i>variation (optional)</i> | <i>valid indentifier</i> | Variation identifier. Mainly composed of ManufacturerId and ProductId that uses it. |

**response Gateway to Client:**

| parameter | datatype | value / formatting | d e s c r i p t i o n |
|-----------|----------|--------------------|-----------------------|
| header    | object   | {}                 |                       |

|         |        |    |  |
|---------|--------|----|--|
| profile | object | {} |  |
|---------|--------|----|--|

|  |     |        |          |  |
|--|-----|--------|----------|--|
|  | eep | string | xx-xx-xx | E<br>n<br>O<br>c<br>c<br>e<br>a<br>n<br>E<br>q<br>u<br>i<br>p<br>m<br>e<br>n<br>t<br>P<br>r<br>o<br>f<br>i<br>l<br>e<br>s<br>,<br>d<br>e<br>f<br>i<br>n<br>i<br>t<br>i<br>o<br>n<br>o<br>f<br>E<br>n<br>o<br>c<br>c<br>e<br>a<br>n<br>r<br>a<br>d<br>i<br>o<br>t |
|--|-----|--------|----------|--|

|  |                |                  |      |   |
|--|----------------|------------------|------|---|
|  |                |                  |      | e<br>l<br>e<br>g<br>r<br>a<br>m<br>s<br>t<br>r<br>u<br>c<br>t<br>u<br>r<br>e                                    |
|  | title          | string           |      | D<br>e<br>s<br>c<br>r<br>i<br>p<br>t<br>i<br>o<br>n<br>o<br>f<br>e<br>e<br>p<br>p<br>r<br>o<br>f<br>i<br>l<br>e |
|  | functionGroups | array of objects | [{}] |   |

|   |           |                  |           |   |
|---|-----------|------------------|-----------|---|
|   | direction | string           | from   to | d<br>i<br>r<br>e<br>c<br>t<br>i<br>o<br>n<br>o<br>f<br>t<br>r<br>a<br>n<br>s<br>p<br>o<br>r<br>t<br>(<br><i>f</i><br><i>r</i><br><i>o</i><br><i>r</i><br><i>t</i><br>o<br>d<br>e<br>v<br>i<br>c<br>e<br>) |
|   | functions | array of objects | [{}]      |   |
| <i>followed by:</i>   |           |                  |           |   |
| <i>Per key multiple values including value ranges and/or value enumerations</i> |           |                  |           |   |

|  |     |        |  |                                     |
|--|-----|--------|--|-------------------------------------|
|  | key | string |  | key value, as a function identifier |
|--|-----|--------|--|-------------------------------------|

|                     |   |                  |      |  |
|---------------------|---|------------------|------|--|
|                     | <i>description<br/>(optional)</i>       | <i>string</i>    |      | <i>d<br/>e<br/>s<br/>c<br/>r<br/>i<br/>p<br/>t<br/>i<br/>o<br/>n<br/>o<br/>f<br/>f<br/>u<br/>n<br/>c<br/>t<br/>i<br/>o<br/>n</i> |
|                     | values                                  | array of objects | [{}] |  |
| <i>value ranges</i> |   |                  |      |  |
|                     | <i>meanin<br/>g<br/>(option<br/>al)</i> | <i>string</i>    |      | <i>m<br/>e<br/>a<br/>n<br/>i<br/>n<br/>g<br/>o<br/>f<br/>v<br/>a<br/>l<br/>u<br/>e</i>   |
|                     | range                                   | object           | {}   |  |

|  |     |       |  |  |
|--|-----|-------|--|--|
|  | min | float |  | L<br>O<br>g<br>i<br>c<br>a<br>l<br>m<br>i<br>n<br>i<br>m<br>a<br>l<br>v<br>a<br>l<br>u<br>e<br>. |
|  | max | float |  | L<br>O<br>g<br>i<br>c<br>a<br>l<br>m<br>a<br>x<br>i<br>m<br>a<br>l<br>v<br>a<br>l<br>u<br>e<br>. |



|                           |                               |               |               |   |
|---------------------------|-------------------------------|---------------|---------------|---|
|                           | step                          | float         |               | V<br>a<br>l<br>u<br>e<br>s<br>t<br>e<br>p<br>p<br>i<br>n<br>g<br>.                          |
|                           | <i>unit<br/>(optional)</i>    | <i>string</i> | <b>[unit]</b> | V<br>a<br>l<br>u<br>e<br>u<br>n<br>i<br>t<br>i<br>n<br>E<br>n<br>g<br>l<br>i<br>s<br>h<br>. |
| <i>value enumerations</i> |                               |               |               |   |
|                           | value                         | float         |               |   |
|                           | <i>meaning<br/>(optional)</i> | <i>string</i> |               |   |

|  |                                   |                                 |  |  |
|--|-----------------------------------|---------------------------------|--|--|
|  | <i>defaultValue</i><br>(optional) | <i>float   string   integer</i> |  | <i>for<br/>direction<br/>, if<br/>default<br/>value<br/>is<br/>specified<br/>, the<br/>parameter</i> |
|--|-----------------------------------|---------------------------------|--|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | <i>t<br/>e<br/>r<br/>c<br/>a<br/>n<br/>b<br/>e<br/>o<br/>n<br/>i<br/>t<br/>t<br/>e<br/>d<br/>w<br/>h<br/>e<br/>n<br/>c<br/>r<br/>e<br/>a<br/>t<br/>i<br/>n<br/>g<br/>a<br/>t<br/>e<br/>l<br/>e<br/>g<br/>r<br/>a<br/>n</i> |
|--|--|--|--|--|

**example:**

The example shows the profile A5-20-04 “Radiator Valve Actuating Drive”. A device using this profile sends periodically information about temperature and others states of the device (direction “from”) and receives configuration instructions about valve settings, display settings and so on (direction “to”).

**Basic structure of gateway request & response:**

```
GET http://hostname:api_port/profiles/A5-20-04

{
  "header" : {
    "httpStatus" : 200,
    "content" : "profile",
    "gateway" : "STC-IoT v0.99.1",
    "timestamp" : "2016-05-09T16:32:04.206+0200"
  },
  "profile" : {
    "eep" : "A5-20-04",
    "title" : "HVAC Components, Heating Radiator Valve Actua",
    "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 control panel",
  "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",
      "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 ena
    }, {
      "value" : "true",
      "meaning" : "Manual room temperature selection dis
    } ],
    "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 measur
    } ],
    "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
      "range" : {
        "min" : 30,
        "max" : 1500,
        "step" : 30,
        "unit" : "s"
      }
    } ]
  }, {
    "meaning" : "Cyclic wake-up time, 3-42 hours in 3
    "range" : {
      "min" : 10800,
      "max" : 151200,

```

```
        "step" : 10800,  
        "unit" : "s"  
    }  
}, {  
    "value" : "10",  
    "meaning" : "Minimal cyclic wake-up time = 10 seconds"  
} ],  
"defaultValue" : 600  
} ]  
} ]  
}  
}
```

### 3.2.4.3.1.3 GET /profiles/{eepId}.html

#### Overview

See [HTML description](#)

### 3.2.4.3.1.4 GET /profiles/{eepId}.pdf

#### Overview

See [PDF description](#)

### 3.2.4.4 System

#### Overview

The /system resource is helpful, when you want to know details about the Gateway version or EnOcean.

#### All System Calls

- [GET /system/info](#)

#### 3.2.4.4.1 Systeminfo

#### Overview

Get version of gateway with EnOcean base information

**Client to Gateway:**



| Resource Path | HTTP method |
|---------------|-------------|
| /system/info  | GET         |

no additional call parameters are necessary

response Gateway to Client:

| parameter  | datatype              | value / formatting | description  |
|------------|-----------------------|--------------------|--|
| header     | object                | {}                 |  |
| systemInfo | object                | {}                 |  |
|            | coreVersion           | string             | contains the current firmware version, date and time of STC-IoT  |
|            | baseId                | string             | 4 byte hex value<br>current BaseID of STC-IoT  |
|            | possibleBaseIdChanges | integer            | number (10..0)<br>remaining number of BaseID changes (maximum 10 times, according to specifications of the chip) |
|            | eurid                 | string             | 4 byte hex value<br>eurid of the built-in EnOcean Chipset  |

example:

Basic structure of gateway response:

`http://hostname:api_port/system/info`

```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "systemInfo",
    "gateway" : "STC-IoT v0.99.1",
    "timestamp" : "2016-05-10T09:38:51.679+0200"
  },
  "systemInfo" : {
    "coreVersion" : "STC-IoT v0.99.0b 2016.05.06 17:47",
    "baseId" : "FFA9FF00",
    "possibleBaseIdChanges" : 10,
    "eurid" : "018553FE"
  }
}
```

### 3.2.5 HTTP Response Codes

#### Response Codes

The following HTTP response codes apply to all requests to the API services.

| HTTP Status Code      | description  |
|-----------------------|--|
| 200 (OK)              | Returned when an application request is successful.  |
| 202 (will send later) | The Telegram will be delivered after the device wakes up and sends something. (Device uses Soft Smart Ack)   |
| 400 (Bad Request)     | Returned when an error has occurred. Most likely whenever wrong or incomplete JSON has been transmitted. In addition, an error key in the header object is transferred                                   |
| 401 (Unauthorized)    | Returned when HTTP basic authentication (name/password) is invalid or privileges are not enough.   |
| 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 <a href="#">list of known application addresses</a> . |
| 404 (Not Found)       | Returned when the requested resource does not exist. For example, the device does not exist or the profile does not exist.   |
| 500 (internal error)  | Returned when a error on gateway side has occurred. In addition, an error key in the header object is transferred  |

#### Response body structure

For a successful request with HTTP Response Code 200, the response message body contains a valid representation of the requested resource in JSON format.

For an error message with HTTP Response Code 400 and 500, the error in the message body is passed with internal error number and description

```
{
  "header": {
    "httpStatus": HTTP status code,
    "code": "error code + error content",
```

```

        "content": "content of Message",
        "gateway" : "current firmware version",
        "timestamp": "transmission time in UTC format"
    },
    "object": {...}

```

### 3.2.6 Gateway Response Codes

The Gateway adds a application specific return 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

| Group         | Code        | HTTP Status | Meaning                                  | Suggested Message        |
|---------------|-------------|-------------|--|--------------------------|
| Everything OK | 1000        | 200         |  | OK                       |
|               | 1002        | 202         |  | will send telegram later |
|               | 1500 - 1999 |             | Reserved for manufacturer specific codes |                          |
| Invalid req   | 200         | 400         | invalid REST Resource,                   | invalid url %s           |

|  |       |     |  |                                   |
|--|-------|-----|--|-----------------------------------|
| uest from client (invalid URL, wrong JSON) | 00    |     | e.g. /devices/<deviceId>/invalid         |                                   |
|  | 20001 | 404 | unknown Path, e.g. /unknown/resource     | %s not found                      |
|  | 20002 | 400 | e.g. DELETE /profiles/F6-02-01           | unsupported request method %s     |
|  | 20003 | 400 | e.g. /devices/telegram?direction=invalid | parameter %s has unknown value %s |
|  | 20004 | 400 | POST without DATA                        | no data                           |
|  | 20005 | 400 | Malformed JSON                           | malformed JSON: %s                |
|  | 20009 |     | Reserved for manufacturer specific codes |                                   |
| Invalid data from client                   | 30000 | 400 | e.g. D4-01-02                            | invalid EEP %s                    |
|  | 30001 | 400 | valid but not supported EEP              | unsupported EEP %s                |
|  | 30004 | 404 | e.g. /devices/unknown                    | unknown device %s                 |

|   |    |   |   |
|---|----|---|---|
| 0 |    |   |   |
| 0 |    |   |   |
| 3 | 40 | device exists but is e.g. in learn in state           | device %s in unsupported state              |
| 1 | 0  |   |   |
| 0 |    |   |   |
| 1 |    |   |   |
| 3 | 40 | POST of different devices with e.g. same friendlyId   | duplicated ID %s                            |
| 1 | 0  |   |   |
| 0 |    |   |   |
| 2 |    |   |   |
| 3 | 40 | POST of device without deviceId and without IoT label | Missing deviceId                            |
| 1 | 0  |   |   |
| 0 |    |   |   |
| 3 |    |   |   |
| 3 | 40 |   | no functions to set specified               |
| 1 | 0  |   |   |
| 2 |    |   |   |
| 0 |    |   |   |
| 3 | 40 |   | incomplete request - fuction(s) missing: %s |
| 1 | 0  |   |   |
| 2 |    |   |   |
| 1 |    |   |   |
| 3 | 40 |   | unknown function %s                         |
| 1 | 0  |   |   |
| 2 |    |   |   |
| 2 |    |   |   |
| 3 | 40 |   | duplicated function %s                      |
| 1 | 0  |   |   |
| 2 |    |   |   |
| 3 |    |   |   |
| 3 | 40 |   | function without key                        |
| 1 | 0  |   |   |
| 2 |    |   |   |
| 4 |    |   |   |
| 3 | 40 |   | missing value of function %s                |
| 1 | 0  |   |   |
| 2 |    |   |   |
| 5 |    |   |   |
| 3 | 40 |   | invalid value %s of function %s             |
| 1 | 0  |   |   |
| 2 |    |   |   |
| 6 |    |   |   |

|                             |   |         |  |   |
|-----------------------------|---|---------|--|---|
|                             | 3<br>1<br>2<br>7                          | 40<br>0 |  | invalid value %s of function s, only scalar allowed |
|                             | 3<br>1<br>2<br>8                          | 40<br>0 |  | invalid value %s of function s, min: s, max:%s)     |
|                             |   |         |  |   |
|                             | 3<br>5<br>0<br>0<br>-<br>3<br>9<br>9<br>9 |         | Reserved for manufacturer specific codes |   |
|                             |   |         |  |   |
|                             | 4<br>0<br>0<br>0<br>-<br>7<br>9<br>9<br>9 |         | Reserved                                 |   |
|                             |   |         |  |   |
| Internal error rate gateway | 8<br>0<br>0<br>0                          | 50<br>0 |  | internal error (%s)                                 |
|                             | 8<br>0<br>0<br>1                          | 50<br>0 |  | not implemented                                     |
|                             |   |         |  |   |
|                             | 8<br>5<br>0<br>0<br>-<br>8<br>9           |         | Reserved for manufacturer specific codes |   |

|  |   |  |          |  |
|--|---|--|----------|--|
|  | 9 |  |          |  |
|  | 9 |  |          |  |
|  |   |  |          |  |
|  | 9 |  | Reserved |  |
|  | 0 |  |          |  |
|  | 0 |  |          |  |
|  | 0 |  |          |  |
|  | - |  |          |  |
|  | 9 |  |          |  |
|  | 9 |  |          |  |
|  | 9 |  |          |  |
|  | 9 |  |          |  |

### 3.3 JSON Administrator API

---

#### Administrator API

The Administrator API enables a client to use the same functions as in normal JSON API. But in addition CRUD (create, update, delete) operations on device and gateway resources are also allowed.

**Firmware 0.99.1 is recommended to work with Admin API.**

#### Security:

The Admin API can be accessed via a different login (User/Password) on HTTP basic authentication level

To use Admin-API, just login to the JSON API with the user "admin" and the password defined in the API settings in the WebUI

#### Additional exposed Functions are:

- Put the Gateway in Learn-In Mode (Listen to Learn-In Telegrams over the air)
- Create a device
- Update a device



- Delete a device
- Streaming Admin-API (get all the information from the gateway over one channel)

### Reference Client for Admin-API in Java

We have released a Java reference Client to facilitate the programming on Client side.

You can download it here: [Java Admin Client](#)

## 3.3.1 Understanding the admin API

### Overview

The Admin API fulfills the functionality that your application can administer devices in the database. Through this Admin API you are able to create, update or delete devices. New status elements have been introduced to the Admin API, that your application gets notified in real time, without polling the status.

No additional HTTP-endpoints are used in comparison to Streaming API. With other words it comes along with all endpoints you have used so far, but now empowers to use HTTP-Verbs like DELETE, POST, PUT apart from GET in case of device options. You only need to use different credentials that you have used previously. So Admin API shares the same base functionality in receiving telegrams and last states, but introduces configuration data from devices as well as some gateway attributes. Please see section Admin API credentials for more information.

The Admin API (<http://IP-Address/devices/stream>) provides a flow of information that happened in a stream of JSON data.

A message received through Admin API always follows this structure:

```
{
  "header" : {
    "content" : "<Content-Type>"
  },
  "<Content-Type>" : <Array of objects> | <single object>
```

```
}

```

These content-types you will encounter, depending on what kind of change has been made:

| Content-Type | Short description   |
|--------------|---|
| devices      | The first data you receive when connect to HTTP-Endpoint of Admin API. Any device attributes are given that have been made earlier in time.                   |
| device       | Most likely when a device configuration has been added, updated or deleted.   |
| telegram     | A device has sent a status update (direction FROM) or an IP-client has sent a update-request of a device (direction TO).                                      |
| states       | Are no longer given as content type, they are now being enveloped by a device. You will only see this content-type when you connect without admin privileges. |

### 3.3.1.1 Learning in a new device

#### Process Overview

The most common learn-in procedure should be quite easy to handle if you follow these basic steps:

- [Change operation mode into Learnmode \(POST\)](#)
- Produce EnOcean telegrams from the device (look for the device manual/help)
- Get the information of a new device over the [streaming admin-API](#) or issue a [Get New devices \(GET\)](#) call.
- Write the new devices back to the gateway with at least: (POST)
  - friendlyId
  - EEP (check on RPS telegrams)
  - operable=true

At the same time, you can monitor the different changes on the streaming admin-API

### 3.3.1.2 Streaming Admin API vs Streaming API

#### Overview

As opposed to the streaming-API which delivers states and then telegrams, the admin streaming-API delivers devices including states followed by the telegrams.

On initial connect with admin privileges you will see JSON data like this:

| Admin streaming-API  | Streaming API   |
|--|---|
| <pre>{   "header" : {     "httpStatus" : 200,     "content" : "devices",     "gateway" : "STC-IoT v0.99.1",     "timestamp" : "2016-05-10T14   },   "devices" : [ {     "deviceId" : "FFF26803",     "friendlyId" : "Rocker_Ch2",     "learnInProcedure" : "UTE",     "eeps" : [ {       "eep" : "D2-01-08",       "version" : 1.0,       "direction" : "both"     } ],     "manufacturer" : "Peha",     "firstSeen" : "2016-05-09T09     "lastSeen" : "2016-05-09T16:     "secured" : false,     "softSmartAck" : false,     "transmitModes" : [ {       "key" : "overcurrentSwitch       "transmitOnConnect" : true       "transmitOnEvent" : true,       "transmitOnDuplicate" : tr     }, {       "key" : "power",       "transmitOnConnect" : true       "transmitOnEvent" : true,       "transmitOnDuplicate" : tr     }, {       "key" : "localControl",       "transmitOnConnect" : true       "transmitOnEvent" : true,       "transmitOnDuplicate" : tr     } ], {       "key" : "energy",       "transmitOnConnect" : true       "transmitOnEvent" : true,       "transmitOnDuplicate" : tr     }, {       "key" : "errorLevel",       "transmitOnConnect" : true       "transmitOnEvent" : true,</pre> | <pre>{   "header" : {     "httpStatus" : 200,     "content" : "states",     "gateway" : "STC-IoT/EO-IP v     "timestamp" : "2016-05-10T14   },   "states" : [ {     "deviceId" : "FFF26803",     "friendlyId" : "Rocker_Ch2",     "functions" : [ {       "key" : "errorLevel",       "value" : "notSupported",       "meaning" : "Error level n       "timestamp" : "2016-05-09T       "age" : 77310501     }, {       "key" : "localControl",       "value" : "on",       "meaning" : "Local control       "timestamp" : "2016-05-09T       "age" : 77310501     }, {       "key" : "overcurrentSwitch       "value" : "false",       "meaning" : "Over current       "timestamp" : "2016-05-09T       "age" : 77310501     }, {       "key" : "switch",       "value" : "off",       "meaning" : "Output value       "timestamp" : "2016-05-09T       "age" : 77310501     } ]   } ]</pre> |

```

    "transmitOnDuplicate" : tr
  }, {
    "key" : "switch",
    "transmitOnConnect" : true
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : tr
  } ],
  "states" : [ {
    "key" : "errorLevel",
    "value" : "notSupported",
    "meaning" : "Error level n
    "timestamp" : "2016-05-09T
    "age" : 77019782
  }, {
    "key" : "localControl",
    "value" : "on",
    "meaning" : "Local control
    "timestamp" : "2016-05-09T
    "age" : 77019782
  }, {
    "key" : "overcurrentSwitch
    "value" : "false",
    "meaning" : "Over current
    "timestamp" : "2016-05-09T
    "age" : 77019782
  }, {
    "key" : "switch",
    "value" : "off",
    "meaning" : "Output value
    "timestamp" : "2016-05-09T
    "age" : 77019782
  } ],
  "operable" : true,
  "supported" : true
}
(...)

```

There will be more JSON Types as the admin-API will get more features over the time.

### 3.3.1.3 Streaming message types

#### Overview

There are three types of messages that will be sent over the streaming Admin-API

- [Devices](#) All the devices and their states at the beginning of the stream
- [Device](#) A device and its states when there has been a change in the device
- [Telegrams](#) Telegrams that are received or transmitted.

### 3.3.1.3.1 Devices

#### Devices

When connecting to the streaming Admin API, the gateway will deliver at first the devices with all their properties and states. States will come for all the devices that have the flag "TransmitOnConnect" activated.

Example for a streaming Admin API starting with the first two devices:

```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "devices",
    "gateway" : "STC-IoT v0.99.0b",
    "timestamp" : "2016-05-10T15:11:48.409+0200"
  },
  "devices" : [ {
    "deviceId" : "FFF26803",
    "friendlyId" : "Rocker_Ch2",
    "learnInProcedure" : "UTE",
    "eeps" : [ {
      "eep" : "D2-01-08",
      "version" : 1.0,
      "direction" : "both"
    } ],
    "manufacturer" : "ManufacturerName",
    "firstSeen" : "2016-05-09T09:27:05.260+0200",
    "lastSeen" : "2016-05-09T16:46:30.822+0200",
    "secured" : false,
    "softSmartAck" : false,
    "transmitModes" : [ {
      "key" : "overcurrentSwitchOff",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "power",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "localControl",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "energy",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
```

```

    "key" : "errorLevel",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  }, {
    "key" : "switch",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  } ],
  "states" : [ {
    "key" : "errorLevel",
    "value" : "notSupported",
    "meaning" : "Error level not supported",
    "timestamp" : "2016-05-09T16:46:30.822+0200",
    "age" : 80717587
  }, {
    "key" : "localControl",
    "value" : "on",
    "meaning" : "Local control enabled",
    "timestamp" : "2016-05-09T16:46:30.822+0200",
    "age" : 80717587
  }, {
    "key" : "overcurrentSwitchOff",
    "value" : "false",
    "meaning" : "Over current switch off: ready / not supported",
    "timestamp" : "2016-05-09T16:46:30.822+0200",
    "age" : 80717587
  }, {
    "key" : "switch",
    "value" : "off",
    "meaning" : "Output value OFF",
    "timestamp" : "2016-05-09T16:46:30.822+0200",
    "age" : 80717587
  } ],
  "operable" : true,
  "supported" : true
}, {
  "deviceId" : "0195DA34",
  "friendlyId" : "WaterDetector",
  "learnInProcedure" : "receivedRps",
  "eeps" : [ {
    "eep" : "F6-05-01",
    "version" : 1.0,
    "direction" : "from"
  } ],
  "firstSeen" : "2016-04-27T16:22:58.802+0200",
  "secured" : false,
  "softSmartAck" : false,
  "transmitModes" : [ {
    "key" : "liquidDetected",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  } ],
  "states" : [ ],
  "operable" : true,

```

```
    "supported" : true
  },
```

```
(...)
```

### 3.3.1.3.2 Device

#### Device

Anytime a device configuration changes, the gateway sends the full device object.

This happens whenever:

- a device has been created
- a device configuration has been updated
- a device has sent a learn-in telegram during the gateway is in receiveMode = LearnMode:
  - Learn-in not yet completed (Missing Friendly Name, etc): then the Key "operable" is false
  - Learn-In completed: then the Key "operable" is true
- a device has been deleted

Example:

```
{
  "header" : {
    "content" : "device",
    "gateway" : "STC-IoT v0.99.0b",
    "timestamp" : "2016-05-11T11:22:44.771+0200"
  },
  "device" : {
    "deviceId" : "018FBB0B",
    "learnInProcedure" : "UTE",
    "eeps" : [ {
      "eep" : "D2-01-09",
      "version" : 1.0,
      "direction" : "both"
    } ],
    "manufacturer" : "permundo GmbH",
    "firstSeen" : "2016-05-11T11:22:44.744+0200",
    "secured" : false,
    "softSmartAck" : false,
    "transmitModes" : [ {
      "key" : "overcurrentSwitchOff",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "dimValue",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
```

```

    "transmitOnDuplicate" : false
  }, {
    "key" : "power",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  }, {
    "key" : "localControl",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  }, {
    "key" : "errorLevel",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  }, {
    "key" : "energy",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  } ],
  "operable" : false,
  "supported" : true
}
}

```

### 3.3.1.3.3 Telegram

#### Telegrams

After the device information, the connection will be held open and depending on the direction parameter, telegrams will be transmitted over the stream.

```

{
  "header" : {
    "content" : "telegram",
    "gateway" : "STC-IoT v0.99.0b",
    "timestamp" : "2016-05-11T11:25:32.463+0200"
  },
  "telegram" : {
    "deviceId" : "01872C13",
    "friendlyId" : "CO2Sensor",
    "physicalDevice" : "SR04_CO2",
    "timestamp" : "2016-05-11T11:25:32.463+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "co2",
      "value" : 611.76,
      "unit" : "ppm"
    } ],
    "telegramInfo" : {
      "data" : "00004E08",
      "status" : "0",
      "dbm" : -57,
      "rorg" : "A5"
    }
  }
}

```



```

    }
  }
}

```

### 3.3.2 REST Resources

#### Overview of used Objects

To handle the response messages easier, the following REST resource are split into 3 objects.

Each of these objects can be found in the response messages at least once again. It is intended as an aid and not to be confused with the actually available REST resource paths.

|          |  |
|----------|--|
| <b>1</b> | <b>system</b>  |
|          | version of gateway with EnOcean base information   |
| <b>2</b> | <b>profile</b>   |
|          | EEP functionality / functions: Which information will send a specific profile or device and which states can be set? |
| <b>3</b> | <b>device</b>  |
|          | Information about known devices of the gateway   |

#### 3.3.2.1 Devices

##### Overview

The /devices calls are helpful, when you want to know the details about the devices, their profiles and also their states. You can also issue commands and get the devices to fulfill actions

##### GET

- [GET /devices](#)

##### PUT

- [PUT /devices](#)

##### POST

- [POST /devices](#)

## DELETE

- [DELETE /devices](#)

### 3.3.2.1.1 GET

## Overview

The GET /devices of the admin-API provide extended information about devices in different states.

## All devices

- [GET /devices](#)  
Known devices with additional information
- [GET /devices?newDevice=true](#)  
Returns new Devices that are not yet in the database
- [GET /devices/{deviceId}/profile](#)  
Returns a profile of a device with extended information

### 3.3.2.1.1.1 GET /devices

## Overview

Get detailed information of devices that has been fully learned-in.

In the Admin API, the call "GET /devices" delivers basically the same content as "GET /devices/{deviceId}" from the User API but with the following additional information:

- learnInProcedure
- version
- transmitModes
  - Key
  - transmitOnConnect
  - transmitOnEvent
  - transmitOnDuplicate
- operable
- supported
- secured

- softSmartAck

**Client to Gateway:**

| Resource Path | HTTP method |
|---------------|-------------|
| /devices      | <b>GET</b>  |

**Response Gateway to Client:**

| parameter | datatype                | value / formatting | description |
|-----------|-------------------------|--------------------|-------------|
| header    | <b>object</b>           | {}                 |             |
| devices   | <b>array of objects</b> | [{}]               |             |

|  |          |        |                             |   |
|--|----------|--------|-----------------------------|---|
|  | deviceld | string | 4<br>byte hexadecimal value | Sender ID of Enocce and device, Enocce and module's usual system name |
|--|----------|--------|-----------------------------|---|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | I<br>e<br>g<br>r<br>a<br>m<br>s<br>w<br>i<br>t<br>h<br>t<br>h<br>e<br>i<br>r<br>u<br>n<br>i<br>q<br>u<br>e<br>3<br>2<br>-<br>b<br>i<br>t<br>C<br>h<br>i<br>p<br>I<br>D<br>.I<br>n<br>t<br>h<br>e<br>o<br>t<br>h<br>e<br>r<br>c<br>a<br>s<br>e<br>, |
|--|--|--|--|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | t<br>h<br>e<br>s<br>e<br>a<br>r<br>e<br>t<br>h<br>e<br>B<br>a<br>s<br>e<br>I<br>D<br>o<br>f<br>t<br>h<br>e<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>. |
|--|--|--|--|--|

|  |            |        |  |  |
|--|------------|--------|--|--|
|  | friendlyId | string |  | Us<br>e<br>r<br>-<br>a<br>s<br>s<br>i<br>g<br>n<br>e<br>d<br>n<br>a<br>m<br>e<br>o<br>f<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e |
|--|------------|--------|--|--|

|  |                                      |               |  |  |
|--|--------------------------------------|---------------|--|--|
|  | <i>physicalDevice<br/>(optional)</i> | <i>string</i> |  | <i>Us<br/>er-<br/>as<br/>sig<br/>ne<br/>d<br/>na<br/>m<br/>e<br/>to<br/>gr<br/>ou<br/>p<br/>ac<br/>tu<br/>at<br/>or<br/>s<br/>an<br/>d<br/>se<br/>ns<br/>or<br/>s.</i> |
|--|--------------------------------------|---------------|--|--|



|  |                  |        |   |                            |
|--|------------------|--------|---|----------------------------|
|  | learnInProcedure | string | manually Configured   UT E   Smart Ack   User Interaction Required   receivedRps   received1Bs   received4Bs   incomplete | Type of Learning Procedure |
|--|------------------|--------|---|----------------------------|

|  |      |             |                     |   |  |
|--|------|-------------|---------------------|---|--|
|  |      |             |                     | eE<br>ep<br> <br>int<br>er<br>ne<br>tO<br>fT<br>hi<br>ng<br>s |  |
|  | eeps |             | array of<br>objects | [{}]  |  |
|  |      | eep         | string              | xx-<br>xx-<br>xx  | En<br>Oc<br>ea<br>n<br>Eq<br>ui<br>p<br>m<br>en<br>t<br>Pr<br>ofi<br>les<br>,<br>de<br>fin<br>iti<br>on<br>of<br>En<br>oc<br>ea<br>n<br>ra<br>di<br>o<br>tel<br>eg<br>ra<br>m<br>str<br>uc<br>tur<br>e |
|  |      | versio<br>n | float               |   | Ver<br>si  |

|  |  |                      |        |                          |                                 |
|--|--|----------------------|--------|--------------------------|---------------------------------|
|  |  |                      |        |                          | on of the API Commands          |
|  |  | direction            | string | from   to   both         | "from", "to" or "both"          |
|  |  | variation (optional) | string | manufacturer_productname | Manufacturer specific variation |

|  |                                    |               |   |  |
|--|------------------------------------|---------------|---|--|
|  | <i>manufacturer<br/>(optional)</i> | <i>string</i> |   | <i>M<br/>a<br/>n<br/>u<br/>f<br/>a<br/>c<br/>t<br/>u<br/>r<br/>e<br/>r<br/>o<br/>f<br/>t<br/>h<br/>e<br/>d<br/>e<br/>v<br/>i<br/>c<br/>e</i>   |
|  | <i>firstSeen</i>                   | <i>string</i> | <i>yy<br/>y<br/>y<br/>-<br/>m<br/>m<br/>-<br/>d<br/>d<br/>T<br/>h<br/>h<br/>:<br/>m<br/>m<br/>:<br/>s<br/>s<br/>.<br/>s<br/>s<br/>s<br/>+<br/>h<br/>h<br/>m<br/>m</i> | <i>ti<br/>m<br/>es<br/>ta<br/>m<br/>p<br/>on<br/>w<br/>hic<br/>h<br/>th<br/>e<br/>de<br/>vic<br/>e<br/>ha<br/>s<br/>be<br/>en<br/>de<br/>te<br/>ct<br/>ed<br/>for<br/>th<br/>e<br/>fir</i> |

|  |                               |         |  |  |
|--|-------------------------------|---------|--|--|
|  |                               |         |  | st<br>ti<br>me   |
|  | <i>lastSeen</i><br>(optional) | string  | yy<br>y<br>y<br>-<br>n<br>n<br>-<br>d<br>d<br>T<br>h<br>h<br>:<br>n<br>n<br>:<br>s<br>s<br>.<br>s<br>s<br>s<br>+<br>h<br>h<br>n<br>n | ti<br>m<br>e<br>s<br>t<br>a<br>m<br>p<br>o<br>n<br>w<br>h<br>i<br>c<br>h<br>t<br>h<br>e<br>d<br>e<br>v<br>i<br>c<br>e<br>h<br>a<br>s<br>b<br>e<br>e<br>n<br>d<br>e<br>t<br>e<br>c<br>t<br>e<br>d<br>f<br>o<br>r<br>t<br>h<br>e<br>l<br>a<br>s<br>t<br>t<br>i<br>m<br>e |
|  | secured                       | boolean | fal<br>s<br>e  | De<br>vic<br>e<br>u<br>s<br>i<br>n<br>g<br>S<br>e<br>c<br>u<br>r<br>i<br>t<br>y<br>o<br>v<br>e<br>r<br>a<br>i<br>r<br>(N<br>o<br>t   |

|  |               |                  |      |                             |
|--|---------------|------------------|------|-----------------------------|
|  |               |                  |      | supported at the moment)    |
|  | version       | float            |      | Version of the API Commands |
|  | softSmartAck  | boolean          |      | Device using Soft Smart-Ack |
|  | transmitModes | array of objects | [{}] |                             |

|  |        |                     |                         |                    |                             |
|--|--------|---------------------|-------------------------|--------------------|-----------------------------|
|  |        | key                 | string                  |                    | Key<br>of<br>the<br>profile |
|  |        | transmitOnConnect   | boolean                 | true<br> <br>false |                             |
|  |        | transmitOnEvent     | boolean                 | true<br> <br>false |                             |
|  |        | transmitOnDuplicate | boolean                 | true<br> <br>false |                             |
|  | states |                     | <b>array of objects</b> | [ {} ]             |                             |

|  |  |                           |                |               |   |
|--|--|---------------------------|----------------|---------------|---|
|  |  | <i>key</i>                | <i>string</i>  |               | <i>Technical key of function.</i>           |
|  |  | <i>channel (optional)</i> | <i>integer</i> |               | <i>Channel if supported by this device.</i> |
|  |  | <i>value</i>              | <i>float</i>   |               | <i>Value of the function.</i>               |
|  |  | <i>unit (optional)</i>    | <i>string</i>  | <i>[unit]</i> | <i>Unit ISO.</i>                            |



|  |  |                           |               |  |   |
|--|--|---------------------------|---------------|--|---|
|  |  | <i>meaning (optional)</i> | <i>string</i> |  | <i>Description of value in English.</i>                                       |
|  |  | <i>timestamp</i>          | <i>string</i> | <i>yy<br/>yy<br/>-<br/>m<br/>m-<br/>dd<br/>T<br/>hh<br/>:m<br/>m:<br/>ss.<br/>sss<br/>+<br/>hh<br/>m<br/>m</i> | <i>Timestamp function value was received in case of last state functions.</i> |

|  |  |            |                |  |   |
|--|--|------------|----------------|--|---|
|  |  | <i>age</i> | <i>integer</i> |  | <i>Age of function value in case of last state functions in milliseconds.</i> |
|--|--|------------|----------------|--|---|

|  |           |         |  |   |
|--|-----------|---------|--|---|
|  | operable  | boolean |  | Shows if the device is operable respectively the learn in procedure has been completed. |
|  | supported | boolean |  | Shows if the device   |

|  |  |  |  |                      |
|--|--|--|--|----------------------|
|  |  |  |  | is supported or not. |
|--|--|--|--|----------------------|

**example:**

Basic structure of gateway response:

`http://hostname:api_port/devices/`

```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "devices",
    "gateway" : "STC-IoT v0.99.1",
    "timestamp" : "2016-05-11T16:02:19.174+0200"
  },
  "devices" : [ {
    "deviceId" : "0195DA34",
    "friendlyId" : "WaterDetector",
    "learnInProcedure" : "receivedRps",
    "eeps" : [ {
      "eep" : "F6-05-01",
      "version" : 1.0,
      "direction" : "from"
    } ],
    "firstSeen" : "2016-04-27T16:22:58.802+0200",
    "secured" : false,
    "softSmartAck" : false,
    "transmitModes" : [ {
      "key" : "liquidDetected",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    } ],
    "states" : [ ],
    "operable" : true,
    "supported" : true
  }, {
    "deviceId" : "FEFCB98F",
    "friendlyId" : "ButtonMarion",
    "learnInProcedure" : "receivedRps",
    "eeps" : [ {
      "eep" : "F6-02-01",
      "version" : 1.0,
      "direction" : "from"
    } ],
    "firstSeen" : "2016-05-04T10:45:25.259+0200",
    "secured" : false,
  }
}
```

```

"softSmartAck" : false,
"transmitModes" : [ {
  "key" : "buttonA0",
  "transmitOnConnect" : false,
  "transmitOnEvent" : true,
  "transmitOnDuplicate" : true
}, {
  "key" : "buttonBI",
  "transmitOnConnect" : false,
  "transmitOnEvent" : true,
  "transmitOnDuplicate" : true
}, {
  "key" : "buttonB0",
  "transmitOnConnect" : false,
  "transmitOnEvent" : true,
  "transmitOnDuplicate" : true
}, {
  "key" : "buttonAI",
  "transmitOnConnect" : false,
  "transmitOnEvent" : true,
  "transmitOnDuplicate" : true
}, {
  "key" : "multipleButtons",
  "transmitOnConnect" : false,
  "transmitOnEvent" : true,
  "transmitOnDuplicate" : true
} ],
"states" : [ ],
"operable" : true,
"supported" : true
}
(...)

```

### 3.3.2.1.1.2 GET /devices/stream

#### Overview

Get the actual state of all devices and continue directly with updates of the devices via a telegram stream.

#### Transmitted telegrams of all devices

##### Client to Gateway:

| Resource Path  | HTTP method |
|--|-------------|
| /devices/stream?direction={ <i>direction</i> }<br>&delimited={ <i>delimited</i> }<br>&output={ <i>output</i> } | <b>GET</b>  |

optional additional call parameters:

| parameter                   | valid values  | default option   | functionality   |
|-----------------------------|---|------------------|---|
| <i>direction (optional)</i> | from   to   both  | <b>both</b>      | direction of transport ( <i>from</i> device sent / <i>to</i> device sent / <i>both</i> directions ) |
| <i>delimited (optional)</i> | length   lengthBytes   lengthCharacters   newLine   emptyLine | <b>emptyLine</b> | <a href="#">Format and delimiter options</a>  |
| <i>output (optional)</i>    | formatted   singleLine  | <b>formatted</b> | <a href="#">Format and delimiter options</a>  |

**response Gateway to Client:**

| parameter | datatype                | value / formatting | description |
|-----------|-------------------------|--------------------|-------------|
| header    | <b>object</b>           | {}                 |             |
| states    | <b>array of objects</b> | [{}]               |             |
|           | deviceId                | xxxx<br>xxxx       | Sender ID   |

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | o<br>f<br>E<br>n<br>o<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>,<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>m<br>o<br>d<br>u<br>l<br>e<br>s<br>u<br>s<br>u<br>a<br>l<br>l<br>y<br>s<br>e<br>n<br>d<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>s |
|--|--|--|--|--|

|  |  |  |  |   |
|--|--|--|--|---|
|  |  |  |  | w<br>i<br>t<br>h<br>t<br>h<br>e<br>i<br>r<br>u<br>n<br>i<br>q<br>u<br>e<br>3<br>2<br>-<br>b<br>i<br>t<br>C<br>h<br>i<br>p<br>I<br>D<br>. I<br>n<br>t<br>h<br>e<br>o<br>t<br>h<br>e<br>r<br>c<br>a<br>s<br>e<br>,<br>t<br>h<br>e<br>s<br>e<br>a<br>r |
|--|--|--|--|---|



|  |  |  |  |   |
|--|--|--|--|---|
|  |  |  |  | e<br>t<br>h<br>e<br>B<br>a<br>s<br>e<br>I<br>D<br>o<br>f<br>t<br>h<br>e<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>. |
|--|--|--|--|---|

|  |            |        |  |   |
|--|------------|--------|--|---|
|  | friendlyId | string |  | User-assigned name of EnOcean device. Must be unique. |
|--|------------|--------|--|---|

|  |                                     |                     |      |  |
|--|-------------------------------------|---------------------|------|--|
|  |                                     |                     |      | U<br>s<br>e<br>r<br>-<br>a<br>s<br>s<br>i<br>g<br>n<br>e<br>d<br>n<br>a<br>m<br>e<br>t<br>o<br>g<br>r<br>o<br>u<br>p<br>a<br>c<br>t<br>u<br>a<br>t<br>o<br>r<br>s<br>a<br>n<br>d<br>s<br>e<br>n<br>s<br>o<br>r<br>s<br>. |
|  | <i>physicalDevice</i><br>(optional) | <i>string</i>       |      |  |
|  | functions                           | array of<br>objects | [{}] |  |

|  |     |        |  |   |
|--|-----|--------|--|---|
|  | key | string |  | T<br>e<br>c<br>h<br>n<br>i<br>c<br>a<br>l<br>k<br>e<br>y<br>o<br>f<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>. |
|--|-----|--------|--|---|

|  |                               |                |  |  |
|--|-------------------------------|----------------|--|--|
|  | <i>channel<br/>(optional)</i> | <i>integer</i> |  | <i>C<br/>h<br/>a<br/>n<br/>n<br/>e<br/>l<br/>i<br/>f<br/>s<br/>u<br/>p<br/>p<br/>o<br/>r<br/>t<br/>e<br/>d<br/>b<br/>y<br/>t<br/>h<br/>i<br/>s<br/>d<br/>e<br/>v<br/>i<br/>c<br/>e<br/>.</i> |
|  | <i>value</i>                  | <i>float</i>   |  | <i>V<br/>a<br/>l<br/>u<br/>e<br/>o<br/>f<br/>t<br/>h<br/>e<br/>f<br/>u<br/>n<br/>c<br/>t<br/>i<br/>o<br/>n<br/>.</i>   |

|  |   |               |                    |  |
|--|---|---------------|--------------------|--|
|  | <i>unit<br/>(optional)</i>              | <i>string</i> | <i>[unit<br/>]</i> | <i>U<br/>n<br/>i<br/>t<br/>I<br/>S<br/>O<br/>.</i>   |
|  | <i>meanin<br/>g<br/>(option<br/>al)</i> | <i>string</i> |                    | <i>D<br/>e<br/>s<br/>c<br/>r<br/>i<br/>p<br/>t<br/>i<br/>o<br/>n<br/>o<br/>f<br/>v<br/>a<br/>l<br/>u<br/>e<br/>i<br/>n<br/>E<br/>n<br/>g<br/>l<br/>i<br/>s<br/>h<br/>.</i> |

|  |                       |               |   |  |
|--|-----------------------|---------------|---|--|
|  | <p>timesta<br/>mp</p> | <p>string</p> | <p>yyyy<br/>-<br/>mm-<br/>ddT<br/>hh:m<br/>m:ss.<br/>sss+<br/>hhm<br/>m</p> | <p>T<br/>i<br/>m<br/>e<br/>s<br/>t<br/>a<br/>m<br/>p<br/>f<br/>u<br/>n<br/>c<br/>t<br/>i<br/>o<br/>n<br/>v<br/>a<br/>l<br/>u<br/>e<br/>w<br/>a<br/>s<br/>r<br/>e<br/>c<br/>e<br/>i<br/>v<br/>e<br/>d<br/>i<br/>n<br/>c<br/>a<br/>s<br/>e<br/>o<br/>f<br/>l<br/>a<br/>s<br/>t<br/>s<br/>t<br/>a<br/>t<br/>e</p> |
|--|-----------------------|---------------|---|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>s<br>. |
|--|--|--|--|--|



|  |     |         |  |   |
|--|-----|---------|--|---|
|  | age | integer |  | A<br>g<br>e<br>o<br>f<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>v<br>a<br>l<br>u<br>e<br>i<br>n<br>c<br>a<br>s<br>e<br>o<br>f<br>l<br>a<br>s<br>t<br>s<br>t<br>a<br>t<br>e<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>s<br>i<br>m<br>i<br>l<br>l |
|--|-----|---------|--|---|

|   |  |               |           |   |
|---|--|---------------|-----------|---|
|   |  |               |           | i<br>s<br>e<br>c<br>o<br>n<br>d<br>s<br>. |
| <i>followed by live telegram stream</i>   |  |               |           |   |
| <i>header</i>                             |  | <b>object</b> | <b>{}</b> |   |
| <i>telegram</i>                           |  | <b>object</b> | <b>{}</b> |   |
| <i>Case 1/2: telegram of known device</i> |  |               |           |   |

|  |          |        |              |  |
|--|----------|--------|--------------|--|
|  | deviceId | string | xxxx<br>xxxx | S<br>e<br>n<br>d<br>e<br>r<br>I<br>D<br>o<br>f<br>E<br>n<br>o<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>,<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>m<br>o<br>d<br>u<br>l<br>e<br>s<br>u<br>s<br>u<br>a<br>l<br>l<br>y<br>s<br>e<br>n<br>d<br>t |
|--|----------|--------|--------------|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | e<br>l<br>e<br>g<br>r<br>a<br>n<br>s<br>w<br>i<br>t<br>h<br>t<br>h<br>e<br>i<br>r<br>u<br>n<br>i<br>q<br>u<br>e<br>3<br>2<br>-<br>b<br>i<br>t<br>C<br>h<br>i<br>p<br>I<br>D<br>·<br>I<br>n<br>t<br>h<br>e<br>o<br>t<br>h<br>e<br>r<br>c<br>a<br>s<br>e |
|--|--|--|--|--|

|  |  |  |  |   |
|--|--|--|--|---|
|  |  |  |  | ,<br>t<br>h<br>e<br>s<br>e<br>a<br>r<br>e<br>t<br>h<br>e<br>B<br>a<br>s<br>e<br>I<br>D<br>o<br>f<br>t<br>h<br>e<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>. |
|--|--|--|--|---|

|  |            |        |  |   |
|--|------------|--------|--|---|
|  | friendlyId | string |  | User-assigned name of EnOcean device. Must be unique. |
|--|------------|--------|--|---|

|  |                                      |               |  |  |
|--|--------------------------------------|---------------|--|--|
|  | <i>physicalDevice<br/>(optional)</i> | <i>string</i> |  | <i>U<br/>s<br/>e<br/>r<br/>-<br/>a<br/>s<br/>s<br/>i<br/>g<br/>n<br/>e<br/>d<br/>n<br/>a<br/>m<br/>e<br/>t<br/>o<br/>g<br/>r<br/>o<br/>u<br/>p<br/>a<br/>c<br/>t<br/>u<br/>a<br/>t<br/>o<br/>r<br/>s<br/>a<br/>n<br/>d<br/>s<br/>e<br/>n<br/>s<br/>o<br/>r<br/>s<br/>.</i> |
|--|--------------------------------------|---------------|--|--|

|  |           |        |  |   |
|--|-----------|--------|--|---|
|  | timestamp | string | yyyy<br>-<br>mm-<br>ddT<br>hh:m<br>m:ss.<br>sss+<br>hhm<br>m | t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>s<br>s<br>i<br>o<br>n<br>t<br>i<br>m<br>e<br>i<br>n<br>U<br>T<br>C<br>f<br>o<br>r<br>m<br>a<br>t |
|  | direction | string | from<br> <br>to  | d<br>i<br>r<br>e<br>c<br>t<br>i<br>o<br>n<br>o<br>f<br>t<br>r<br>a<br>n   |



|  |           |                  |      |   |
|--|-----------|------------------|------|---|
|  |           |                  |      | support (from device sent / to device sent) |
|  | functions | array of objects | [{}] |   |

|  |     |        |  |   |
|--|-----|--------|--|---|
|  | key | string |  | T<br>e<br>c<br>h<br>n<br>i<br>c<br>a<br>l<br>k<br>e<br>y<br>o<br>f<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>. |
|--|-----|--------|--|---|

|  |   |                       |  |   |
|--|---|-----------------------|--|---|
|  | <p><i>channel</i><br/><i>(optional)</i></p> | <p><i>integer</i></p> |  | <p><i>C</i><br/><i>h</i><br/><i>a</i><br/><i>n</i><br/><i>n</i><br/><i>e</i><br/><i>l</i><br/><i>i</i><br/><i>f</i><br/><i>s</i><br/><i>u</i><br/><i>p</i><br/><i>p</i><br/><i>o</i><br/><i>r</i><br/><i>t</i><br/><i>e</i><br/><i>d</i><br/><i>b</i><br/><i>y</i><br/><i>t</i><br/><i>h</i><br/><i>i</i><br/><i>s</i><br/><i>d</i><br/><i>e</i><br/><i>v</i><br/><i>i</i><br/><i>c</i><br/><i>e</i><br/><i>.</i></p> |
|  | <p>value</p>                                | <p>float</p>          |  | <p><i>V</i><br/><i>a</i><br/><i>l</i><br/><i>u</i><br/><i>e</i><br/><i>o</i><br/><i>f</i><br/><i>t</i><br/><i>h</i><br/><i>e</i><br/><i>f</i><br/><i>u</i><br/><i>n</i><br/><i>c</i><br/><i>t</i><br/><i>i</i><br/><i>o</i><br/><i>n</i><br/><i>.</i></p>   |

|  |                                    |               |                    |  |
|--|------------------------------------|---------------|--------------------|--|
|  | <i>unit<br/>(optional)</i>         | <i>string</i> | <i>[unit<br/>]</i> | <i>U<br/>n<br/>i<br/>t<br/>I<br/>S<br/>O<br/>.</i>   |
|  | <i>meanin<br/>g<br/>(optional)</i> | <i>string</i> |                    | <i>D<br/>e<br/>s<br/>c<br/>r<br/>i<br/>p<br/>t<br/>i<br/>o<br/>n<br/>o<br/>f<br/>v<br/>a<br/>l<br/>u<br/>e<br/>i<br/>n<br/>E<br/>n<br/>g<br/>l<br/>i<br/>s<br/>h<br/>.</i> |
|  | telegramInfo                       | object        | {}                 |  |

|  |      |        |                               |   |
|--|------|--------|-------------------------------|---|
|  | data | string | x<br>byte<br>hex<br>valu<br>e | P<br>a<br>y<br>l<br>o<br>a<br>d<br>o<br>f<br>E<br>R<br>P<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>s<br>o<br>r<br>E<br>S<br>P<br>p<br>a<br>c<br>k<br>e<br>t<br>s |
|--|------|--------|-------------------------------|---|

|  |        |         |  |  |
|--|--------|---------|--|--|
|  | status | integer |  | i<br>d<br>e<br>n<br>t<br>i<br>f<br>i<br>e<br>s<br>i<br>f<br>t<br>h<br>e<br>s<br>u<br>b<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>i<br>s<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>t<br>t<br>e<br>d<br>f<br>r<br>o<br>m<br>a<br>r<br>e<br>p<br>e<br>a<br>t |
|--|--------|---------|--|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | e<br>r<br>a<br>n<br>d<br>t<br>h<br>e<br>t<br>y<br>p<br>e<br>o<br>f<br>i<br>n<br>t<br>e<br>g<br>r<br>i<br>t<br>y<br>c<br>o<br>n<br>t<br>r<br>o<br>l<br>m<br>e<br>c<br>h<br>a<br>n<br>i<br>s<br>m<br>u<br>s<br>e<br>d<br>(<br>n<br>o<br>t<br>e<br>:<br>n |
|--|--|--|--|--|

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

o  
t  
p  
r  
e  
s  
e  
n  
t  
i  
n  
a  
R  
P  
S  
t  
e  
l  
e  
g  
r  
a  
m  
)



|  |     |         |  |  |
|--|-----|---------|--|--|
|  | dbm | integer |  | s<br>i<br>g<br>n<br>a<br>l<br>s<br>t<br>r<br>e<br>n<br>g<br>t<br>h<br>o<br>f<br>r<br>e<br>c<br>e<br>i<br>v<br>e<br>d<br>/<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>t<br>t<br>e<br>d<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m |
|--|-----|---------|--|--|

|   |          |        |                               |  |
|---|----------|--------|-------------------------------|--|
|   | rorg     | string | 1<br>byte<br>hex<br>valu<br>e | i<br>d<br>e<br>n<br>t<br>i<br>f<br>i<br>e<br>r<br>f<br>o<br>r<br>s<br>u<br>b<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>t<br>y<br>p<br>e |
| <i>Case 2/2: telegram of unknown device (only if Filtermode is Off)</i> |          |        |                               |  |
|   | deviceId | string | 4<br>byte<br>hex<br>valu<br>e | S<br>e<br>n<br>d<br>e<br>r<br>I<br>D<br>o<br>f<br>E<br>n<br>o<br>c<br>e<br>a<br>n<br>d<br>e<br>v   |

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | i<br>c<br>e<br>,<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>m<br>o<br>d<br>u<br>l<br>e<br>s<br>u<br>s<br>u<br>a<br>l<br>l<br>y<br>s<br>e<br>n<br>d<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>s<br>w<br>i<br>t<br>h<br>t<br>h<br>e<br>i<br>r<br>u<br>n<br>i |
|--|--|--|--|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | q<br>u<br>e<br>3<br>2<br>-<br>b<br>i<br>t<br>C<br>h<br>i<br>p<br>I<br>D<br>.<br>I<br>n<br>t<br>h<br>e<br>o<br>t<br>h<br>e<br>r<br>c<br>a<br>s<br>e<br>,<br>t<br>h<br>e<br>s<br>e<br>a<br>r<br>e<br>t<br>h<br>e<br>B<br>a<br>s<br>e<br>I<br>D<br>o<br>f |
|--|--|--|--|--|

|  |  |  |  |   |
|--|--|--|--|---|
|  |  |  |  | t<br>h<br>e<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>. |
|--|--|--|--|---|

|  |           |        |  |   |
|--|-----------|--------|--|---|
|  | timestamp | string | yyyy<br>-<br>mm-<br>ddT<br>hh:m<br>m:ss.<br>sss+<br>hhm<br>m | t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>s<br>s<br>i<br>o<br>n<br>t<br>i<br>m<br>e<br>i<br>n<br>U<br>T<br>C<br>f<br>o<br>r<br>m<br>a<br>t |
|--|-----------|--------|--|---|

|  |              |        |              |  |
|--|--------------|--------|--------------|--|
|  | direction    | string | from<br>  to | d<br>i<br>r<br>e<br>c<br>t<br>i<br>o<br>n<br>o<br>f<br>t<br>r<br>a<br>n<br>s<br>p<br>o<br>r<br>t<br>(<br>f<br>r<br>o<br>m<br>G<br>W<br>s<br>e<br>n<br>t<br>o<br>r<br>r<br>e<br>c<br>e<br>i<br>v<br>e<br>d<br>) |
|  | telegramInfo | object | {}           |  |

|  |      |        |                               |  |
|--|------|--------|-------------------------------|--|
|  | data | string | x<br>byte<br>hex<br>valu<br>e | P<br>a<br>y<br>l<br>o<br>a<br>d<br>o<br>f<br>E<br>R<br>P<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>s<br>o<br>r<br>E<br>S<br>P<br>p<br>a<br>c<br>k<br>e<br>t<br>s<br>. |
|--|------|--------|-------------------------------|--|



|  |        |         |  |  |
|--|--------|---------|--|--|
|  | status | integer |  | i<br>d<br>e<br>n<br>t<br>i<br>f<br>i<br>e<br>s<br>i<br>f<br>t<br>h<br>e<br>s<br>u<br>b<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>i<br>s<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>t<br>t<br>e<br>d<br>f<br>r<br>o<br>m<br>a<br>r<br>e<br>p<br>e<br>a<br>t |
|--|--------|---------|--|--|

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

e  
r  
a  
n  
d  
t  
h  
e  
t  
y  
p  
e  
o  
f  
i  
n  
t  
e  
g  
r  
i  
t  
y  
c  
o  
n  
t  
r  
o  
l  
m  
e  
c  
h  
a  
n  
i  
s  
m  
u  
s  
e  
d  
.  
(  
n  
o  
t  
e  
:

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

n  
o  
t  
p  
r  
e  
s  
e  
n  
t  
i  
n  
a  
R  
P  
S  
t  
e  
l  
e  
g  
r  
a  
n  
)

|  |     |         |  |  |
|--|-----|---------|--|--|
|  | dbm | integer |  | s<br>i<br>g<br>n<br>a<br>l<br>s<br>t<br>r<br>e<br>n<br>g<br>t<br>h<br>o<br>f<br>r<br>e<br>c<br>e<br>i<br>v<br>e<br>d<br>/<br>t<br>r<br>a<br>n<br>s<br>m<br>i<br>t<br>t<br>e<br>d<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m |
|--|-----|---------|--|--|

|  |      |        |                               |  |
|--|------|--------|-------------------------------|--|
|  | rorg | string | 1<br>byte<br>hex<br>valu<br>e | i<br>d<br>e<br>n<br>t<br>i<br>f<br>i<br>e<br>r<br>f<br>o<br>r<br>s<br>u<br>b<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m<br>t<br>y<br>p<br>e |
|--|------|--------|-------------------------------|--|

**Example:**

In that example we received the state of `multiFuncAlphaEos` and telegrams from an unknown device and a `myRocker` switch.

Basic structure of gateway response:

```
http://hostname:api_port/devices/stream

{
  "header" : {
    "httpStatus" : 200,
    "content" : "states",
    "timestamp" : "2016-05-04T10:00:02.570+0200"
  },
  "states" : [ {
    "deviceId" : "0190A078",
    "friendlyId" : "multiFunc",
    "physicalDevice" : "sense-tf-h",
    "functions" : [ {
      "key" : "humidity",
      "value" : 28.80,

```

```

        "unit" : "%",
        "timestamp" : "2016-05-04T09:59:35.758+0200",
        "age" : 26866
    }, {
        "key" : "illumination",
        "value" : 1411.76,
        "unit" : "lx",
        "timestamp" : "2016-05-04T09:59:35.758+0200",
        "age" : 26867
    }, {
        "key" : "temperature",
        "value" : 25.12,
        "unit" : "°C",
        "timestamp" : "2016-05-04T09:59:35.758+0200",
        "age" : 26867
    } ]
} ]
}

{
  "header" : {
    "content" : "telegram",
    "timestamp" : "2016-05-04T10:00:04.758+0200"
  },
  "telegram" : {
    "deviceId" : "0187D64B",
    "timestamp" : "2016-05-04T10:00:04.758+0200",
    "direction" : "from",
    "telegramInfo" : {
      "data" : "00000E5D",
      "status" : "0",
      "dbm" : -77,
      "rorg" : "A5"
    }
  }
}

{
  "header" : {
    "content" : "telegram",
    "timestamp" : "2016-05-04T10:00:58.859+0200"
  },
  "telegram" : {
    "deviceId" : "0029CFC5",
    "friendlyId" : "myRocker",
    "timestamp" : "2016-05-04T10:00:58.859+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "buttonAI",
      "value" : "pressed",
      "meaning" : "Button pressed"
    } ],
    "telegramInfo" : {
      "data" : "10",
      "status" : "30",
      "dbm" : -71,
      "rorg" : "F6"
    }
  }
}

```

```

    }
  }
}
{
  "header" : {
    "content" : "telegram",
    "timestamp" : "2016-05-04T10:00:59.027+0200"
  },
  "telegram" : {
    "deviceId" : "0029CFC5",
    "friendlyId" : "myRocker",
    "timestamp" : "2016-05-04T10:00:59.027+0200",
    "direction" : "from",
    "functions" : [ {
      "key" : "buttonAI",
      "value" : "released",
      "meaning" : "Button released"
    } ],
    "telegramInfo" : {
      "data" : "00",
      "status" : "20",
      "dbm" : -71,
      "rorg" : "F6"
    }
  }
}
}
}

```

### 3.3.2.1.1.3 GET /devices/{deviceId}

#### Overview

Get detailed information of a single device.

#### Client to Gateway:

| Resource Path       | HTTP method |
|---------------------|-------------|
| /devices/{deviceId} | GET         |

#### Response Gateway to Client:

| parameter | datatype      | value / formatting | description |
|-----------|---------------|--------------------|-------------|
| header    | <b>object</b> | {}                 |             |
| device    | <b>object</b> | {}                 |             |



|  |          |        |                           |   |
|--|----------|--------|---------------------------|---|
|  | deviceld | string | 4<br>byte<br>hex<br>value | Se<br>n<br>d<br>e<br>r<br>I<br>D<br>o<br>f<br>E<br>n<br>o<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>,<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>m<br>o<br>d<br>u<br>l<br>e<br>s<br>u<br>s<br>u<br>a<br>l<br>l<br>y<br>s<br>e<br>n<br>d<br>t<br>e |
|--|----------|--------|---------------------------|---|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | I<br>e<br>g<br>r<br>a<br>m<br>s<br>w<br>i<br>t<br>h<br>t<br>h<br>e<br>i<br>r<br>u<br>n<br>i<br>q<br>u<br>e<br>3<br>2<br>-<br>b<br>i<br>t<br>C<br>h<br>i<br>p<br>I<br>D<br>.I<br>n<br>t<br>h<br>e<br>o<br>t<br>h<br>e<br>r<br>c<br>a<br>s<br>e<br>, |
|--|--|--|--|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | t<br>h<br>e<br>s<br>e<br>a<br>r<br>e<br>t<br>h<br>e<br>B<br>a<br>s<br>e<br>I<br>D<br>o<br>f<br>t<br>h<br>e<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>. |
|--|--|--|--|--|

|  |            |        |  |  |
|--|------------|--------|--|--|
|  | friendlyId | string |  | Us<br>e<br>r<br>-<br>a<br>s<br>s<br>i<br>g<br>n<br>e<br>d<br>n<br>a<br>m<br>e<br>o<br>f<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e |
|--|------------|--------|--|--|

|  |                                      |               |  |  |
|--|--------------------------------------|---------------|--|--|
|  | <i>physicalDevice<br/>(optional)</i> | <i>string</i> |  | <i>Us<br/>er-<br/>as<br/>sig<br/>ne<br/>d<br/>na<br/>m<br/>e<br/>to<br/>gr<br/>ou<br/>p<br/>ac<br/>tu<br/>at<br/>or<br/>s<br/>an<br/>d<br/>se<br/>ns<br/>or<br/>s.</i> |
|--|--------------------------------------|---------------|--|--|

|  |                  |        |   |                                 |
|--|------------------|--------|---|---------------------------------|
|  | learnInProcedure | string | manually Configured   UT E   Smart Ack   User Interaction Required   receivedRps   received1Bs   received4Bs   incomplete | Type of Learning - In Procedure |
|--|------------------|--------|---|---------------------------------|

|  |      |             |                     |   |  |
|--|------|-------------|---------------------|---|--|
|  |      |             |                     | eE<br>ep<br> <br>int<br>er<br>ne<br>tO<br>fT<br>hi<br>ng<br>s |  |
|  | eeps |             | array of<br>objects | [{}]  |  |
|  |      | eep         | string              | xx-<br>xx-<br>xx  | En<br>Oc<br>ea<br>n<br>Eq<br>ui<br>p<br>m<br>en<br>t<br>Pr<br>ofi<br>les<br>,<br>de<br>fin<br>iti<br>on<br>of<br>En<br>oc<br>ea<br>n<br>ra<br>di<br>o<br>tel<br>eg<br>ra<br>m<br>str<br>uc<br>tur<br>e |
|  |      | versio<br>n | float               |   | Ver<br>si  |

|  |  |                      |        |                          |                                 |
|--|--|----------------------|--------|--------------------------|---------------------------------|
|  |  |                      |        |                          | on of the API Commands          |
|  |  | direction            | string | from   to   both         | "from", "to" or "both"          |
|  |  | variation (optional) | string | manufacturer_productname | Manufacturer specific variation |



|  |                                    |               |   |  |
|--|------------------------------------|---------------|---|--|
|  | <i>manufacturer<br/>(optional)</i> | <i>string</i> |   | <i>M<br/>a<br/>n<br/>u<br/>f<br/>a<br/>c<br/>t<br/>u<br/>r<br/>e<br/>r<br/>o<br/>f<br/>t<br/>h<br/>e<br/>d<br/>e<br/>v<br/>i<br/>c<br/>e</i>   |
|  | <i>firstSeen</i>                   | <i>string</i> | <i>yy<br/>y<br/>y<br/>-<br/>m<br/>m<br/>-<br/>d<br/>d<br/>T<br/>h<br/>h<br/>:<br/>m<br/>m<br/>:<br/>s<br/>s<br/>.<br/>s<br/>s<br/>s<br/>+<br/>h<br/>h<br/>m<br/>m</i> | <i>ti<br/>m<br/>es<br/>ta<br/>m<br/>p<br/>on<br/>w<br/>hic<br/>h<br/>th<br/>e<br/>de<br/>vic<br/>e<br/>ha<br/>s<br/>be<br/>en<br/>de<br/>te<br/>ct<br/>ed<br/>for<br/>th<br/>e<br/>fir</i> |

|  |                               |         |  |  |
|--|-------------------------------|---------|--|--|
|  |                               |         |  | st<br>ti<br>me   |
|  | <i>lastSeen</i><br>(optional) | string  | yy<br>y<br>y<br>-<br>n<br>n<br>-<br>d<br>d<br>T<br>h<br>h<br>:<br>n<br>n<br>:<br>s<br>s<br>.<br>s<br>s<br>s<br>+<br>h<br>h<br>n<br>n | ti<br>m<br>e<br>s<br>t<br>a<br>m<br>p<br>o<br>n<br>w<br>h<br>i<br>c<br>h<br>t<br>h<br>e<br>d<br>e<br>v<br>i<br>c<br>e<br>h<br>a<br>s<br>b<br>e<br>e<br>n<br>d<br>e<br>t<br>e<br>c<br>t<br>e<br>d<br>f<br>o<br>r<br>t<br>h<br>e<br>l<br>a<br>s<br>t<br>t<br>i<br>m<br>e |
|  | secured                       | boolean | fal<br>s<br>e  | De<br>vic<br>e<br>u<br>s<br>i<br>n<br>g<br>S<br>e<br>c<br>u<br>r<br>i<br>t<br>y<br>o<br>v<br>e<br>r<br>a<br>i<br>r<br>(N<br>o<br>t   |

|  |               |                  |      |                             |
|--|---------------|------------------|------|-----------------------------|
|  |               |                  |      | supported at the moment)    |
|  | version       | float            |      | Version of the API Commands |
|  | softSmartAck  | boolean          |      | Device using Soft Smart-Ack |
|  | transmitModes | array of objects | [{}] |                             |

|  |                              |                     |                             |                    |                             |
|--|------------------------------|---------------------|-----------------------------|--------------------|-----------------------------|
|  |                              | key                 | string                      |                    | Key<br>of<br>the<br>profile |
|  |                              | transmitOnConnect   | boolean                     | true<br> <br>false |                             |
|  |                              | transmitOnEvent     | boolean                     | true<br> <br>false |                             |
|  |                              | transmitOnDuplicate | boolean                     | true<br> <br>false |                             |
|  | <i>states<br/>(optional)</i> |                     | <b>array of<br/>objects</b> | <b>[{ }<br/>]</b>  |                             |

|  |  |                           |                |               |   |
|--|--|---------------------------|----------------|---------------|---|
|  |  | <i>key</i>                | <i>string</i>  |               | <i>Technical key of function.</i>           |
|  |  | <i>channel (optional)</i> | <i>integer</i> |               | <i>Channel if supported by this device.</i> |
|  |  | <i>value</i>              | <i>float</i>   |               | <i>Value of the function.</i>               |
|  |  | <i>unit (optional)</i>    | <i>string</i>  | <i>[unit]</i> | <i>Unit ISO.</i>                            |

|  |  |                           |               |  |   |
|--|--|---------------------------|---------------|--|---|
|  |  | <i>meaning (optional)</i> | <i>string</i> |  | <i>Description of value in English.</i>                                       |
|  |  | <i>timestamp</i>          | <i>string</i> | <i>yy<br/>yy<br/>-<br/>m<br/>m-<br/>dd<br/>T<br/>hh<br/>:m<br/>m:<br/>ss.<br/>sss<br/>+<br/>hh<br/>m<br/>m</i> | <i>Timestamp function value was received in case of last state functions.</i> |

|  |  |            |                |  |   |
|--|--|------------|----------------|--|---|
|  |  | <i>age</i> | <i>integer</i> |  | <i>Age of function value in case of last state functions in milliseconds.</i> |
|--|--|------------|----------------|--|---|

|  |           |         |  |   |
|--|-----------|---------|--|---|
|  | operable  | boolean |  | Shows if the device is operable respectively the learn in procedure has been completed. |
|  | supported | boolean |  | Shows if the device   |



|  |  |  |  |                      |
|--|--|--|--|----------------------|
|  |  |  |  | is supported or not. |
|--|--|--|--|----------------------|

**example:**

Basic structure of gateway response:

`http://hostname:api_port/devices/FFF26803`

```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "device",
    "gateway" : "STC-IoT v0.99.0b",
    "timestamp" : "2016-05-12T09:50:26.000+0200"
  },
  "device" : {
    "deviceId" : "FFF26803",
    "friendlyId" : "Rocker_Ch2",
    "learnInProcedure" : "UTE",
    "eeps" : [ {
      "eep" : "D2-01-08",
      "version" : 1.0,
      "direction" : "both"
    }, {
      "eep" : "A5-04-01",
      "variation" : "alphaEOS_SENSETF-H",
      "version" : 1.0,
      "direction" : "both"
    } ],
    "manufacturer" : "Manufacturer_Name",
    "firstSeen" : "2016-05-09T09:27:05.260+0200",
    "lastSeen" : "2016-05-09T16:46:30.822+0200",
    "secured" : false,
    "softSmartAck" : false,
    "transmitModes" : [ {
      "key" : "overcurrentSwitchOff",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "power",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "localControl",
      "transmitOnConnect" : true,
```

```

    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  }, {
    "key" : "energy",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  }, {
    "key" : "errorLevel",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  }, {
    "key" : "switch",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  } ],
  "operable" : true,
  "supported" : true
}
}

```

#### 3.3.2.1.1.4 GET /devices?newDevice=true

### Overview

Returns information of learned-in devices, that are not yet in the database. The gateway has received a learn in telegram from these devices but are held internally with a flag "operable=false" and are shown in the WebInterface under "New Device". Because they are lacking a friendlyId before they can be stored. Each time the client issues the command, it will get all the devices that are listed as New Devices.

#### Client to Gateway:

| Resource Path           | HTTP method |
|-------------------------|-------------|
| /devices?newDevice=true | <b>GET</b>  |

#### Response Gateway to Client:

| parameter | datatype                    | value /<br>formatting | description |
|-----------|-----------------------------|-----------------------|-------------|
| header    | <b>object</b>               | {}                    |             |
| devices   | <b>array of<br/>objects</b> | {}                    |             |

|  |          |        |                             |   |
|--|----------|--------|-----------------------------|---|
|  | deviceld | string | 4<br>byte hexadecimal value | Se<br>n<br>d<br>e<br>r<br>I<br>D<br>o<br>f<br>E<br>n<br>o<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>,<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>m<br>o<br>d<br>u<br>l<br>e<br>s<br>u<br>s<br>u<br>a<br>l<br>l<br>y<br>s<br>e<br>n<br>d<br>t<br>e |
|--|----------|--------|-----------------------------|---|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | I<br>e<br>g<br>r<br>a<br>m<br>s<br>w<br>i<br>t<br>h<br>t<br>h<br>e<br>i<br>r<br>u<br>n<br>i<br>q<br>u<br>e<br>3<br>2<br>-<br>b<br>i<br>t<br>C<br>h<br>i<br>p<br>I<br>D<br>.I<br>n<br>t<br>h<br>e<br>o<br>t<br>h<br>e<br>r<br>c<br>a<br>s<br>e<br>, |
|--|--|--|--|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | t<br>h<br>e<br>s<br>e<br>a<br>r<br>e<br>t<br>h<br>e<br>B<br>a<br>s<br>e<br>I<br>D<br>o<br>f<br>t<br>h<br>e<br>E<br>n<br>O<br>c<br>e<br>a<br>n<br>d<br>e<br>v<br>i<br>c<br>e<br>. |
|--|--|--|--|--|

|  |                  |        |   |                            |
|--|------------------|--------|---|----------------------------|
|  | learnInProcedure | string | manually Configured   UT E   Smart Ack   User Interaction Required   receivedRps   received1Bs   received4Bs   incomplete | Type of Learning Procedure |
|--|------------------|--------|---|----------------------------|

|  |      |             |                     |   |  |
|--|------|-------------|---------------------|---|--|
|  |      |             |                     | eE<br>ep<br> <br>int<br>er<br>ne<br>tO<br>fT<br>hi<br>ng<br>s |  |
|  | eeps |             | array of<br>objects | [{}]  |  |
|  |      | eep         | string              | xx-<br>xx-<br>xx  | En<br>Oc<br>ea<br>n<br>Eq<br>ui<br>p<br>m<br>en<br>t<br>Pr<br>ofi<br>les<br>,<br>de<br>fin<br>iti<br>on<br>of<br>En<br>oc<br>ea<br>n<br>ra<br>di<br>o<br>tel<br>eg<br>ra<br>m<br>str<br>uc<br>tur<br>e |
|  |      | versio<br>n | float               |   | Ver<br>si  |



|  |  |                      |        |                          |                                 |
|--|--|----------------------|--------|--------------------------|---------------------------------|
|  |  |                      |        |                          | on of the API Commands          |
|  |  | direction            | string | from   to   both         | "from", "to" or "both"          |
|  |  | variation (optional) | string | manufacturer_productname | Manufacturer specific variation |

|  |                                    |               |   |  |
|--|------------------------------------|---------------|---|--|
|  | <i>manufacturer<br/>(optional)</i> | <i>string</i> |   | <i>M<br/>a<br/>n<br/>u<br/>f<br/>a<br/>c<br/>t<br/>u<br/>r<br/>e<br/>r<br/>o<br/>f<br/>t<br/>h<br/>e<br/>d<br/>e<br/>v<br/>i<br/>c<br/>e</i>   |
|  | <i>firstSeen</i>                   | <i>string</i> | <i>yy<br/>y<br/>y<br/>-<br/>m<br/>m<br/>-<br/>d<br/>d<br/>T<br/>h<br/>h<br/>:<br/>m<br/>m<br/>:<br/>s<br/>s<br/>.<br/>s<br/>s<br/>s<br/>+<br/>h<br/>h<br/>m<br/>m</i> | <i>ti<br/>m<br/>es<br/>ta<br/>m<br/>p<br/>on<br/>w<br/>hic<br/>h<br/>th<br/>e<br/>de<br/>vic<br/>e<br/>ha<br/>s<br/>be<br/>en<br/>de<br/>te<br/>ct<br/>ed<br/>for<br/>th<br/>e<br/>fir</i> |

|  |                               |         |  |  |
|--|-------------------------------|---------|--|--|
|  |                               |         |  | st<br>ti<br>me   |
|  | <i>lastSeen</i><br>(optional) | string  | yy<br>y<br>y<br>-<br>n<br>n<br>-<br>d<br>d<br>T<br>h<br>h<br>:<br>n<br>n<br>:<br>s<br>s<br>.<br>s<br>s<br>s<br>+<br>h<br>h<br>n<br>n | ti<br>m<br>e<br>s<br>t<br>a<br>m<br>p<br>o<br>n<br>w<br>h<br>i<br>c<br>h<br>t<br>h<br>e<br>d<br>e<br>v<br>i<br>c<br>e<br>h<br>a<br>s<br>b<br>e<br>e<br>n<br>d<br>e<br>t<br>e<br>c<br>t<br>e<br>d<br>f<br>o<br>r<br>t<br>h<br>e<br>l<br>a<br>s<br>t<br>t<br>i<br>m<br>e |
|  | secured                       | boolean | fal<br>s<br>e  | De<br>vic<br>e<br>u<br>s<br>i<br>n<br>g<br>S<br>e<br>c<br>u<br>r<br>i<br>t<br>y<br>o<br>v<br>e<br>r<br>a<br>i<br>r<br>(N<br>o<br>t   |

|  |               |                  |      |                             |
|--|---------------|------------------|------|-----------------------------|
|  |               |                  |      | supported at the moment)    |
|  | version       | float            |      | Version of the API Commands |
|  | softSmartAck  | boolean          |      | Device using Soft Smart-Ack |
|  | transmitModes | array of objects | [{}] |                             |

|  |        |                     |                  |                    |                             |
|--|--------|---------------------|------------------|--------------------|-----------------------------|
|  |        | key                 | string           |                    | Key<br>of<br>the<br>profile |
|  |        | transmitOnConnect   | boolean          | true<br> <br>false |                             |
|  |        | transmitOnEvent     | boolean          | true<br> <br>false |                             |
|  |        | transmitOnDuplicate | boolean          | true<br> <br>false |                             |
|  | states |                     | array of objects | [ {} ]             |                             |

|  |  |                           |                |               |   |
|--|--|---------------------------|----------------|---------------|---|
|  |  | <i>key</i>                | <i>string</i>  |               | <i>Technical key of function.</i>           |
|  |  | <i>channel (optional)</i> | <i>integer</i> |               | <i>Channel if supported by this device.</i> |
|  |  | <i>value</i>              | <i>float</i>   |               | <i>Value of the function.</i>               |
|  |  | <i>unit (optional)</i>    | <i>string</i>  | <i>[unit]</i> | <i>Unit ISO.</i>                            |

|  |  |                           |               |  |   |
|--|--|---------------------------|---------------|--|---|
|  |  | <i>meaning (optional)</i> | <i>string</i> |  | <i>Description of value in English.</i>                                       |
|  |  | <i>timestamp</i>          | <i>string</i> | <i>yy<br/>yy<br/>-<br/>m<br/>m-<br/>dd<br/>T<br/>hh<br/>:m<br/>m:<br/>ss.<br/>sss<br/>+<br/>hh<br/>m<br/>m</i> | <i>Timestamp function value was received in case of last state functions.</i> |

|  |  |            |                |  |   |
|--|--|------------|----------------|--|---|
|  |  | <i>age</i> | <i>integer</i> |  | <i>Age of function value in case of last state functions in milliseconds.</i> |
|--|--|------------|----------------|--|---|



|  |           |         |       |   |
|--|-----------|---------|-------|---|
|  | operable  | boolean | false | Shows if the device is operable respectively the learn in procedure has been completed. |
|  | supported | boolean |       | Shows if the device   |

|  |  |  |  |                      |
|--|--|--|--|----------------------|
|  |  |  |  | is supported or not. |
|--|--|--|--|----------------------|

**example:**

Basic structure of gateway response:

`http://hostname:api_port/devices?newDevice=true`

```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "devices",
    "gateway" : "STC-IoT v0.99.0b",
    "timestamp" : "2016-05-12T10:01:39.147+0200"
  },
  "devices" : [ {
    "deviceId" : "018FBB0B",
    "learnInProcedure" : "UTE",
    "eeps" : [ {
      "eep" : "D2-01-09",
      "version" : 1.0,
      "direction" : "both"
    } ],
    "manufacturer" : "Manufacturer_Name",
    "firstSeen" : "2016-05-11T11:22:44.744+0200",
    "lastSeen" : "2016-05-11T17:19:17.563+0200",
    "secured" : false,
    "softSmartAck" : false,
    "transmitModes" : [ {
      "key" : "overcurrentSwitchOff",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "dimValue",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "power",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "localControl",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,

```

```

    "transmitOnDuplicate" : false
  }, {
    "key" : "errorLevel",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  }, {
    "key" : "energy",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  } ],
  "states" : [ {
    "key" : "dimValue",
    "value" : 0.00,
    "meaning" : "Output value 0% to 100%",
    "unit" : "%",
    "timestamp" : "2016-05-11T17:19:17.563+0200",
    "age" : 60141584
  }, {
    "key" : "errorLevel",
    "value" : "notSupported",
    "meaning" : "Error level not supported",
    "timestamp" : "2016-05-11T17:19:17.563+0200",
    "age" : 60141584
  }, {
    "key" : "localControl",
    "value" : "on",
    "meaning" : "Local control enabled",
    "timestamp" : "2016-05-11T17:19:17.563+0200",
    "age" : 60141584
  }, {
    "key" : "overcurrentSwitchOff",
    "value" : "false",
    "meaning" : "Over current switch off: ready / not supp",
    "timestamp" : "2016-05-11T17:19:17.563+0200",
    "age" : 60141584
  } ],
  "operable" : false,
  "supported" : true
}, {
  "deviceId" : "0191047D",
  "learnInProcedure" : "receivedRps",
  "eeps" : [ {
    "eep" : "F6-02-01",
    "version" : 1.0,
    "direction" : "from"
  } ],
  "firstSeen" : "2016-05-11T11:14:54.773+0200",
  "lastSeen" : "2016-05-11T16:55:34.185+0200",
  "secured" : false,
  "softSmartAck" : false,
  "transmitModes" : [ {
    "key" : "buttonA0",
    "transmitOnConnect" : false,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  } ]
} ]

```

```

    }, {
      "key" : "buttonB0",
      "transmitOnConnect" : false,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "buttonBI",
      "transmitOnConnect" : false,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "buttonAI",
      "transmitOnConnect" : false,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "multipleButtons",
      "transmitOnConnect" : false,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    } ],
    "states" : [ ],
    "operable" : false,
    "supported" : true
  } ]
}

```

### 3.3.2.1.1.5 GET /device/{deviceId}/profile

#### Profile detail

Get detailed profile informations of a device

In the Admin API, the call "GET /devices/{deviceId}/profile" delivers basically the same content as "GET /devices/{deviceId}/profile" from the User API but with the following additional information:

- functions
  - transmitOnConnect
  - transmitOnEvent
  - transmitOnDuplicate
  - testExists

#### Client to Gateway:

| Resource Path               | HTTP method |
|-----------------------------|-------------|
| /devices/{deviceId}/profile | GET         |

#### response Gateway to Client:

| parameter |                |                         | datatype         | value / formatting | description                          |
|-----------|----------------|-------------------------|------------------|--------------------|--------------------------------------|
| header    |                |                         | object           | {}                 |                                      |
| profile   |                |                         | object           | {}                 |                                      |
|           | functionGroups |                         | array of objects | [{}]               |                                      |
|           |                | <i>title (optional)</i> | <i>string</i>    |                    | <i>description of function group</i> |
|           |                | direction               | string           | from   to   both   |                                      |
|           |                | functions               | array of objects | [{}]               |                                      |

|  |  |  |                     |         |  |  |
|--|--|--|---------------------|---------|--|--|
|  |  |  | key                 | string  |  |  |
|  |  |  | transmitOnConnect   | boolean |  |  |
|  |  |  | transmitOnEvent     | boolean |  |  |
|  |  |  | transmitOnDuplicate | boolean |  |  |

|  |  |  |            |         |  |  |
|--|--|--|------------|---------|--|--|
|  |  |  | testExists | boolean |  | Specifies whether the rate test for the device exists in the data base |
|--|--|--|------------|---------|--|--|

|  |  |  |              |                         |                  |  |  |
|--|--|--|--------------|-------------------------|------------------|--|--|
|  |  |  |              |                         |                  |  | s<br>e<br>o<br>r<br>n<br>o<br>t<br>.<br>T<br>h<br>e<br>t<br>e<br>s<br>t<br>c<br>a<br>n<br>b<br>e<br>u<br>s<br>e<br>d<br>t<br>o<br>i<br>d<br>e<br>n<br>t<br>i<br>f<br>y<br>a<br>d<br>e<br>v<br>i<br>c<br>e<br>. |
|  |  |  | defaultValue |                         | string   float   |  |  |
|  |  |  | values       |                         | array of objects |  |  |
|  |  |  |              | <i>range (optional)</i> | <i>object</i>    |  | {<br>}   |



|  |  |  |  |  |     |       |  |  |
|--|--|--|--|--|-----|-------|--|--|
|  |  |  |  |  | min | float |  | L<br>o<br>g<br>i<br>c<br>a<br>l<br>m<br>i<br>n<br>i<br>m<br>a<br>l<br>v<br>a<br>l<br>u<br>e<br>. |
|  |  |  |  |  | max | float |  | L<br>o<br>g<br>i<br>c<br>a<br>l<br>m<br>a<br>x<br>i<br>m<br>a<br>l<br>v<br>a<br>l<br>u<br>e<br>. |

|  |  |  |  |                       |                    |              |        |   |
|--|--|--|--|-----------------------|--------------------|--------------|--------|---|
|  |  |  |  |                       | step               | float        |        | V<br>a<br>l<br>u<br>e<br>s<br>t<br>e<br>p<br>p<br>i<br>n<br>g<br>.                          |
|  |  |  |  |                       | unit<br>(optional) | string       | [unit] | V<br>a<br>l<br>u<br>e<br>u<br>n<br>i<br>t<br>i<br>n<br>E<br>n<br>g<br>l<br>i<br>s<br>h<br>. |
|  |  |  |  | value<br>(optional)   |                    | float   stri |        |   |
|  |  |  |  | meaning<br>(optional) |                    | string       |        |   |

**example:**

Basic structure of gateway response:

```

http://hostname:api_port/devices/FFF26803/profile

{
  "header" : {
    "httpStatus" : 200,
    "content" : "profile",
    "gateway" : "STC-IoT v0.99.0b",
    "timestamp" : "2016-05-12T10:42:36.937+0200"
  },
  "profile" : {
    "functionGroups" : [ {
      "title" : "Actuator Set Output",
      "direction" : "to",
      "functions" : [ {
        "key" : "switch",
        "values" : [ {
          "value" : "off",
          "meaning" : "Output value OFF"
        }, {
          "value" : "on",
          "meaning" : "Output value ON"
        } ] ,
        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false,
        "testExists" : true
      } ]
    }, {
      "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"
        } ] ,
        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false,
        "defaultValue" : "previousState"
      }, {
        "key" : "localControl",

```

```

    "values" : [ {
      "value" : "off",
      "meaning" : "Disable local control"
    }, {
      "value" : "on",
      "meaning" : "Enable local control"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "defaultValue" : "on"
  }, {
    "key" : "overcurrentSwitchOffMode",
    "values" : [ {
      "value" : "restart",
      "meaning" : "automatic restart"
    }, {
      "value" : "staticOff",
      "meaning" : "static off"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "defaultValue" : "staticOff"
  }, {
    "key" : "overcurrentSwitchOffReset",
    "values" : [ {
      "value" : "false",
      "meaning" : "Reset over current shut down: not act
    }, {
      "value" : "true",
      "meaning" : "Reset over current shut down: trigger
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "defaultValue" : "false"
  }, {
    "key" : "taughtInDevices",
    "values" : [ {
      "value" : "off",
      "meaning" : "Disable taught-in devices (with diffe
    }, {
      "value" : "on",
      "meaning" : "Enable taught-in devices (with differ
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "defaultValue" : "on"
  }, {
    "key" : "userInterfaceIndication",
    "values" : [ {
      "value" : "day",
      "meaning" : "User interface indication: day operat
    }, {
      "value" : "night",

```

```

        "meaning" : "User interface indication: night oper
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "defaultValue" : "day"
} ]
}, {
"title" : "Actuator Status Response",
"direction" : "from",
"functions" : [ {
    "key" : "errorLevel",
    "values" : [ {
        "value" : "failure",
        "meaning" : "Error level 2: hardware failure"
    }, {
        "value" : "noError",
        "meaning" : "Error level 0: hardware OK"
    }, {
        "value" : "notSupported",
        "meaning" : "Error level not supported"
    }, {
        "value" : "warning",
        "meaning" : "Error level 1: hardware warning"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
}, {
    "key" : "localControl",
    "values" : [ {
        "value" : "off",
        "meaning" : "Local control disabled / not supporte
    }, {
        "value" : "on",
        "meaning" : "Local control enabled"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
}, {
    "key" : "overcurrentSwitchOff",
    "values" : [ {
        "value" : "false",
        "meaning" : "Over current switch off: ready / not
    }, {
        "value" : "true",
        "meaning" : "Over current switch off: executed"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
}, {
    "key" : "switch",
    "values" : [ {
        "value" : "off",
        "meaning" : "Output value OFF"
    } ]
} ]

```

```
    }, {
      "value" : "on",
      "meaning" : "Output value ON"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  } ]
}, {
  "title" : "Actuator Set Measurement",
  "direction" : "to",
  "functions" : [ {
    "key" : "energyDelta",
    "description" : "Delta of energy to be reported",
    "values" : [ {
      "range" : {
        "min" : 0,
        "max" : 4095000,
        "step" : 0.000278,
        "unit" : "Wh"
      }
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  }, {
    "key" : "maxTimeBetweenReports",
    "description" : "Measurement Response messages",
    "values" : [ {
      "range" : {
        "min" : 10,
        "max" : 2550,
        "step" : 10,
        "unit" : "s"
      }
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "defaultValue" : 60
  }, {
    "key" : "minTimeBetweenReports",
    "description" : "Measurement Response messages",
    "values" : [ {
      "range" : {
        "min" : 0,
        "max" : 255,
        "step" : 1,
        "unit" : "s"
      }
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "defaultValue" : 10
  }, {
    "key" : "reportMeasurement",
```

```

    "values" : [ {
      "value" : "queryAndAuto",
      "meaning" : "Report measurement: query / auto report"
    }, {
      "value" : "queryOnly",
      "meaning" : "Report measurement: query only"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "defaultValue" : "queryAndAuto"
  }, {
    "key" : "resetMeasurement",
    "values" : [ {
      "value" : "false",
      "meaning" : "Reset measurement: not active"
    }, {
      "value" : "true",
      "meaning" : "Reset measurement: trigger signal"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "defaultValue" : "false"
  } ]
}, {
  "title" : "Actuator Set Measurement",
  "direction" : "to",
  "functions" : [ {
    "key" : "maxTimeBetweenReports",
    "description" : "Measurement Response messages",
    "values" : [ {
      "range" : {
        "min" : 10,
        "max" : 2550,
        "step" : 10,
        "unit" : "s"
      }
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "defaultValue" : 60
  }, {
    "key" : "minTimeBetweenReports",
    "description" : "Measurement Response messages",
    "values" : [ {
      "range" : {
        "min" : 0,
        "max" : 255,
        "step" : 1,
        "unit" : "s"
      }
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,

```

```

    "defaultValue" : 10
  }, {
    "key" : "powerDelta",
    "description" : "Delta of power to be reported",
    "values" : [ {
      "range" : {
        "min" : 0,
        "max" : 4095000,
        "step" : 1,
        "unit" : "W"
      }
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  }, {
    "key" : "reportMeasurement",
    "values" : [ {
      "value" : "queryAndAuto",
      "meaning" : "Report measurement: query / auto report"
    }, {
      "value" : "queryOnly",
      "meaning" : "Report measurement: query only"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "defaultValue" : "queryAndAuto"
  }, {
    "key" : "resetMeasurement",
    "values" : [ {
      "value" : "false",
      "meaning" : "Reset measurement: not active"
    }, {
      "value" : "true",
      "meaning" : "Reset measurement: trigger signal"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "defaultValue" : "false"
  } ]
}, {
  "title" : "Actuator Query",
  "direction" : "to",
  "functions" : [ {
    "key" : "query",
    "values" : [ {
      "value" : "energy",
      "meaning" : "Query energy"
    }, {
      "value" : "power",
      "meaning" : "Query power"
    }, {
      "value" : "status",
      "meaning" : "Query status"
    } ],
  } ],

```



```

        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false
    } ]
}, {
    "title" : "Actuator Measurement Response",
    "direction" : "from",
    "functions" : [ {
        "key" : "energy",
        "description" : "Cumulative electricity value from m
        "values" : [ {
            "range" : {
                "min" : 0,
                "max" : 4294967295000,
                "step" : 0.000278,
                "unit" : "Wh"
            }
        } ],
        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false
    }, {
        "key" : "power",
        "description" : "Current power value from meter",
        "values" : [ {
            "range" : {
                "min" : 0,
                "max" : 4294967295000,
                "step" : 1,
                "unit" : "W"
            }
        } ],
        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false
    } ]
}, {
    "direction" : "from",
    "functions" : [ {
        "key" : "humidity",
        "description" : "Rel. Humidity (linear)",
        "values" : [ {
            "range" : {
                "min" : 0,
                "max" : 100,
                "step" : 0.4,
                "unit" : "%"
            }
        } ],
        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false
    }, {
        "key" : "illumination",
        "description" : "Illumination (linear)",
        "values" : [ {
            "range" : {

```



If a device has the flag "testExists" in its profile or device description, the gateway can send out a test procedure to the device to make it do some device specific actions like switch on/of or dim 0%/25%/50%/75%/100%.

The test ends by itself or can be stopped with the vale "testState = false"

**Client to Gateway:**

| Resource Path                   | HTTP method |
|---------------------------------|-------------|
| <i>/devices/{deviceId}/test</i> | <b>PUT</b>  |

**Necessary call parameters:**

| parameter  | datatype  | value / formatting | description  |
|------------|-----------|--------------------|--|
| header     | object    | {}                 |  |
| testDevice | object    | {}                 |  |
|            | deviceId  | string             | 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. |
|            | testState | string             | true   false   |

|  |     |        |          |  |
|--|-----|--------|----------|--|
|  | eep | string | xx-xx-xx | EnOcean Equipment Profiles, definition of EnOcean radio telegram structure |
|--|-----|--------|----------|--|

**example:**

The example shows the request for the gateway to test a device

Basic structure of gateway call:

```
PUT http://hostname:api_port/devices/FFF26803/test
{
  "testDevice" : {
    "deviceId" : "FFF26803",
    "testState" : "true",
    "eep" : "D2-01-08"
  }
}
```

**3.3.2.1.2.2 PUT /devices/{deviceId}/learnIn****LearnIn**

The call sends out a learnIn telegram to bidirectional devices, which are not using UTE, SmartAck or 4BS-TeachIn Variation 3 as Learn-in procedure.

LearnIn Telegrams to be sent out this way are:

- receivedRps
- received1Bs
- received4Bs

The function can only be called for devices that have an already set EEP-Value, from which the gateway knows the learnIn telegram to be sent out.

These devices have to be manually set to listen to learnIn telegrams by the user, before sending out this telegram.

**Client to Gateway:**

| Resource Path                            | HTTP method |
|--|-------------|
| <code>/devices/{deviceId}/learnIn</code> | <b>PUT</b>  |

**Necessary call parameters:**

| parameter | datatype               | value / formatting | description   |
|-----------|------------------------|--------------------|---|
| header    | object                 | {}                 | standard header object with content = learnIn   |
| learnIn   | object                 | {}                 | learnIn object  |
|           | deviceId               | string             | deviceId  |
|           | <i>mode (optional)</i> | <i>string</i>      | <i>firstButton   secondButton</i><br><i>Only used when sending a RPS telegram.</i>  |
|           | eep                    | string             | xx-xx-xx based on receivedRps   received1Bs   received4Bs<br>EnOcean Equipment Profiles, definition of EnOcean radio telegram structure |

**example:**

The example shows the request for the gateway to send out a learnIn telegram to a already known device.

Basic structure of gateway call:

```

http://hostname:api_port/{deviceId}/learnIn

{
  "learnIn" : {
    "deviceId" : "00290FE2",
    "mode" : "secondButton",
    "eep" : "F6-02-01"
  }
}

```

### 3.3.2.1.3 POST

#### Overview

The POST /devices call of the admin-API creates or updates devices in the database.

#### Single device

- [POST /devices/{deviceId}](#) Creates or updates a device in the database of the gateway

### 3.3.2.1.3.1 POST /devices/{deviceId}

#### Overview

If you want to learn-in a device completely via API, be aware of:

- It probably makes only sense for unidirectional sensors
  - If you want to learn-in bidirectional actuators, you will have to coordinate the Learn-in telegram that needs to be transmitted to the device by hand.

#### Client to Gateway:

| Resource Path       | HTTP method |
|---------------------|-------------|
| /devices/{deviceId} | <b>POST</b> |

#### POST body:

| parameter | datat<br>ype       | valu<br>e /<br>form<br>attin<br>g | Re<br>qui<br>red | Re<br>ad/<br>Wri<br>te | des<br>cri<br>pti<br>on |
|-----------|--------------------|-----------------------------------|------------------|------------------------|-------------------------|
| device    | <b>objec<br/>t</b> | { }                               |                  |                        |                         |

|  |          |        |                  |     |       |   |
|--|----------|--------|------------------|-----|-------|---|
|  | deviceId | string | 4 byte hex value | Yes | Write | Sender ID of Oceanan device, Ocean modules usually send telegrams with their unique 32-bit Chip ID. In the other case, these are the Base |
|--|----------|--------|------------------|-----|-------|---|



|  |                                  |               |                                     |     |              |   |
|--|----------------------------------|---------------|-------------------------------------|-----|--------------|---|
|  |                                  |               |                                     |     |              | ID of the En Ocean device.                                |
|  | friendlyId                       | string        | alphanumeric, hyphen and underscore | Yes | Write        | use re-assigned name of En Ocean device                   |
|  | <i>physicalDevice (optional)</i> | <i>string</i> |                                     | No  | <i>Write</i> | <i>User-assigned name to group actuators and sensors.</i> |

|  |                  |                  |  |  |       |   |
|--|------------------|------------------|--|--|-------|---|
|  | learnInProcedure | string           | manually Configured   UTE   SmartAck   UserInteraction Required   receivedRps   received1Bs   received4Bs   incompleteEep   internetOfThings |  | Write | Device should be set to manually Configured |
|  | eeps             | array of objects | [{}]   |  |       |   |

|  |  |                   |        |                        |     |       |  |
|--|--|-------------------|--------|------------------------|-----|-------|--|
|  |  | ee<br>p           | string | xx-<br>xx-xx           | Yes | Write | En<br>Oc<br>ea<br>n<br>Eq<br>ui<br>p<br>me<br>nt<br>Pro<br>file<br>s,<br>def<br>ini<br>tion<br>of<br>En<br>oc<br>ea<br>n<br>ra<br>dio<br>tel<br>egr<br>am<br>str<br>uct<br>ure |
|  |  | ver<br>sio<br>n   | float  |                        | No  | Write | Ver<br>sio<br>n<br>of<br>the<br>API<br>Co<br>m<br>ma<br>nds  |
|  |  | dir<br>ect<br>ion | string | from  <br>to  <br>both | Yes | Write | "fr<br>om<br>",<br>"to<br>"<br>or<br>"bo<br>th"  |

|  |  |                      |        |  |    |           |  |
|--|--|----------------------|--------|--|----|-----------|--|
|  |  | variation (optional) | string | manufacturer_productname                 | No | Write     | Manufacturer specific variation                                    |
|  |  |                      |        |  |    |           |  |
|  |  | firstSeen            | string | yyyy-mm-ddT<br>hh:mm:ss.<br>sss+<br>hhmm | No | Read only | timestamp on which the device has been detected for the first time |
|  |  | lastSeen             | string | yyyy-mm-ddT<br>hh:mm:ss.<br>sss+<br>hhmm | No | Read only | timestamp on which the device has been detected for                |

|  |                                |                |                     |           |                  |   |
|--|--------------------------------|----------------|---------------------|-----------|------------------|---|
|  |                                |                |                     |           |                  | <i>the last time</i>  |
|  | <i>secured</i>                 | <i>boolean</i> | <i>false</i>        | <i>No</i> | <i>Read only</i> | <i>Device using Security over air (Not supported at the moment)</i> |
|  | <i>manufacturer (optional)</i> | <i>string</i>  |                     | <i>No</i> | <i>Write</i>     | <i>Manufacturer of the device.</i>                                  |
|  | <i>softSmartAck (optional)</i> | <i>boolean</i> | <i>true   false</i> | <i>No</i> | <i>Write</i>     | <i>Device using Soft Smart - Ack</i>                                |

|  |               |                     |                  |      |     |           |                            |
|--|---------------|---------------------|------------------|------|-----|-----------|----------------------------|
|  | transmitModes |                     | array of objects | [{}] |     |           |                            |
|  |               | key                 | string           |      | Yes | Write     | Key of the profile         |
|  |               | transmitOnConnect   | boolean          |      | Yes | Write     | True/False                 |
|  |               | transmitOnEvent     | boolean          |      | Yes | Write     | True/False                 |
|  |               | transmitOnDuplicate | boolean          |      | Yes | Write     | True/False                 |
|  | states        |                     | array of objects | [{}] |     | Read only |                            |
|  |               | key                 | string           |      |     | Read only | Technical key of function. |

|  |  |                           |                |               |  |                  |   |
|--|--|---------------------------|----------------|---------------|--|------------------|---|
|  |  | <i>channel (optional)</i> | <i>integer</i> |               |  | <i>Read only</i> | <i>Channel if supported by this device.</i> |
|  |  | <i>value</i>              | <i>float</i>   |               |  | <i>Read only</i> | <i>Value of the function.</i>               |
|  |  | <i>unit (optional)</i>    | <i>string</i>  | <i>[unit]</i> |  | <i>Read only</i> | <i>Unit ISO.</i>                            |
|  |  | <i>meaning (optional)</i> | <i>string</i>  |               |  | <i>Read only</i> | <i>Description of value in English.</i>     |

|  |  |                  |                |  |  |                  |  |
|--|--|------------------|----------------|--|--|------------------|--|
|  |  | <i>timestamp</i> | <i>string</i>  | <i>yyyy-mm-ddT<br/>hh:mm:ss.<br/>sss+<br/>hhmm</i> |  | <i>Read only</i> | <i>Timestamp function value was received in case of last state functions.</i>  |
|  |  | <i>age</i>       | <i>integer</i> |  |  | <i>Read only</i> | <i>Age of function value in case of last state functions in milli seconds.</i> |



|  |                  |                |                     |     |                  |   |
|--|------------------|----------------|---------------------|-----|------------------|---|
|  | operable         | boolean        | true   false        | Yes | Write            | Shows if the device is operable respectively the learn in procedure has been completed. |
|  | <i>supported</i> | <i>boolean</i> | <i>true   false</i> | No  | <i>Read Only</i> | <i>Shows if the device is supported as of the profile</i>                               |

example 1:

This first example shows a post request to store the device in the database with all required properties.

#### POST a new device to the database

##### POST /devices/FFF26803

```
{
  "device" : {
    "deviceId" : "FFF26803",
    "friendlyId" : "Peha_Ch2",
    "learnInProcedure" : "manuallyConfigured",
    "eeps" : [ {
      "eep" : "D2-01-08",
      "version" : 1.0,
      "direction" : "both"
    } ],
    "manufacturer" : "Peha",
    "firstSeen" : "2016-05-09T09:27:05.260+0200",
    "lastSeen" : "2016-05-09T16:46:30.822+0200",
    "secured" : false,
    "softSmartAck" : false,
    "transmitModes" : [ {
      "key" : "overcurrentSwitchOff",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "power",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "localControl",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "energy",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "errorLevel",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "switch",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    } ],
    "operable" : true,
    "supported" : true
  }
}
```

```
}

```

**example 2:**

This example shows also how to set up a Post request to learn in a device. But in this case it is shown that we can use many properties from the JSON we received earlier during streaming. That requires the gateway to be in learn in mode while the device sends a learn in telegram. Equivalent information can be retrieved on this resource:

`/devices/newDevice=true.`

**GET /devices/stream**

```
(...)
{
  "header" : {
    "content" : "device",
    "gateway" : "STC-IoT v0.99.1",
    "timestamp" : "2016-05-11T11:22:44.771+0200"
  },
  "device" : {
    "deviceId" : "018FBB0B",
    "learnInProcedure" : "UTE",
    "eeps" : [ {
      "eep" : "D2-01-09",
      "version" : 1.0,
      "direction" : "both"
    } ],
    "manufacturer" : "Manufacturer_Name",
    "firstSeen" : "2016-05-11T11:22:44.744+0200",
    "secured" : false,
    "softSmartAck" : false,
    "transmitModes" : [ {
      "key" : "overcurrentSwitchOff",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "dimValue",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "power",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "localControl",

```

```

        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false
    }, {
        "key" : "errorLevel",
        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false
    }, {
        "key" : "energy",
        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false
    } ],
    "operable" : false,
    "supported" : true
}
}
(...)

```

**Attach a friendlyId and set operable to true.**

**POST /devices/019724DB**

```

{
  "header" : {
    "content" : "devices"
  },
  "devices" : [ {
    "deviceId" : "019724DB",
    "friendlyId" : "OFIPostTest",
    "learnInProcedure" : "UTE",
    "eep" : "D2-01-09",
    "manufacturer" : "Manufacturer_Name",
    "firstSeen" : "2015-12-15T14:54:30.058+0100",
    "secured" : false,
    "softSmartAck" : false,
    "transmitModes" : [ {
      "key" : "overcurrentSwitchOff",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "dimValue",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "power",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {

```

```
    "key" : "localControl",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  }, {
    "key" : "errorLevel",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  }, {
    "key" : "energy",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  } ],
  "states" : [ ],
  "operable" : true,
  "supported" : true
} ]
}
```

### 3.3.2.1.3.2 Internet of Things Device

#### Overview

The upcoming InternetOfThings requires EnOcean Hardware devices to be mass market compatible. The overall agreement is that End User should not be bothered with complicated Learn-In procedures.

These requirements for IoT compatible EnOcean Hardware can be met by combining already existing EnOcean Specifications into an IoT Label, which can then be given to devices that meet these requirements.

InternetOfThings hardware must fulfill the following requirements:

- Product ID (QR-Code)
- Remote Management
- Remote Commissioning
- Device Definition File
- Security over Radio

When a device complies to all of these specifications it gets an IoT-Seal as proof. These devices are in preparation but do not yet exist.

#### IoT Learn-In Example

The Learn-In procedure for InternetOfThings hardware will look like this:

- User scans a new hardware with an appropriate App
- The App transmits the QR-Code to the gateway

- The gateway gets the Device Definition File (ReCom-Information about the device) from the central DDF repository (hosted by EnOcean Alliance)
- The gateway creates a new device in state "operable=false" and sends the Information over the streaming-admin API
- As soon as the gateway gets its first telegram from the IoT device, it starts the ReMan/ReCom procedure and pairs the device with the gateway
- The gateway changes the device state to "operable=true" and sends the Information over the streaming-admin API
- The device can be used.

## Topics

- [IoT Label](#)
- [Create a IoT device](#)
- [QR Scan for existing Sensors](#)

## QR-Code

The functionality is achieved through a learn-in with the help of a QR-code and the support of Remote Management and Remote Commissioning.

The QR Code has the following format:

- EURID (EnOcean Unique Radio Identifier )
- Product ID - which is composed of
  - Manufacturer ID
  - ProductIdentifier (number)
- Version Numbering
- Security Code (optional)

Details can be looked up in the Specification "Product ID and Standardized Labeling Specification" dated 2014/05/16 from the EnOcean Alliance

## Example QR Text

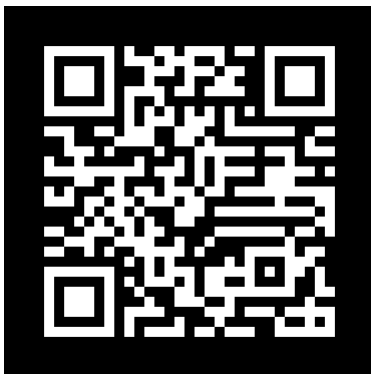
QR-Code for the IoT Learn-In:  
30S000001910EB8+1P000C00000002+10Z00+11ZFFFFFFFE

## Details

| Identifier | Identifier length | Length of data excluding identifier | Value             |
|------------|-------------------|-------------------------------------|-------------------|
| 30S        | 3 characters      | 12 characters                       | EURID             |
| 1P         | 2 characters      | 12 characters                       | Product ID        |
| 10Z        | 3 characters      | 2 characters                        | Version numbering |
| 11Z        | 3 characters      | 8 characters                        | Security Code     |

### Example QR Picture

QR-Code looks like this:



QR-Codes can be produced for example here: <http://www.qrcode-generator.de/>

### InternetOfThings Device

Any App using "InternetOfThings" learn in needs to transmit the QR-Code in a property called `iotLabel` to the gateway.

**Client to Gateway:**

| Resource Path         | HTTP method |
|-----------------------|-------------|
| <code>/devices</code> | <b>POST</b> |

**Necessary call parameters:**

| parameter |                  | datatype | value / formatting   | description   |
|-----------|------------------|----------|--|---|
| header    |                  | object   | {}   |   |
| device    |                  | object   | {}   |   |
|           | learnInProcedure | string   | internetOfThings   |   |
|           | firstSeen        | string   | yyyy-mm-ddT<br>hh:mm:ss<br>.sss+<br>hhmm                             | date/time   |
|           | iotLabel         | string   | 30S000001<br>90F448+1<br>P000C000<br>00002+10Z<br>00+11ZFFF<br>FFFFE | Valid IoT Label. Refer to <a href="#">IoT Label</a> |

**example:**

HTTP POST command including QR-Code

- a. HTTP Response of the POST command
- b. Message from the gateway over the streaming admin-API after the scan (Device has not been operating so far)
- c. Message after the device has been discovered by any first telegram

**POST /devices**

Body:

```

{
  "header" : {
    "content" : "device"
  },
  "device" : {
    "learnInProcedure" : "internetOfThings",
    "firstSeen" : "2017-04-04T20:53:03.795+0100",
    "iotLabel" : "30S000001910188+1P0040000000002+10Z00+11Z27D"
  }
}

```

**a. Answer of the gateway to an IoT POST command**



```
{
  "header": {
    "httpStatus": 200,
    "content": "device",
    "gateway": "STC-IoT v0.99.39",
    "timestamp": "2017-06-19T13:35:48.934+0200"
  },
  "device": {
    "deviceId": "01910188",
    "friendlyId": "device",
    "learnInProcedure": "internetOfThings",
    "eeps": [
      {
        "eep": "D2-01-11",
        "version": 0.9,
        "direction": "both"
      },
      {
        "eep": "F6-03-01",
        "version": 0.9,
        "direction": "from"
      }
    ],
    "manufacturer": "Manufacturer_Name",
    "firstSeen": "2017-04-04T21:53:03.795+0200",
    "secured": false,
    "iotLabel": "30S000001910188+1P0040000000002+10Z00+11",
    "productId": "0040000000002",
    "remanState": "NotSeen",
    "remanCode": "27DF41AA",
    "softSmartAck": false,
    "transmitModes": [
      {
        "key": "buttonDI",
        "transmitOnConnect": false,
        "transmitOnEvent": true,
        "transmitOnDuplicate": true
      },
      {
        "key": "buttonA0",
        "transmitOnConnect": false,
        "transmitOnEvent": true,
        "transmitOnDuplicate": true
      },
      {
        "key": "buttonB0",
        "transmitOnConnect": false,
        "transmitOnEvent": true,
        "transmitOnDuplicate": true
      },
      {
        "key": "buttonC0",
        "transmitOnConnect": false,
        "transmitOnEvent": true,
        "transmitOnDuplicate": true
      },
      {

```

```

        "key": "buttonD0",
        "transmitOnConnect": false,
        "transmitOnEvent": true,
        "transmitOnDuplicate": true
    },
    {
        "key": "buttonBI",
        "transmitOnConnect": false,
        "transmitOnEvent": true,
        "transmitOnDuplicate": true
    },
    {
        "key": "buttonCI",
        "transmitOnConnect": false,
        "transmitOnEvent": true,
        "transmitOnDuplicate": true
    },
    {
        "key": "switch",
        "transmitOnConnect": true,
        "transmitOnEvent": true,
        "transmitOnDuplicate": false
    },
    {
        "key": "multipleButtons",
        "transmitOnConnect": false,
        "transmitOnEvent": true,
        "transmitOnDuplicate": true
    },
    {
        "key": "buttonAI",
        "transmitOnConnect": false,
        "transmitOnEvent": true,
        "transmitOnDuplicate": true
    }
],
"operable": false,
"supported": true
}
}

```

#### b. Streaming admin-API Event (Device has not been operating)

JSON Object:

```

{
  "header" : {
    "content" : "device",
    "gateway" : "STC-IoT v0.99.39",
    "timestamp" : "2017-06-19T13:35:48.926+0200"
  },
  "device" : {
    "deviceId" : "01910188",
    "friendlyId" : "device",
    "learnInProcedure" : "internetOfThings",
    "eeps" : [ {
      "eep" : "D2-01-11",
      "version" : 0.9,
    }
  ]
}

```

```
    "direction" : "both"
  }, {
    "eep" : "F6-03-01",
    "version" : 0.9,
    "direction" : "from"
  } ],
  "manufacturer" : "Manufacturer_Name",
  "firstSeen" : "2017-04-04T21:53:03.795+0200",
  "secured" : false,
  "iotLabel" : "30S000001910188+1P0040000000002+10Z00+11Z27",
  "productId" : "004000000002",
  "remanState" : "NotSeen",
  "remanCode" : "27DF41AA",
  "softSmartAck" : false,
  "transmitModes" : [ {
    "key" : "buttonDI",
    "transmitOnConnect" : false,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  }, {
    "key" : "buttonA0",
    "transmitOnConnect" : false,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  }, {
    "key" : "buttonB0",
    "transmitOnConnect" : false,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  }, {
    "key" : "buttonC0",
    "transmitOnConnect" : false,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  }, {
    "key" : "buttonD0",
    "transmitOnConnect" : false,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  }, {
    "key" : "buttonBI",
    "transmitOnConnect" : false,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  }, {
    "key" : "buttonCI",
    "transmitOnConnect" : false,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  }, {
    "key" : "switch",
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  }, {
    "key" : "multipleButtons",
    "transmitOnConnect" : false,
```

```

    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  }, {
    "key" : "buttonAI",
    "transmitOnConnect" : false,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : true
  } ],
  "operable" : false,
  "supported" : true
}
}

```

### c. Streaming admin-API Event (Device has been seen for the f

JSON Object:

```

{
  "header" : {
    "content" : "device",
    "gateway" : "STC-IoT v0.99.39",
    "timestamp" : "2017-06-19T13:38:50.775+0200"
  },
  "device" : {
    "deviceId" : "01910188",
    "friendlyId" : "device",
    "learnInProcedure" : "internetOfThings",
    "eeps" : [ {
      "eep" : "D2-01-11",
      "version" : 0.9,
      "direction" : "both"
    }, {
      "eep" : "F6-03-01",
      "version" : 0.9,
      "direction" : "from"
    } ],
    "manufacturer" : "Manufacturer_Name",
    "firstSeen" : "2017-04-04T21:53:03.795+0200",
    "secured" : false,
    "iotLabel" : "30S000001910188+1P004000000002+10Z00+11Z27",
    "productId" : "004000000002",
    "remanState" : "LearnInInProgress",
    "remanCode" : "27DF41AA",
    "softSmartAck" : false,
    "transmitModes" : [ {
      "key" : "buttonDI",
      "transmitOnConnect" : false,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "buttonA0",
      "transmitOnConnect" : false,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : true
    }, {
      "key" : "buttonB0",

```

```
        "transmitOnConnect" : false,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : true
    }, {
        "key" : "buttonC0",
        "transmitOnConnect" : false,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : true
    }, {
        "key" : "buttonD0",
        "transmitOnConnect" : false,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : true
    }, {
        "key" : "buttonBI",
        "transmitOnConnect" : false,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : true
    }, {
        "key" : "buttonCI",
        "transmitOnConnect" : false,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : true
    }, {
        "key" : "switch",
        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false
    }, {
        "key" : "multipleButtons",
        "transmitOnConnect" : false,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : true
    }, {
        "key" : "buttonAI",
        "transmitOnConnect" : false,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : true
    } ],
    "operable" : true,
    "supported" : true
}
}
```

### 3.3.2.1.3.3 QR Code for existing Sensors

#### QR-Code

The IoT learn-in mechanism can also be used for existing unidirectional sensors (including rocker buttons).

As there is no ReMan/ReCom functionality with existing devices, all the information needed to learn in a device must be transmitted in the QR Code.

The QR Code has to be build backwards with the existing information to comply to the device information.

The first two and the last Identifier are used (EURID, ProductId and Name of the device), the third identifier can contain random values.

The Smart EnOcean QR-Code Variation for existing unidirectional devices is defined as follows:

- The necessary Information about the device for the gateway is:
  - EURID (EnOcean Unique Radio Identifier )
  - Manufacturer ID
  - EEP Used
  - Name of the device -> will be used to create the FriendlyName
  
- The ManufacturerId (1Pxxxxxxxxxxxx) will be used to store
  - The manufacturer ID from Digital Concepts (4B) = 1P and the next 4 bytes
    - the flag for the gateway that the learn-in should handle existing sensors = 1P004B
  - The manufacturer of the device = 0C (for example PROBARE, ViCOS GmbH)
    - 1P004BOC
  - The EEP in original notation for the next 6 Bytes (for example D5-00-01)
    - 1P004BOCD50001
  
- The Name of the device (20Zxxxxxxxxxxxx) that will be used for the creation of the friendlyName. The gateway will append numbers to the name to create uniqueness:
  - 20ZRockerButton

### Example QR Text

We assume a device (Window Contact) for which we want to build a QR-Code.

- EURID:                      FEFEAE38
- ManufacturerId:            0C (PROBARE, ViCOS GmbH)
- EEP                            D5-00-01
- Name                         RockerButton

QR-Code for the IoT Learn-In:

30S0000FEFEAE38+1P004BOCD50001+10Z00+20ZRockerButton

### Example QR Picture

QR-Code looks like this:



QR-Codes can be produced for example here: <http://www.qrcode-generator.de/>

### Existing Sensor

If you have scanned the QR Code, the following call will transmit it to the gateway.

**NOTE:** The QR-Code has to be generated according to the following rules: [QR-Code for existing Sensors](#)

#### Client to Gateway:

| Resource Path   | HTTP method |
|-----------------|-------------|
| <i>/devices</i> | <b>POST</b> |

#### Necessary call parameters:

| parameter | datatype | value / formatting | description |
|-----------|----------|--------------------|-------------|
|           |          |                    |             |

|        |                  |        |  |
|--------|------------------|--------|--|
| header | object           | {}     | standard header object with content = devices                      |
| device | object           | {}     | test object  |
|        | learnInProcedure | string | internetOfThings   |
|        | firstSeen        | string | yyyy-mm-ddT hh:mm:ss.sss+hhmm                                      |
|        | iotLabel         | string | 30S0000FEFEAE38+1P004B0CD50001+10Z00+20ZRockeRButton               |
|        |                  |        | Valid Label. Refer to <a href="#">QR-Code for existing Sensors</a> |

**example:**

- App POST command having scanned a QR-Code
- Return Value of the POST command

```

POST /devices
Body:
{
  "header" : {
    "content" : "device",
    "timestamp" : "2015-11-13T20:53:03.795+0100"
  },
  "device" : {
    "learnInProcedure" : "internetOfThings",
    "firstSeen" : "2015-11-13T20:53:03.795+0100",
    "iotLabel" : "30S0000FFBE7880+1P004B49A52001+10Z00+20ZRo
  }
}
Answer of the gateway to a QR- POST command
    
```



```
{
  "header" : {
    "httpStatus" : 200,
    "content" : "device",
    "gateway" : "STC-IOT v0.98.22",
    "timestamp" : "2015-12-18T17:40:41.261+0100"
  },
  "device" : {
    "deviceId" : "FFBE7880",
    "friendlyId" : "RockerButton",
    "communicationType" : "bidirectional",
    "learnInProcedure" : "internetOfThings",
    "eep" : "A5-20-01",
    "manufacturer" : "Thermokon",
    "baseIdChannel" : "0",
    "firstSeen" : "2015-11-13T20:53:03.795+0100",
    "secured" : false,
    "iotLabel" : "30S0000FFBE7880+1P004B49A52001+10Z0020ZRoc",
    "softSmartAck" : false,
    "transmitModes" : [ {
      "key" : "energyInput",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "serviceMode",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "contact",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "temperature",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "batteryLow",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "energyStorageCharged",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "window",
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false
    }, {
      "key" : "valve",
      "transmitOnConnect" : true,
```

```

        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false
    }, {
        "key" : "actuatorObstructed",
        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false
    } ],
    "operable" : true,
    "supported" : true
}
}

```

### 3.3.2.1.4 DELETE

#### Overview

The DELETE /devices/{deviceId} call of the admin-API deletes a device in the database.

#### Single device

- [DELETE /devices/{deviceId}](#) Deletes a device in the database of the gateway

#### 3.3.2.1.4.1 DELETE /devices/{deviceId}

#### DELETE a device

To delete a device, simply call DELETE /devices/{deviceID} with the corresponding deviceID.

The gateway will:

- answer the request
- delete the device
- issue a "device" notification on the streaming API that the device has been deleted

#### Client to Gateway:

| Resource Path       | HTTP method   |
|---------------------|---------------|
| /devices/{deviceId} | <b>DELETE</b> |

example:

The example shows the request for the gateway to delete a device

#### DELETE /devices/002B6CB5

##### Gateway answer for a successful DELETE call:

```
{
  "header": {
    "httpStatus": 200,
    "gateway": "STC-IOT v0.98.22",
    "timestamp": "2015-12-14T12:13:12.713+0100"
  }
}
```

##### Gateway answer for a failed DELETE call:

```
{
  "header": {
    "httpStatus": 400,
    "code": 3000,
    "message": "unknown device 002B6CB5",
    "gateway": "STC-IOT v0.98.22",
    "timestamp": "2015-12-14T12:14:28.426+0100"
  }
}
```

##### Gateway Streaming admin-API after a successful DELETE call

```
.....
{
  "header" : {
    "content" : "device",
    "timestamp" : "2015-12-14T12:17:02.176+0100"
  },
  "device" : {
    "deviceId" : "002B6CB5",
    "friendlyId" : "MyRocker",
    "deleted" : true
  }
}
.....
```

### 3.3.2.2 Profiles

#### Overview

The /profiles calls are helpful, when you want to know the supported devices of the gateway and the detailed JSON description of a supported profile.

#### GET

- [GET /profiles](#)

#### 3.3.2.2.1 GET

#### Overview

The /profiles calls of the admin-API return more information about the profiles than the user API.

#### All profiles

- [GET /profiles](#)

#### Single profile

- [GET /profiles/{eepId}](#)

#### 3.3.2.2.1.1 GET /profiles

#### Profile Overview

Get informations of profiles known to the gateway

In the Admin API, the call "GET /profiles" delivers basically the same content as "GET /profiles" from the User API but with the following additional information:

- Array of Variations
- Version

#### Client to Gateway:

| Resource Path | HTTP method |
|---------------|-------------|
| /profiles     | GET         |

no additional call parameters are necessary

**Response Gateway to Client:**

| parameter                                  |            | datatype         | value / formatting | description  |
|--|------------|------------------|--------------------|--|
| header                                     |            | object           | {}                 | standard header object with content = profiles                             |
| profiles                                   |            | array of objects | [{}]               | array of profile objects   |
|  | eep        | string           | xx-xx-xx           | EnOcean Equipment Profiles, definition of EnOcean radio telegram structure |
|  | title      | string           |                    | short description of eep profile   |
|  | variations | array of objects | [{}]               |  |
| <i>followed by:</i>                        |            |                  |                    |  |
| <i>Per key multiple or none iterations</i> |            |                  |                    |  |

|  |            |           |                  |    |                                  |
|--|------------|-----------|------------------|----|----------------------------------|
|  | variations |           | array of objects | {} | array of an eep variation object |
|  |            | version   | string           |    | eepVariation and version         |
|  |            | variation | string           |    |                                  |
|  | version    |           | string           |    | version of the API               |

**example:**

Basic structure of gateway response:

```

http://hostname:api_port/profiles

{
  "header" : {
    "httpStatus" : 200,
    "content" : "profiles",
    "gateway" : "STC-IoT v0.99.0",
    "timestamp" : "2016-02-22T17:31:48.211+0100"
  },
  "profiles" : [ {
    "eep" : "A5-02-01",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-02",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-03",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-04",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-05",
    "title" : "Temperature Sensors, Temperature Sensor Range"
  } ]
}
    
```

```
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-06",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-07",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-08",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-09",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-0A",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-0B",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-10",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-11",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-12",
    "title" : "Temperature Sensors, Temperature Sensor Range",
    "variations" : [ {
      "version" : "1.0"
    } ]
  } ]
```

```
}, {
  "eep" : "A5-02-13",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "version" : "1.0"
  } ]
}, {
  "eep" : "A5-02-14",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "version" : "1.0"
  } ]
}, {
  "eep" : "A5-02-15",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "version" : "1.0"
  } ]
}, {
  "eep" : "A5-02-16",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "version" : "1.0"
  } ]
}, {
  "eep" : "A5-02-17",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "version" : "1.0"
  } ]
}, {
  "eep" : "A5-02-18",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "version" : "1.0"
  } ]
}, {
  "eep" : "A5-02-19",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "version" : "1.0"
  } ]
}, {
  "eep" : "A5-02-1A",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "version" : "1.0"
  } ]
}, {
  "eep" : "A5-02-1B",
  "title" : "Temperature Sensors, Temperature Sensor Range",
  "variations" : [ {
    "version" : "1.0"
  } ]
}, {
  "eep" : "A5-02-20",
  "title" : "Temperature Sensors, 10 Bit Temperature Sensor"
```



```

    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-02-30",
    "title" : "Temperature Sensors, 10 Bit Temperature Senso
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-04-01",
    "title" : "Temperature and Humidity Sensor, Range 0°C to
    "variations" : [ {
      "version" : "1.0"
    } ], {
      "variation" : "alphaEOS_SENSETF-H",
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-04-02",
    "title" : "Temperature and Humidity Sensor, Range -20°C
    "variations" : [ {
      "version" : "1.0"
    } ], {
      "variation" : "Eltako_FAFT60,FIFT65S,FBH65TFB",
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-04-03",
    "title" : "Temperature and Humidity Sensor, Range -20°C
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-05-01",
    "title" : "Barometric Sensor, Range 500 to 1150 hPa",
    "variations" : [ {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-06-01",
    "title" : "Light Sensor, Range 300lx to 60.000lx",
    "variations" : [ {
      "variation" : "Eltako_FAH60,FAH60B,FAH65S,FIH65S",
      "version" : "1.0"
    } ], {
      "version" : "1.0"
    } ]
  }, {
    "eep" : "A5-06-02",
    "title" : "Light Sensor, Range 0lx to 1.020lx",
    "variations" : [ {
      "version" : "1.0"
    } ], {
      "variation" : "Eltako_FIH65B",
      "version" : "1.0"
    } ]

```

```

    } ]
  }

```

**3.3.2.2.1.2 GET /profiles/{eepId}**

**Profile detail**

Get detailed informations of a standard or a vendor specific EEP variation functionality / functions

In the Admin API, the call "GET /profiles/{eepID}" delivers basically the same content as "GET /profiles{eepID}" from the User API but with the following additional information:

- Array of TransmitModes
- testExists

**Client to Gateway:**

| Resource Path     | HTTP method |
|-------------------|-------------|
| /profiles/{eepId} | <b>GET</b>  |

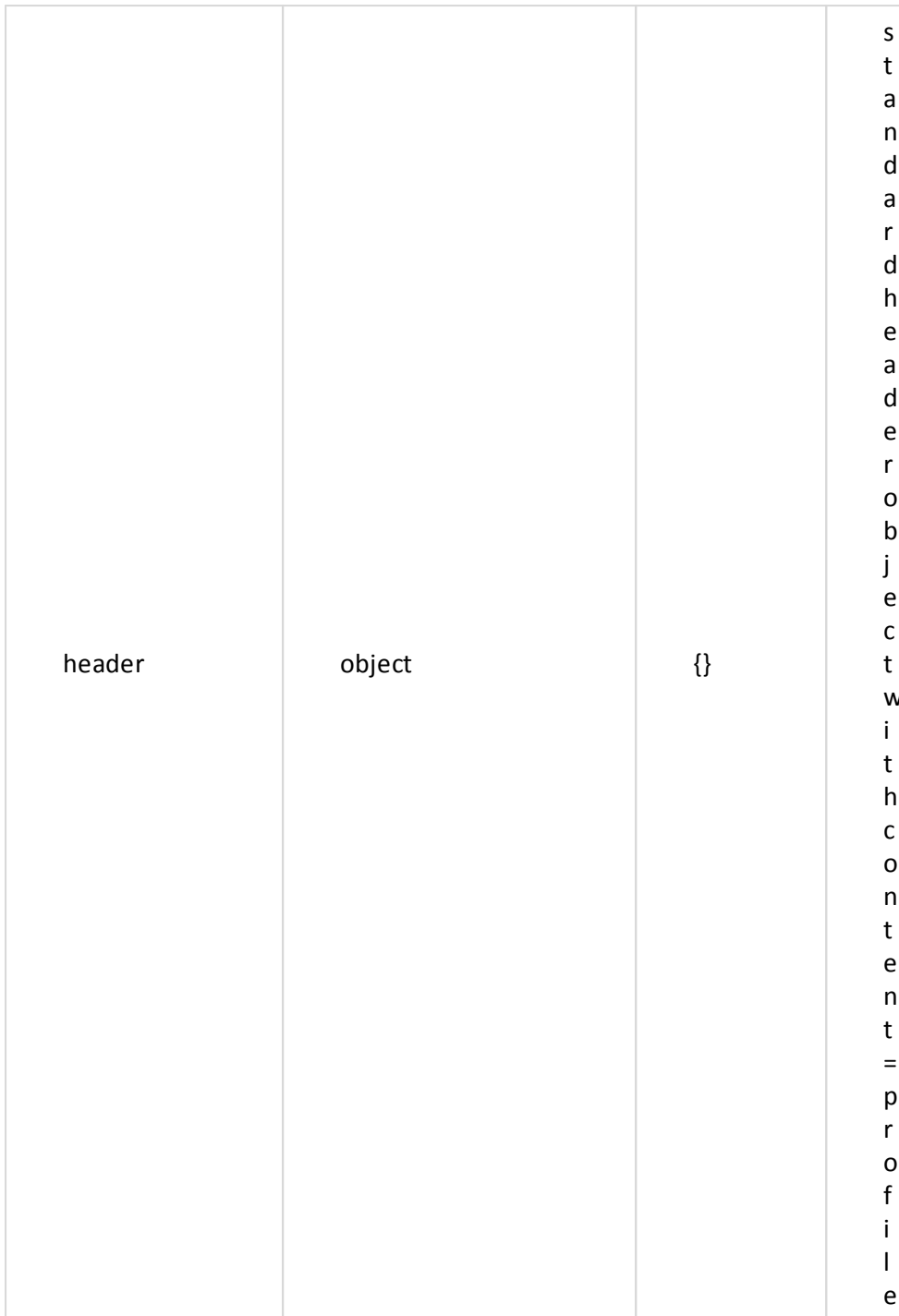
**Necessary and optional additional call parameters:**

| parameter | datatype | valid values | call option | description   |
|-----------|----------|--------------|-------------|---|
| eep       | string   | valid EEP    | required    | EnOcean Equipment Profile, definition of EnOcean radio telegram structure |

|              |        |                    |          |  |
|--------------|--------|--------------------|----------|--|
| manufacturer | string | valid manufacturer | optional | eepVariation and manufacturer, by which the eep profile variation is supported |
| eepVariation | string | valid device       | optional |  |

**response Gateway to Client:**

| parameter | datatype | value / formatting | description |
|-----------|----------|--------------------|-------------|
|-----------|----------|--------------------|-------------|



|         |        |    |   |
|---------|--------|----|---|
| profile | object | {} | p<br>r<br>o<br>f<br>i<br>l<br>e<br>o<br>b<br>j<br>e<br>c<br>t |
|---------|--------|----|---|

|  |     |        |          |  |
|--|-----|--------|----------|--|
|  | eep | string | xx-xx-xx | E<br>n<br>O<br>c<br>c<br>e<br>a<br>n<br>E<br>q<br>u<br>i<br>p<br>m<br>e<br>n<br>t<br>P<br>r<br>o<br>f<br>i<br>l<br>e<br>s<br>,<br>d<br>e<br>f<br>i<br>n<br>i<br>t<br>i<br>o<br>n<br>o<br>f<br>E<br>n<br>o<br>c<br>c<br>e<br>a<br>n<br>r<br>a<br>d<br>i<br>o<br>t |
|--|-----|--------|----------|--|

|  |       |        |  |   |
|--|-------|--------|--|---|
|  |       |        |  | e<br>l<br>e<br>g<br>r<br>a<br>n<br>s<br>t<br>r<br>u<br>c<br>t<br>u<br>r<br>e                                    |
|  | title | string |  | d<br>e<br>s<br>c<br>r<br>i<br>p<br>t<br>i<br>o<br>n<br>o<br>f<br>e<br>e<br>p<br>p<br>r<br>o<br>f<br>i<br>l<br>e |

|  |                |                  |      |   |
|--|----------------|------------------|------|---|
|  | functionGroups | array of objects | [{}] | a<br>r<br>r<br>a<br>y<br>o<br>f<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>G<br>r<br>o<br>u<br>p<br>o<br>b<br>j<br>e<br>c<br>t<br>s |
|--|----------------|------------------|------|---|



|  |           |        |           |   |
|--|-----------|--------|-----------|---|
|  | direction | string | from   to | d<br>i<br>r<br>e<br>c<br>t<br>i<br>o<br>n<br>o<br>f<br>t<br>r<br>a<br>n<br>s<br>p<br>o<br>r<br>t<br>(<br><i>f</i><br><i>r</i><br><i>o</i><br><i>r</i><br><i>t</i><br>o<br>d<br>e<br>v<br>i<br>c<br>e<br>) |
|--|-----------|--------|-----------|---|

|   |           |                  |      |  |
|---|-----------|------------------|------|--|
|   | functions | array of objects | [{}] | a<br>r<br>r<br>a<br>y<br>o<br>f<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>o<br>b<br>j<br>e<br>c<br>t<br>s |
| <i>followed by:</i>   |           |                  |      |  |
| <i>Per key multiple values including value ranges and/or value enumerations</i> |           |                  |      |  |

|  |     |        |  |                                     |
|--|-----|--------|--|-------------------------------------|
|  | key | string |  | key value, as a function identifier |
|--|-----|--------|--|-------------------------------------|

|                                |                         |                         |             |  |
|--------------------------------|-------------------------|-------------------------|-------------|--|
| <p>op<br/>tio<br/>na<br/>l</p> | <p>descriptio<br/>n</p> | <p>string</p>           |             | <p>d<br/>e<br/>s<br/>c<br/>r<br/>i<br/>p<br/>t<br/>i<br/>o<br/>n<br/>o<br/>f<br/>f<br/>u<br/>n<br/>c<br/>t<br/>i<br/>o<br/>n</p> |
|                                | <p>values</p>           | <p>array of objects</p> | <p>[{}]</p> | <p>a<br/>r<br/>r<br/>a<br/>y<br/>o<br/>f<br/>v<br/>a<br/>l<br/>u<br/>e<br/>o<br/>b<br/>j<br/>e<br/>c<br/>t<br/>s</p>             |
| <p><i>value ranges</i></p>     |                         |                         |             |  |

|                 |                 |               |  |  |
|-----------------|-----------------|---------------|--|--|
| <p>optional</p> | <p>valueKey</p> | <p>string</p> |  | <p>c<br/>o<br/>r<br/>r<br/>e<br/>s<br/>p<br/>o<br/>n<br/>d<br/>i<br/>n<br/>g<br/>c<br/>o<br/>n<br/>s<br/>t<br/>a<br/>n<br/>t<br/>o<br/>f<br/>t<br/>h<br/>i<br/>s<br/>v<br/>a<br/>l<br/>u<br/>e</p> |
| <p>optional</p> | <p>meaning</p>  | <p>string</p> |  | <p>m<br/>e<br/>a<br/>n<br/>i<br/>n<br/>g<br/>o<br/>f<br/>v<br/>a<br/>l<br/>u<br/>e</p>   |

|  |       |        |    |   |
|--|-------|--------|----|---|
|  | range | object | {} | r<br>a<br>n<br>g<br>e<br>o<br>b<br>j<br>e<br>c<br>t |
|--|-------|--------|----|---|

|  |     |       |  |  |
|--|-----|-------|--|--|
|  | min | float |  | R<br>e<br>p<br>r<br>e<br>s<br>e<br>n<br>t<br>a<br>t<br>i<br>o<br>n<br>o<br>f<br>t<br>h<br>e<br>v<br>a<br>l<br>i<br>d<br>r<br>a<br>n<br>g<br>e<br>o<br>f<br>v<br>a<br>l<br>u<br>e<br>s<br>w<br>i<br>t<br>h<br>i<br>n<br>c<br>r<br>e<br>m<br>e<br>n<br>t |
|  | max | float |  |  |

|                           |      |        |        |   |
|---------------------------|------|--------|--------|---|
|                           | step | float  |        | a<br>n<br>d<br>,<br>w<br>h<br>e<br>r<br>e<br>a<br>p<br>p<br>r<br>o<br>p<br>r<br>i<br>a<br>t<br>e<br>,<br>t<br>h<br>e<br>c<br>o<br>r<br>r<br>e<br>s<br>p<br>o<br>n<br>d<br>i<br>n<br>g<br>u<br>n<br>i<br>t |
| option<br>al              | unit | string | [unit] |   |
| <i>value enumerations</i> |      |        |        |   |



|          | value    | float  |  |  |
|----------|----------|--------|--|--|
| optional | valueKey | string |  | valid value and additional optional parameters are represented |

|          |         |        |  |  |
|----------|---------|--------|--|--|
| optional | meaning | string |  | entered by : value key ( instead of a number a correct response indicating con |
|----------|---------|--------|--|--|

|  |  |  |  |   |
|--|--|--|--|---|
|  |  |  |  | s<br>t<br>a<br>n<br>t<br>o<br>f<br>t<br>h<br>i<br>s<br>f<br>u<br>n<br>c<br>t<br>i<br>o<br>n<br>)<br>,<br>m<br>e<br>a<br>n<br>i<br>n<br>g<br>(<br>l<br>i<br>m<br>p<br>a<br>c<br>t<br>o<br>f<br>t<br>h<br>e<br>v<br>a<br>l<br>u<br>e<br>) |
|--|--|--|--|---|

|  |                     |        |  |              |
|--|---------------------|--------|--|--------------|
|  | transmitOnConnect   | string |  | True / False |
|  | transmitOnEvent     | string |  | True / False |
|  | transmitOnDuplicate | string |  | True / False |

|  |            |        |  |  |
|--|------------|--------|--|--|
|  | testExists | string |  | S<br>p<br>e<br>c<br>i<br>f<br>i<br>e<br>s<br>w<br>h<br>e<br>t<br>h<br>e<br>r<br>a<br>t<br>e<br>s<br>t<br>f<br>o<br>r<br>t<br>h<br>e<br>d<br>e<br>v<br>i<br>c<br>e<br>e<br>x<br>i<br>s<br>t<br>s<br>i<br>n<br>t<br>h<br>e<br>d<br>a<br>t<br>a<br>b<br>a |
|--|------------|--------|--|--|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | s<br>e<br>o<br>r<br>n<br>o<br>t<br>.<br>T<br>h<br>e<br>t<br>e<br>s<br>t<br>c<br>a<br>n<br>b<br>e<br>u<br>s<br>e<br>d<br>t<br>o<br>i<br>d<br>e<br>n<br>t<br>i<br>f<br>y<br>a<br>d<br>e<br>v<br>i<br>c<br>e<br>. |
|--|--|--|--|--|

|          |              |       |  |   |
|----------|--------------|-------|--|---|
| optional | defaultValue | float |  | for direction, if default value is specified, the parameter |
|----------|--------------|-------|--|---|

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  | t<br>e<br>r<br>c<br>a<br>n<br>b<br>e<br>o<br>n<br>l<br>i<br>n<br>e<br>d<br>w<br>h<br>e<br>n<br>c<br>r<br>e<br>a<br>t<br>i<br>n<br>g<br>a<br>t<br>e<br>l<br>e<br>g<br>r<br>a<br>m |
|--|--|--|--|--|



**example:**

The example shows the profile D2-01-09 *“Electronic switches and dimmers with Energy Measurement and Local Control”*.

Basic structure of gateway response:

```

http://hostname:api_port/profiles/D2-01-09

{
  "header" : {
    "httpStatus" : 200,
    "content" : "profile",
    "gateway" : "STC-IoT v0.98.22",
    "timestamp" : "2015-12-11T17:08:04.939+0100"
  },
  "profile" : {
    "eep" : "D2-01-09",
    "title" : "Electronic switches and dimmers with Energy M
    "functionGroups" : [ {
      "title" : "Actuator Set Output",
      "direction" : "to",
      "functions" : [ {
        "key" : "dimValue",
        "values" : [ {
          "value" : 0,
          "valueKey" : "off",
          "meaning" : "Output value 0% or OFF"
        }, {
          "valueKey" : "on",
          "meaning" : "Output value 1% to 100% or ON",
          "range" : {
            "min" : 1,
            "max" : 100,
            "step" : 1,
            "unit" : "%"
          }
        }
      ] ],
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false,
      "testExists" : true
    }, {
      "key" : "rampingMode",
      "values" : [ {
        "value" : 0,
        "valueKey" : "switch",
        "meaning" : "Switch to new output value"
      }, {
        "value" : 1,

```

```

        "valueKey" : "rampingTime1",
        "meaning" : "Dim to new output value using ramping
    }, {
        "value" : 2,
        "valueKey" : "rampingTime2",
        "meaning" : "Dim to new output value using ramping
    }, {
        "value" : 3,
        "valueKey" : "rampingTime3",
        "meaning" : "Dim to new output value using ramping
    }, {
        "value" : 4,
        "valueKey" : "stop",
        "meaning" : "Stop dimming"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "testExists" : true,
    "defaultValue" : 0
} ]
}, {
    "title" : "Configure Actuator",
    "direction" : "to",
    "functions" : [ {
        "key" : "defaultState",
        "values" : [ {
            "value" : 0,
            "valueKey" : "off",
            "meaning" : "Default state: 0% or OFF"
        }, {
            "value" : 1,
            "valueKey" : "on",
            "meaning" : "Default state: 100% or ON"
        }, {
            "value" : 2,
            "valueKey" : "previousState",
            "meaning" : "Default state: remember previous stat
        } ],
        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false,
        "defaultValue" : 2
    }, {
        "key" : "overcurrentSwitchOffReset",
        "values" : [ {
            "value" : 0,
            "valueKey" : "false",
            "meaning" : "Reset over current shut down: not act
        }, {
            "value" : 1,
            "valueKey" : "true",
            "meaning" : "Reset over current shut down: trigger
        } ],
        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false,

```

```
"defaultValue" : 0
}, {
  "key" : "rampingTime1",
  "values" : [ {
    "meaning" : "Dim timer 1",
    "range" : {
      "min" : 0.5,
      "max" : 7.5,
      "step" : 0.5,
      "unit" : "s"
    }
  } ],
  "transmitOnConnect" : true,
  "transmitOnEvent" : true,
  "transmitOnDuplicate" : false,
  "defaultValue" : 1
}, {
  "key" : "rampingTime2",
  "values" : [ {
    "meaning" : "Dim timer 2",
    "range" : {
      "min" : 0.5,
      "max" : 7.5,
      "step" : 0.5,
      "unit" : "s"
    }
  } ],
  "transmitOnConnect" : true,
  "transmitOnEvent" : true,
  "transmitOnDuplicate" : false,
  "defaultValue" : 4
}, {
  "key" : "rampingTime3",
  "values" : [ {
    "meaning" : "Dim timer 3",
    "range" : {
      "min" : 0.5,
      "max" : 7.5,
      "step" : 0.5,
      "unit" : "s"
    }
  } ],
  "transmitOnConnect" : true,
  "transmitOnEvent" : true,
  "transmitOnDuplicate" : false,
  "defaultValue" : 7.5
}, {
  "key" : "taughtInDevices",
  "values" : [ {
    "value" : 0,
    "valueKey" : "off",
    "meaning" : "Disable taught-in devices (with differ
  }, {
    "value" : 1,
    "valueKey" : "on",
    "meaning" : "Enable taught-in devices (with differ
  } ],
```

```

        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false,
        "defaultValue" : 1
    } ]
}, {
    "title" : "Actuator Status Response",
    "direction" : "from",
    "functions" : [ {
        "key" : "dimValue",
        "values" : [ {
            "value" : 0,
            "valueKey" : "off",
            "meaning" : "Output value 0% or OFF"
        }, {
            "valueKey" : "on",
            "meaning" : "Output value 1% to 100% or ON",
            "range" : {
                "min" : 1,
                "max" : 100,
                "step" : 1,
                "unit" : "%"
            }
        }
    ], {
        "value" : 127,
        "valueKey" : "invalid",
        "meaning" : "output value not valid / not set"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
}, {
    "key" : "errorLevel",
    "values" : [ {
        "value" : 0,
        "valueKey" : "noError",
        "meaning" : "Error level 0: hardware OK"
    }, {
        "value" : 1,
        "valueKey" : "warning",
        "meaning" : "Error level 1: hardware warning"
    }, {
        "value" : 2,
        "valueKey" : "failure",
        "meaning" : "Error level 2: hardware failure"
    }, {
        "value" : 3,
        "valueKey" : "notSupported",
        "meaning" : "Error level not supported"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
}, {
    "key" : "localControl",
    "values" : [ {
        "value" : 0,

```

```

        "valueKey" : "off",
        "meaning" : "Local control disabled / not supported",
    }, {
        "value" : 1,
        "valueKey" : "on",
        "meaning" : "Local control enabled"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
}, {
    "key" : "overcurrentSwitchOff",
    "values" : [ {
        "value" : 0,
        "valueKey" : "false",
        "meaning" : "Over current switch off: ready / not ready"
    }, {
        "value" : 1,
        "valueKey" : "true",
        "meaning" : "Over current switch off: executed"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
} ]
}, {
    "title" : "Actuator Set Measurement",
    "direction" : "to",
    "functions" : [ {
        "key" : "energyDelta",
        "description" : "Delta of energy to be reported",
        "values" : [ {
            "range" : {
                "min" : 0,
                "max" : 4095000,
                "step" : 0.000278,
                "unit" : "Wh"
            }
        } ],
        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false
    }, {
        "key" : "maxTimeBetweenReports",
        "description" : "Measurement Response messages",
        "values" : [ {
            "range" : {
                "min" : 10,
                "max" : 2550,
                "step" : 10,
                "unit" : "s"
            }
        } ],
        "transmitOnConnect" : true,
        "transmitOnEvent" : true,
        "transmitOnDuplicate" : false,
        "defaultValue" : 60
    } ]
} ]

```

```

    }, {
      "key" : "minTimeBetweenReports",
      "description" : "Measurement Response messages",
      "values" : [ {
        "range" : {
          "min" : 0,
          "max" : 255,
          "step" : 1,
          "unit" : "s"
        }
      } ],
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false,
      "defaultValue" : 10
    }, {
      "key" : "reportMeasurement",
      "values" : [ {
        "value" : 0,
        "valueKey" : "queryOnly",
        "meaning" : "Report measurement: query only"
      }, {
        "value" : 1,
        "valueKey" : "queryAndAuto",
        "meaning" : "Report measurement: query / auto report"
      } ],
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false,
      "defaultValue" : 1
    }, {
      "key" : "resetMeasurement",
      "values" : [ {
        "value" : 0,
        "valueKey" : "false",
        "meaning" : "Reset measurement: not active"
      }, {
        "value" : 1,
        "valueKey" : "true",
        "meaning" : "Reset measurement: trigger signal"
      } ],
      "transmitOnConnect" : true,
      "transmitOnEvent" : true,
      "transmitOnDuplicate" : false,
      "defaultValue" : 0
    } ]
  }, {
    "title" : "Actuator Set Measurement",
    "direction" : "to",
    "functions" : [ {
      "key" : "maxTimeBetweenReports",
      "description" : "Measurement Response messages",
      "values" : [ {
        "range" : {
          "min" : 10,
          "max" : 2550,
          "step" : 10,

```

```

        "unit" : "s"
    }
} ],
"transmitOnConnect" : true,
"transmitOnEvent" : true,
"transmitOnDuplicate" : false,
"defaultValue" : 60
}, {
"key" : "minTimeBetweenReports",
"description" : "Measurement Response messages",
"values" : [ {
    "range" : {
        "min" : 0,
        "max" : 255,
        "step" : 1,
        "unit" : "s"
    }
} ],
"transmitOnConnect" : true,
"transmitOnEvent" : true,
"transmitOnDuplicate" : false,
"defaultValue" : 10
}, {
"key" : "powerDelta",
"description" : "Delta of power to be reported",
"values" : [ {
    "range" : {
        "min" : 0,
        "max" : 4095000,
        "step" : 1,
        "unit" : "W"
    }
} ],
"transmitOnConnect" : true,
"transmitOnEvent" : true,
"transmitOnDuplicate" : false
}, {
"key" : "reportMeasurement",
"values" : [ {
    "value" : 0,
    "valueKey" : "queryOnly",
    "meaning" : "Report measurement: query only"
}, {
    "value" : 1,
    "valueKey" : "queryAndAuto",
    "meaning" : "Report measurement: query / auto report"
} ],
"transmitOnConnect" : true,
"transmitOnEvent" : true,
"transmitOnDuplicate" : false,
"defaultValue" : 1
}, {
"key" : "resetMeasurement",
"values" : [ {
    "value" : 0,
    "valueKey" : "false",
    "meaning" : "Reset measurement: not active"
} ]
} ]

```

```

    }, {
      "value" : 1,
      "valueKey" : "true",
      "meaning" : "Reset measurement: trigger signal"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false,
    "defaultValue" : 0
  } ]
}, {
  "title" : "Actuator Query",
  "direction" : "to",
  "functions" : [ {
    "key" : "query",
    "values" : [ {
      "value" : 0,
      "valueKey" : "energy",
      "meaning" : "Query energy"
    }, {
      "value" : 1,
      "valueKey" : "power",
      "meaning" : "Query power"
    }, {
      "value" : 2,
      "valueKey" : "status",
      "meaning" : "Query status"
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  } ]
}, {
  "title" : "Actuator Measurement Response",
  "direction" : "from",
  "functions" : [ {
    "key" : "energy",
    "description" : "Cumulative electricity value from m
    "values" : [ {
      "range" : {
        "min" : 0,
        "max" : 4294967295000,
        "step" : 0.000278,
        "unit" : "Wh"
      }
    } ],
    "transmitOnConnect" : true,
    "transmitOnEvent" : true,
    "transmitOnDuplicate" : false
  }, {
    "key" : "power",
    "description" : "Current power value from meter",
    "values" : [ {
      "range" : {
        "min" : 0,
        "max" : 4294967295000,
        "step" : 1,

```



```

        "unit" : "W"
    }
} ],
"transmitOnConnect" : true,
"transmitOnEvent" : true,
"transmitOnDuplicate" : false
} ]
} ]
}
}

```

### 3.3.2.3 System

#### Overview

The /system resource is helpful, when you want to know or change the Gateway's receiving mode.

#### GET

- [GET /system](#)

#### POST

- [POST /system](#)

#### 3.3.2.3.1 GET

#### Overview

The /system/receiveMode resource returns current state of receiveMode.

#### All devices

- [GET /system/receiveMode](#) Returns the actual Mode of the gateway

#### 3.3.2.3.1.1 GET /system/receiveMode

#### receiveMode

The gateway has two operation modes.

- regular operation mode                      it listens to telegrams
- learn-in Mode                                      it listens to Learn-in Telegrams

This command returns the actual state of the gateway regarding these two modes.

**Client to Gateway:**

| Resource Path                    | HTTP method |
|----------------------------------|-------------|
| <code>/system/receiveMode</code> | <b>GET</b>  |

**Example:**

The example shows the gateway in normal operation mode

| Request:  |
|---|
| GET /system/receiveMode   |
|   |
| Gateway answer:   |
| <pre>{   "header" : {     "httpStatus" : 200,     "content" : "receiveMode",     "gateway" : "STC-IoT v0.98.22",     "timestamp" : "2015-12-11T17:14:34.301+0100"   },   "receiveMode" : "normalMode" }</pre> |

### 3.3.2.3.2 POST

#### Overview

The `/system/receiveMode` resource can be used that the gateway also listens to learn in telegrams.

#### All devices



|                    |               |   |  |
|--------------------|---------------|---|--|
| <p>receiveMode</p> | <p>string</p> | <p>normal<br/>Mode  <br/>learnMo<br/>de</p> | <p>V<br/>a<br/>l<br/>u<br/>e<br/>s<br/>t<br/>o<br/>c<br/>h<br/>a<br/>n<br/>g<br/>e<br/>t<br/>h<br/>e<br/>o<br/>p<br/>e<br/>r<br/>a<br/>t<br/>i<br/>o<br/>n<br/>m<br/>o<br/>d<br/>e<br/>o<br/>f<br/>t<br/>h<br/>e<br/>g<br/>a<br/>t<br/>e<br/>w<br/>a<br/>y</p> |
|--------------------|---------------|---|--|

**Example:**

Set receive mode to learn in mode.

| Request:  |
|---|
| POST /system/receiveMode  |
| <pre>{   "receiveMode" : "learnMode" }</pre>  |
| Gateway answer:   |
| <pre>{   "header": {     "httpStatus": 200,     "content": "receiveMode",     "gateway": "STC-IoT v0.99.1",     "timestamp": "2016-05-10T15:17:44.445+0200"   },   "receiveMode": "learnMode" }</pre> |

### 3.3.3 Reference Client Administrator API

#### Reference Client for Admin-API in Java

We have released a Java reference Client to facilitate the programming on Client side.

It covers the the streaming admin-API and generates all necessary objects for the handling of the JSON messages coming over the streaming admin-API.

Please report us any problems you have or any questions related to the whole Administrator Interface.

## 3.4 Simple API

---

### Overview

The Simple API is the second way to talk to the gateway.

Be aware that there is no order in the Simple API as long as all Key-Value pairs are part of the command.

## Codeset

The codeset used by the Simple API is: ISO-8859-1:1998 (Latin-1, West Europe)

### 3.4.1 Security

## Authentication and Encryption

The Simple API has two mechanisms to secure the IP side of the communication.

- [Basic Authentication](#)
- [TLS Encryption](#)

The EnOcean Radio Security is not yet integrated due to the fact that there are very few devices on the market, that support this feature at the moment. We will implement it in an upcoming release update.

#### 3.4.1.1 Basic Authentication

## Simple API password or Simple API port

To access the Simple API, the client must authenticate itself via Basic access authentication.

The Standard user password for user "user" is "user".

After opening the connection to the Simple API Port, the gateway requests the password of the predefined user "user" in the following manner:

```
login;password=user
```

where "user" is the password that is set on the webinterface.

Settings of Simple API password and port can be made via the webinterface.

Menu item "Admin" / API Port Settings.

### 3.4.1.2 TLS Encryption

#### Certificate

The gateway provides a self signed certificate for the TLS encryption.

The certificate is created out of

- the default name of the gateway "STC-IoT" - CN=STC-IoT
- the name of the organization - O=TK
- and the Country Name - C=DE

Example for Thermokon



These values are generated out of your environment and will vary.

Whenever you change the name of the gateway, you will have to create a new certificate.

#### Enabling encryption

The encryption for each API can be activated or deactivated via the webinterface

Menu item "Admin" / API Port Settings

### 3.4.2 Communication Design

#### Overview

The Simple API consists of one single connection where client and gateway talk to each other in full duplex on a TCP Socket connection.

See also: [https://en.wikipedia.org/wiki/Transmission\\_Control\\_Protocol](https://en.wikipedia.org/wiki/Transmission_Control_Protocol)

#### One Connection

After a client has connected and successfully logged in, the gateway will immediately start a streaming state in which first all the last states of the devices are transmitted followed by the actual ongoing events.

This, together with the “send” command is the main function of the SimpleApi.

#### 3.4.2.1 String basics

#### Overview

Characteristic and structure of the SimpleAPI strings:

- all commands and their responses are formatted in the same way
- each command and response ends with a line break and with a chosen end character
- one line holds a complete object
- each command and response starts with the command or response name
- values in a line are separated with semicolons ‘;’
- function names and their values are separated with an equal sign ‘=’
- timestamps are precise down to the millisecond

#### Formatting example

```
send;deviceId=FFE60503;SetPoint=34(CR/LF)
```



|              |           |          |            |          |           |              |            |                |
|--------------|-----------|----------|------------|----------|-----------|--------------|------------|----------------|
| send         | ;         | deviceId | =          | FFE60503 | ;         | SetPoint     | =          | 34             |
| command name | separator | device   | equal sign | Id       | separator | function key | equal sign | function value |

### Timestamp example

2015-04-22T16:45:46.150+0200

### 3.4.3 From JSON profiles to Simple API Commands

#### Overview

As the simple API and the JSON API transfer the same information for the use of the devices, all Key-Value pairs of a profile for the control of the EnOcean devices exist on both APIs

#### Identifying the Key-Value pairs for a specific profile

For a specific profile, please have a look at the [GET /profiles/{eepid}.pdf](#) function. All Key-Value pairs that are part of one Command Set are grouped together. Keys are valid on both APIs so you can use them in Simple API as well.

#### Example

If you are looking for the syntax for the profile A5-04-01 in order to make a module for it, then follow these steps:

- Identify the Key-Value pairs
  - GET /profiles/A5-04-01.pdf
  - It's a "from" profile where Values are only transmitted from the device
  - Keys are:
    - "humidity"; range 0-100 with a step of 0.4
    - "temperature"; range 0-40 with a step of 0.2

If you are looking for the syntax for the profile D2-01-09 in order to make a module for it, then follow these steps:

- Identify the Key-Value pairs
  - GET /profiles/D2-01-09.pdf
  
- There are now several command groups where each group has a set of Key-Value pairs
  - "To" Groups:
    - Actuator Set Output
      - Keys are:
        - "dimValue", range 0-100, step 1
        - "rampingMode", range 0-4
      - A valid Simple API command would be
        - **send;deviceId=18934F2B;dimValue=50;rampingMode=2**
  
    - Actuator Set Local
      - Keys are:
        - "defaultState", range 0-2
        - "dimTimer1", range 0.5-7.5, step 0.5
        - "dimTimer2", range 0.5-7.5, step 0.5
        - "dimTimer3", range 0.5-7.5, step 0.5
        - "overcurrentSwitchOffReset", range 0-1
        - "taughtInDevices", range 0-1
      - A valid Simple API command would be
        - **send;deviceId=18934F2B;defaultState=1;dimTimer1=2;dimTimer2=4;dimTimer3=6;dimTimer1=2;overcurrentSwitchOffReset=2;taughtInDevices=0**

.... and so on.....

3.4.4 Messages from Gateway

Overview

- [state](#); Transmission of states after login
- [telegram](#); Event driven telegrams from EnOcean devices

3.4.4.1 state;

State

After passing login mask, the gateway does immediately transmit last known state of every learned-in devices. Unlike in other cases, e.g. 'device' or 'telegram' where one line contains all information about one object, the 'state' transmission is slightly different. Functions are separated into different lines. Devices might have different "ages" for different "last states". This means, that different functions of a device are sent by the device at different times. For instance an A5-12-XX profile is sending either actual or cumulative values. This way the different functions and corresponding values have different timestamps.

After the last state has been transmitted, the gateway will subsequently deliver telegrams as they occur on the EnOcean side.

|                            |  |
|----------------------------|--|
| <b>Gateway event state</b> | <pre>state;deviceId=01842329;friendlyId=WeatherStation;dawnSensor=278.15[lx];timestamp=2016-05-24T12:56:07.937+0200;age=24908  state;deviceId=01842329;friendlyId=WeatherStation;dayNight=day(Day);timestamp=2016-05-24T12:56:07.937+0200;age=24908  state;deviceId=01842329;friendlyId=WeatherStation;hemisphere=north(North);timestamp=2016-05-24T12:36:27.399+0200;age=1205446  state;deviceId=01842329;friendlyId=WeatherStation;rainIndication=noRain(No rain);timestamp=2016-05-24T12:56:07.937+0200;age=24908  state;deviceId=01842329;friendlyId=WeatherStation;sunEast=0.00[lx];timestamp=2016-05-24T12:36:27.399+0200;age=1205447  state;deviceId=01842329;friendlyId=WeatherStation;sunSouth=0.00[lx];timestamp=2016-05-24T12:36:27.399+0200;age=1205447  state;deviceId=01842329;friendlyId=WeatherStation;sunWest=588.24[lx];timestamp=2016-05-24T12:36:27.399+0200;age=1205448</pre> |
|----------------------------|--|

```
state;deviceId=01842329;friendlyId=WeatherStation;temperature=26.82[?C];timestamp=2016-05-24T12:56:07.937+0200;age=24910

state;deviceId=01842329;friendlyId=WeatherStation;windSpeed=0.00[m/s];timestamp=2016-05-24T12:56:07.937+0200;age=24910

state;deviceId=019591BE;friendlyId=tempSensor;temperature=21.96[?C];timestamp=2016-05-24T12:45:58.449+0200;age=634399

state;deviceId=018096B5;friendlyId=WaterCon;digitalInput0=high(Digital Input 0 is High);timestamp=2016-05-24T12:52:02.401+0200;age=270448

state;deviceId=018096B5;friendlyId=WaterCon;digitalInput1=high(Digital Input 1 is High);timestamp=2016-05-24T12:52:02.401+0200;age=270448

state;deviceId=018096B5;friendlyId=WaterCon;digitalInput2=high(Digital Input 2 is High);timestamp=2016-05-24T12:52:02.401+0200;age=270449

state;deviceId=018096B5;friendlyId=WaterCon;digitalInput3=high(Digital Input 3 is High);timestamp=2016-05-24T12:52:02.401+0200;age=270449

state;deviceId=018096B5;friendlyId=WaterCon;temperature=24.16[?C];timestamp=2016-05-24T12:52:02.401+0200;age=270449

state;deviceId=018096B5;friendlyId=WaterCon;wakeStatus=high(High value of wake signal);timestamp=2016-05-24T12:52:02.401+0200;age=270450

state;deviceId=018FBB0B;friendlyId=permundoRedDot;dimValue=0.00[%](Output value 0% to 100%);timestamp=2016-05-20T17:27:28.576+0200;age=329344275

state;deviceId=018FBB0B;friendlyId=permundoRedDot;errorLevel=notSupported(Error level not supported);timestamp=2016-05-20T17:27:28.576+0200;age=329344276

state;deviceId=018FBB0B;friendlyId=permundoRedDot;localControl=on(Local control enabled);timestamp=2016-05-20T17:27:28.576+0200;age=329344276

state;deviceId=018FBB0B;friendlyId=permundoRedDot;overcurrentSwitchOff=false(Over current switch off: ready / not supported);timestamp=2016-05-20T17:27:28.576+0200;age=329344276
```

### State formatting

| String     | Meaning  | Possible values/Examples     |
|------------|--|------------------------------|
| state      | respond identifier state   | state                        |
| deviceId   | 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. | 1D59038                      |
| friendlyId | User-assigned name of EnOcean device   | temperatureSensor            |
| function   | Function names and their values<br>(see <a href="#">Function format</a> )  | Temperature=23.53[°C]        |
| timestamp  | The time when the telegram and state received, saved   | 2015-04-22T16:45:46.150+0200 |

#### 3.4.4.2 telegram;

### Telegram

On all events (flag "TransmitOnEvent" true), the gateway transmits an event in form of a telegram over the simple API. If a device has several function groups in the "from" or "both" direction, then the gateway will issue a telegram string for each group subsequently.

|                               |   |
|-------------------------------|---|
| <b>Gateway event telegram</b> | telegram;deviceId=01842329;friendlyId=WeatherStation;timestamp=2016-05-24T13:02:43.701+0200;direction=from;dawnSensor=329.08[lx];temperature=26.82[?C];windSpeed=0.00[m/s];dayNight=day(Day);rainIndication=noRain(No rain) |
|-------------------------------|---|

```
telegram;deviceId=FEFEAB5E;friendlyId=RockerSwitch_dbo;timestamp=2016-05-24T13:03:17.983+0200;direction=from;buttonB0=pressed(Button pressed)

telegram;deviceId=FEFEAB5E;friendlyId=RockerSwitch_dbo;timestamp=2016-05-24T13:03:18.785+0200;direction=from;buttonB0=released(Button released)

telegram;deviceId=01842329;friendlyId=WeatherStation;timestamp=2016-05-24T13:03:48.989+0200;direction=from;dawnSensor=297.74[lx];temperature=26.82[?C];windSpeed=0.00[m/s];dayNight=day(Day);rainIndication=noRain(No rain)
```

### Telegram formatting

| String     | Meaning   | Possible values/Examples  |
|------------|---|---|
| telegram   | event transmits a telegram  | telegram  |
| deviceId   | The ID of the device  | 1859038   |
| friendlyId | String name of the device   | OfficeTemperature   |
| timestamp  | The time when the telegram and state received, saved                        | 2015-04-22T16:45:46.150+0200  |
| direction  | Direction of Telegram.<br>from = Destination Gateway<br>to = Origin Gateway | from<br><br>to  |
| function   | Function names and their values<br>(see <a href="#">Function format</a> )   | SetPoint=213[N/A]<br><br>Temperature=23.53[°C]<br><br>SupplyVoltage |

### 3.4.5 Functions format

#### Functions (Key Value pairs) and formatting

Functions are used in both directions to transmit the information a Device is sending or the command to be executed. Please look at [Key Value Pairs](#) for an overview.

- [Function format used from Gateway to Client](#)
- [Functions used from Client to Gateway](#)

#### 3.4.5.1 From Gateway to Client

#### Receiving functions (from Gateway to Client)

The receiving functions can be within a "[state](#)" or a "[telegram](#)" event.

#### Description

| String    | Meaning                       | Optional |
|-----------|-------------------------------|----------|
| key       | The key of the function       | No       |
| =         | Separator                     | No       |
| value     | Value                         | No       |
| [UNIT]    | Unit enclosed by delimiter    | Yes      |
| (MEANING) | Meaning enclosed by delimiter | Yes      |

#### Examples

|                        | key  | separator | Value    | [Unit] | (Meaning)  |
|------------------------|--|-----------|----------|--------|--|
| function<br>(D2-01-09) | overcurrentSwitchOff   | =         | false    |        | (Over current switch off: ready / not supported) |
|                        | overcurrentSwitchOff=false(Over current switch off: ready / not supported) |           |          |        |  |
| function<br>(A5-13-01) | temperature  | =         | 24.31    | [°C]   |  |
|                        | temperature=24.31[°C]  |           |          |        |  |
| function<br>(A5-13-01) | dayNight   | =         | day      |        | (Day)  |
|                        | dayNight=day(Day)  |           |          |        |  |
| function<br>(F6-02-01) | buttonA0   | =         | released |        | (Button released)                                |
|                        | buttonA0=released(Button released)   |           |          |        |  |
| function<br>(D2-01-09) | dimValue   | =         | 100      | [%]    | (Output value 0% to 100%)                        |
|                        | dimValue=0.00[%](Output value 0% to 100%)                                  |           |          |        |  |

**3.4.5.2 From Client to Gateway**

**Sending functions (from client to gateway)**

The sending functions can be within the "[Commands to Gateway](#)" commands.

**Description**

| String | Meaning                 | Optional |
|--------|-------------------------|----------|
| key    | The key of the function | No       |



|       |                  |    |
|-------|------------------|----|
| =     | Separator        | No |
| Value | Value of the Key | No |

**Examples**

|                     | key                    | separator | Value |
|---------------------|------------------------|-----------|-------|
| function (D2-01-09) | dimValue               | =         | 100   |
|                     | dimValue=100           |           |       |
| function (D2-01-09) | dimValue               | =         |       |
|                     | dimValue=on            |           |       |
| function (A5-20-04) | displayOrientation     | =         | 180   |
|                     | displayOrientation=180 |           |       |

**3.4.6 Commands to Gateway**

**Overview**

- [login;](#) Grant access to the gateway (login;password=user)
- [date;](#) Show current system time of gateway
- [devices;](#) Show registered devices
- [help;](#) Show the help
- [send;](#) Send actions to a device (send;deviceId=AA11CCDD;function=value;function=value)
- [set;](#) Customize output (change visibility of fields or end of line letters)
- [state;](#) Show last states of devices
- [systeminfo;](#) Show baseId and EURID of gateway

- [version;](#) Show version of Smart EnOcean Gateway

**3.4.6.1 date;**

**Date**

The 'date' command returns the actual date. Note that the date, time zone and time format can be edited on the Web Interface.

|                                 |                                   |
|---------------------------------|-----------------------------------|
| <b>User Command/Request</b>     | date                              |
| <b>Gateway Response Example</b> | date;2015-04-21T16:26:31.370+0200 |

**Response formatting**

| String | Meaning                              | Possible values/Examples     |
|--------|--------------------------------------|------------------------------|
| date   | Result of the command by the Gateway | date                         |
|        | Actual date                          | 2015-04-21T16:26:31.370+0200 |

**3.4.6.2 devices;**

**Devices**

The 'devices' command returns all known and saved devices. One device object (line) holds information about the device like its unique ID, the user given name, the EEP profile and when it has been saved (learned in) by the Gateway.

|                             |  |
|-----------------------------|--|
| <b>User Command/Request</b> | devices  |
| <b>Gateway Response</b>     | device;deviceId=FFF26803;friendlyId=Peha_Ch2;eeps=both:D2-01-08##1.0+both:A5-04-01#alphaEOS_SENSETF- |

|                |   |
|----------------|---|
| <b>Example</b> | <pre>H#1.0;firstSeen=2016-05-09T09:27:05.260+0200  device;deviceId=FEFCB98F;friendlyId=ButtonMarion;physicalDevice=Group;location=kitchen;eeps=from:F6-02-01##1.0;firstSeen=2016-05-04T10:45:25.259+0200  device;deviceId=0195DA34;friendlyId=WaterEco;eeps=from:F6-05-01##1.0;firstSeen=2016-04-27T16:22:58.802+0200  device;deviceId=01872C13;friendlyId=CO2Sensor;physicalDevice=AfrisoLabco2;eeps=from:A5-09-08##1.0;firstSeen=2016-03-17T08:53:43.293+0100  device;deviceId=01842329;friendlyId=WeatherStation;eeps=from:A5-13-01##1.0;firstSeen=2016-03-17T09:01:03.867+0100  device;deviceId=12345678;friendlyId=gardenLight;eeps=both:D2-01-09##1.0;firstSeen=2016-04-14T15:05:32.276+0200  device;deviceId=019591BE;friendlyId=tempSensor;eeps=from:A5-02-05##1.0;firstSeen=2016-05-11T08:56:15.288+0200</pre> |
|----------------|---|

**Response formatting**

| String   | Meaning  | Possible values/Examples | Optional |
|----------|--|--------------------------|----------|
| device   | Identifier of the respond  | device                   | no       |
| deviceId | 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. | FEFEAE38                 | no       |

|              |  |   |     |
|--------------|--|---|-----|
| friendlyId   | User-assigned name of EnOcean device               | MyRocker  | no  |
| physicalId   | User-assigned name to group actuators and sensors. | LightGroup  | yes |
| location     | Physical location of the device given by the user  | Office<br>Kitchen   | yes |
| eeps         | EnOcean Equipment Profile of the device            | from:F6-02-01##1.0<br><br>from:A5-04-01##1.0+from:F6-02-01##1.0<br><br>both:D2-01-08##1.0+both:A5-04-01#alphaEOS_SENSE TF-H#1.0 | no  |
| manufacturer | The manufacturer of the device                     | Thermokon<br>Peha<br>Kieback+Peter  | yes |
| firstSeen    | Date when the device has been saved by the Gateway | 2015-04-22T14:21:13.501+0200  | no  |

### 3.4.6.3 help;

#### Help

With 'help', all actual commands and with a short description are listed. One can also get the help information with sending just a carriage return at the beginning of an empty line.

|                                 |  |
|---------------------------------|--|
| <b>User Command/Request</b>     | help   |
| <b>Gateway Response Example</b> | <pre> help;available commands  help;set;eol=:      customize end of line output (set;eol=\r\n)  help;set;visible=: customize visible fields (set;visible=friendlyId,valueKey,unit,meaning)  help;login:        enable access to gateway (login;password=xyz)  help;devices:      show registered devices  help;state:        show last states of devices  help;send:         send actions to a device (send;deviceId=...;function=value;function=value)  help;date:         show current system time of gateway  help;version:      show version of Digital Concepts Gateway  help;systemInfo:  show baseId and EURID of gateway  help;help:        show this information </pre> |

#### 3.4.6.4 login;

##### Overview

After connecting to the Gateway the user has to authorize with a password, in order to operate the SimpleAPI.

|                                 |  |
|---------------------------------|--|
| <b>User Command/Request</b>     | login;password=xzy   |
| <b>Gateway Response Example</b> | <pre> result;code=1000;value=OK  result;code=4000;value=invalid login  result;code=4002;value=already logged in </pre> |

**Response formatting**

| String | Meaning                                     | Examples                                 |
|--------|---|--|
| result | Result of the command by the Gateway        | result                                   |
| code   | Error/success code of the command           | 1000<br>4000<br>4002                     |
| value  | String expression of the Error/success code | OK<br>invalid login<br>already logged in |

**3.4.6.5 send;**

**Send**

The 'send' command creates an EnOcean telegram which is sent out from the gateway. To send a correct telegram the user must address the 'deviceID' which he wants to command and the values what he wants to set in the telegram.

Each EnOcean profile has different functions and number of functions.

|                                 |  |
|---------------------------------|--|
| <b>User Command/Request</b>     | <pre>send;deviceID=;function=;function=;.. send;deviceId=18938BF;Dimming=12;RampingTime=45;DimmingRange=0;StoreFinalValue=0;SwitchingCommand=0</pre> |
| <b>Gateway Response Example</b> | <pre>result;code=1000;value=OK result;code=1002;value=will send telegram later result;code=500;value=Code 3011: no matching group found</pre>        |

|  |  |
|--|--|
|  | <pre>result;code=500;value=Code 3015: invalid value 10.0 of function Setpoint, only scalar allowed</pre> <pre>result;code=500;value=Code 3016: invalid value 100 of function Setpoint (min: -12.7, max:12.8)</pre> |
|--|--|

### Response formatting

| String | Meaning  | Possible values/Examples  |
|--------|--|---|
| result | Result of the Command by the Gateway                                     | result  |
| code   | Error/success code of the command  | 1000<br>500   |
| value  | Meaning of the error code or OK in case the telegram is created and sent | OK<br>Code 3011: no matching group found<br>Code 3015: invalid value 10.0 of function Setpoint, only scalar allowed<br>Code 3016: invalid value 100 of function Setpoint (min: -12.7, max:12.8) |

#### 3.4.6.6 set;

### Overview

The 'set' command is responsible for the 'end of lines' settings and for special values' visibility.

- set;eol customizes the end of line outputs  
For end of line character or expression anything could be used. It could be a simple semicolon ';', empty character or string expression like 'end'. Please note that you have to explicitly define \r\n if you want a line separator in standard DOS-format (Carriage return LineFeed). This is the setting by default.
- set;visible customizes field visibilities like 'friendlyID', 'valueKey', 'unit' or 'meaning'

|                                 |  |
|---------------------------------|--|
| <b>User Command/Request</b>     | <pre>set;eol=; set;visible=friendlyID set;visible=unit set;visible=meaning</pre> |
| <b>Gateway Response Example</b> | <pre>result;code=1000;value=OK result;code=2005;value=invalid data</pre>         |

Response formatting

| String | Meaning                                     | Examples           |
|--------|---|--------------------|
| result | Result of the command by the Gateway        | result             |
| code   | Error/success code of the command           | 1000<br>2005       |
| value  | String expression of the Error/success code | OK<br>invalid data |

3.4.6.7 state;

**State**

The 'state' command returns the actual or last known state of the saved devices. Unlike in other cases, e.g. 'device' or 'telegram' where one line contains all information about one object, the 'state' response is slightly different. Here all functions are separated into different lines. The reason for this is that a complicated device might have different "ages" for different "last states". This means, that different functions of a (complicated) device are sent by the device at different times. For instance an A5-12-XX profile is sending either actual or cumulative values. This way the different functions and corresponding values have different timestamps.



|                                 |   |
|---------------------------------|---|
| <b>User Command/Request</b>     | State   |
| <b>Gateway Response Example</b> | <pre>state;deviceId=1859038;friendlyId=OfficeTemperature;SetPoint=213[N/A];timestamp=2015-04-22T16:45:46.150+0200;age=8543</pre> <pre>state;deviceId=1859038;friendlyId=OfficeTemperature;SlideSwitch0I=1(Slide switch On/Day);timestamp=2015-04-22T16:45:46.150+0200;age=8543</pre> <pre>state;deviceId=1859038;friendlyId=OfficeTemperature;Temperature=23.53[°C];timestamp=2015-04-22T16:45:46.150+0200;age=8543</pre> |

**Response formatting**

| String     | Meaning  | Possible values/Examples                                    |
|------------|--|---|
| state      | Result of the command by the Gateway                 | state   |
| deviceId   | The ID of the device                                 | 1859038   |
| friendlyId | String name of the device                            | OfficeTemperature   |
| function   | Function names and their values                      | SetPoint=213[N/A]<br>Temperature=23.53[°C]<br>SupplyVoltage |
| timestamp  | The time when the telegram and state received, saved | 2015-04-22T16:45:46.150+0200                                |

**3.4.6.8 systeminfo;**

**Systeminfo**

The 'systeminfo' command returns the baseId and EURID of the Gateway.

|                                 |   |
|---------------------------------|---|
| <b>User Command/Request</b>     | systeminfo                                |
| <b>Gateway Response Example</b> | systemInfo;baseId=FFF81980;eurid=01841655 |

### Response formatting

| String | Meaning  | Possible values/Examples |
|--------|--|--------------------------|
| baseID | actual baseId of the Gateway.<br>Can be changed via the webinterface | FFF81980                 |
| eurid  | EURID of the EnOcean Chip.<br>Can not be changed                     | 01841655                 |

#### 3.4.6.9 version;

### Version

The 'version' command returns the actual firmware version of the Gateway. The firmware is updated via the Web Interface.

|                                 |   |
|---------------------------------|---|
| <b>User Command/Request</b>     | version                                   |
| <b>Gateway Response Example</b> | version;STC-IoT v0.99.0c 2016.05.19 17:29 |

### Response formatting

| String | Meaning | Possible values/Examples |
|--------|---------|--------------------------|
|--------|---------|--------------------------|

|         |                                      |                                      |
|---------|--------------------------------------|--------------------------------------|
| version | Result of the command by the Gateway | version                              |
| version | STC-IoT firmware version             | STC-IoT v0.99.0c<br>2016.05.19 17:29 |



# **API Connectors/Client Side**

## 4 API Connectors/Client Side

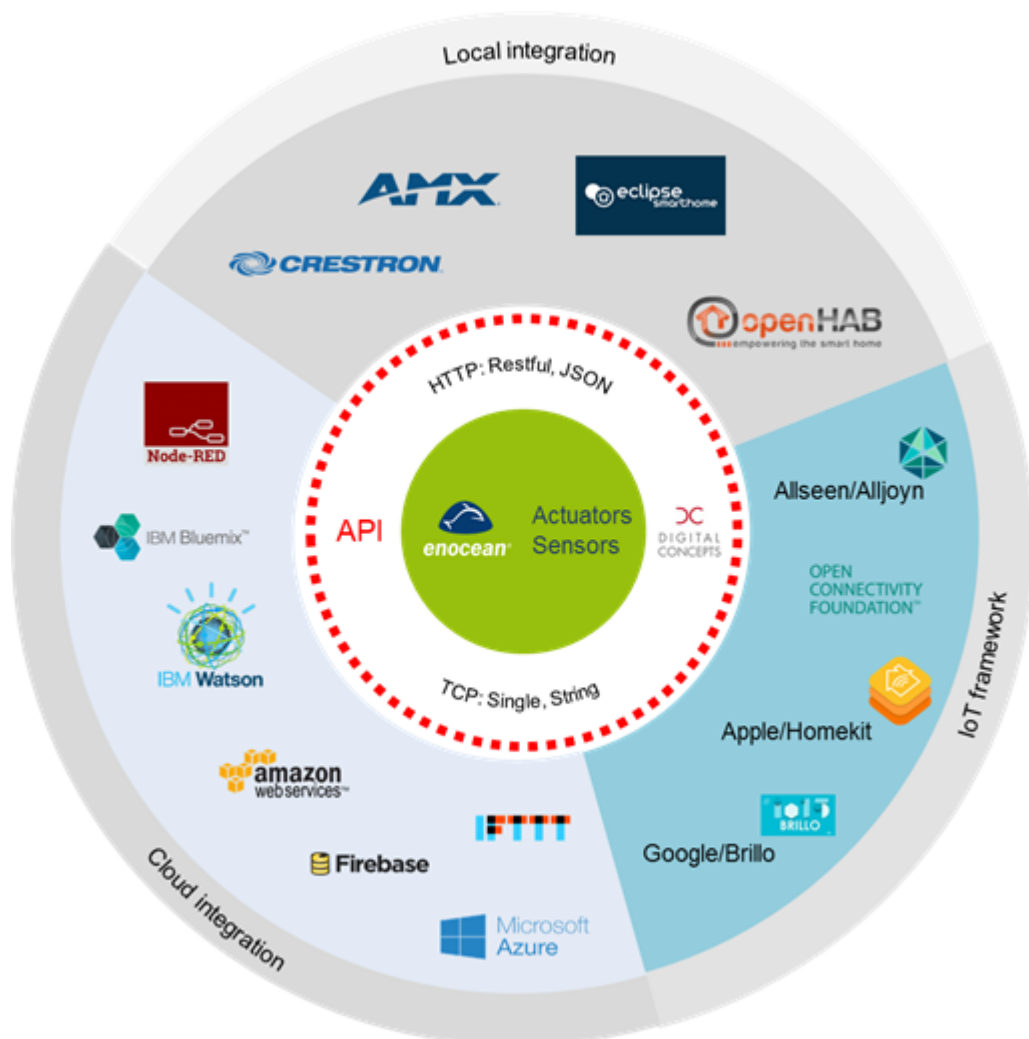
### Overview

This chapter is about the client side of the gateway and the different approaches to build a client side.

We will introduce code examples in different languages shortly.

In the meantime, there is a [Integration guide](#) on how to get started.

- [Integration guide](#)



### 4.1 Integration Guide

For the first steps in getting the Gateway up and running, you can follow the steps below. It is just a rough guideline and gives you an overview of the necessary implementation steps.

| Step                | Goal                     | JSON API  | Simple API   | Remarks   |
|---------------------|--------------------------|---|--|---|
| 1 – Identify        | Identify the right API   | IP Clients with full TCP/IP Stack and “JSON” libraries <ul style="list-style-type: none"> <li>• Windows</li> <li>• Unix</li> <li>• MacOS</li> <li>• etc.</li> </ul> | Industrial Controllers i.e. <ul style="list-style-type: none"> <li>• Siemens</li> <li>• Crestron</li> <li>• AMX</li> <li>• WAGO</li> <li>• Beckhoff</li> <li>• Loxone</li> <li>• etc</li> <li>• ...</li> </ul> | See <a href="#">JSON vs. Simple String</a>  |
| 2 – Device Learn in | Learn in EnOcean devices | Via Administrator API: <ul style="list-style-type: none"> <li>• <a href="#">POST/Device</a></li> </ul>  | -  | Is achieved through Browser User Interface <code>http://&lt;IP-Gateway&gt;</code> |

| Step                   | Goal                                   | JSON API   | Simple API   | Remarks       |
|------------------------|--|--|--|---------------|
| <b>3 – Connect</b>     | Real-Time Connection                   | Two threads in client application besides main(): <ul style="list-style-type: none"> <li>• Receiving Events from the Gateway -&gt; <a href="#">Streaming API</a></li> <li>• Sending Events to the Gateway -&gt; <a href="#">“Keep-alive” connection</a></li> </ul> | One thread in client application besides main(): <ul style="list-style-type: none"> <li>• <a href="#">Full duplex TCP Socket Connection</a></li> </ul> |               |
| <b>4 – Device Info</b> | Get Infos about the Learned in devices | Command: <ul style="list-style-type: none"> <li>• <a href="#">GET /devices</a></li> </ul>  | Command: <ul style="list-style-type: none"> <li>• <a href="#">devices</a></li> </ul>   | No functional |



| Step                    | Goal                         | JSON API  | Simple API    | Remarks  |
|-------------------------|------------------------------|---|---------------|--|
|                         |                              |   |               | difference between APIs                                      |
| <b>5 – Profile Info</b> | Get Profile Infos per Device | Command: <ul style="list-style-type: none"> <li>• <a href="#">GET /devices/{deviceId}/profile</a></li> </ul>                                  | Not Available | Issue Command for each device to get the profile Information |
| <b>6 – Modules</b>      | Build Modules per Profile    | Get the Key Value Pairs associated to each profile and build modules/encapsulations for each profile and/or to the function blocs within more | Not Available | The returned key/Value pairs are nice for human reading      |

| Step | Goal | JSON API  | Simple API | Remarks  |
|------|------|---|------------|--|
|      |      | complex profiles.<br>Command: <ul style="list-style-type: none"> <li>• <a href="#">GET /devices/{deviceId}/profile.pdf</a></li> <li>• or</li> <li>• <a href="#">GET /profiles/{equipmentId}.pdf (ex. GET /profiles/F6-02-01.pdf)</a></li> </ul> |            | and understanding the profile. The data corresponds to the returned JSON description of GET /profile in Step 5. You can also build a generator for the |

| Step    | Goal                    | JSON API | Simple API | Remarks  |
|---------|-------------------------|----------|------------|--|
|         |                         |          |            | modules if you run it against the JSON output of all profiles.             |
| 7 – Run | Let the application run |          |            | You should have full control over events and actions in the EnOcean World. |

## 4.2 Examples for sending commands

---

Over Time, we will add more examples of API Calls

### Examples:

- [Permundo PSC 234](#)

### 4.2.1 Permundo actuator PSC 234

The Permundo PSC 234 uses the EnOcean EEP D2-01-09

This is an example on how to control the device

- [JSON](#)
- [Simple API](#)

#### 4.2.1.1 JSON API

### Sending Telegrams:

To send out a telegram to the actuator, you have to send the following commands over the API:

To turn the actuator on:

```
PUT /devices/{eepid}/state
{
  "state" : {
    "functions" : [
      {
        "key" : "dimValue",
        "value" : "100"
      },
      {
        "key" : "rampingMode",
        "value" : "0"
      }
    ]
  }
}
```

To turn the actuator off:

```
PUT /devices/{eepid}/state
{
  "state" : {
```

```

    "functions" : [
    {
      "key" : "dimValue",
      "value" : "0"
    },
    {
      "key" : "rampingMode",
      "value" : "0"
    }
  ]
}

```

### Sending Telegrams using the "default" option:

if you look at the profile D2-01-09 ([JSON](#), [PDF](#), [html](#)), you can see that the parameter "rampingMode" has a default value of 0/switch. If you don't set the parameter, the default value will be used instead. A valid command to the gateway is then also:

```

PUT /devices/0191388F/state
{
  "state" : {
    "functions" : [
    {
      "key" : "dimValue",
      "value" : "100"
    }
  ]
}

```

To turn the actuator off:

```

PUT /devices/0191388F/state
{
  "state" : {
    "functions" : [
    {
      "key" : "dimValue",
      "value" : "0"
    }
  ]
}

```

### Receiving Telegrams:

Telegrams sent from the device to the gateway produce an event on the API which looks like this:

Actuator has been switched off:

```

{
  "header" : {

```

```

    "content" : "telegram",
    "timestamp" : "2015-10-29T16:57:59.593+0100"
  },
  "telegram" : {
    "deviceId" : "0191388F",
    "friendlyId" : "PSC234",
    "timestamp" : "2015-10-29T16:57:59.593+0100",
    "direction" : "from",
    "functions" : [ {
      "key" : "dimValue",
      "value" : "0",
      "valueKey" : "off",
      "meaning" : "Output value 0% or OFF"
    } ],
    "telegramInfo" : {
      "data" : "046080",
      "status" : "0",
      "dbm" : -80,
      "rorg" : "D2"
    }
  }
}

```

Actuator has been switched on:

```

{
  "header" : {
    "content" : "telegram",
    "timestamp" : "2015-10-29T16:58:56.764+0100"
  },
  "telegram" : {
    "deviceId" : "0191388F",
    "friendlyId" : "PSC234",
    "timestamp" : "2015-10-29T16:58:56.764+0100",
    "direction" : "from",
    "functions" : [ {
      "key" : "dimValue",
      "value" : "100",
      "valueKey" : "on",
      "meaning" : "Output value 1% to 100% or ON",
      "unit" : "%"
    } ],
    "telegramInfo" : {
      "data" : "0460E4",
      "status" : "0",
      "dbm" : -85,
      "rorg" : "D2"
    }
  }
}

```

#### 4.2.1.2 Simple API

### Sending Telegrams:

To send out a telegram to the actuator, you have to send the following commands over the API:

To turn the actuator on:

```
send;deviceId=019498D3;dimValue=100;rampingMode=0
```

To turn the actuator off:

```
send;deviceId=019498D3;dimValue=0;rampingMode=0
```

### **Sending Telegrams using the "default" option:**

if you look at the profile D2-01-09 ([JSON](#), [PDE](#), [html](#)), you can see that the parameter "rampingMode" has a default value. If you don't set the parameter, the default value will be used instead. A valid command also would look like:

To turn the actuator on:

```
send;deviceId=019498D3;dimValue=100
```

To turn the actuator off:

```
send;deviceId=019498D3;dimValue=0
```

### **Receiving Telegrams:**

Telegrams sent from the device to the gateway produce a event on the API which looks like this:

Actuator has been switched off:

```
telegram;deviceId=019498D3;friendlyId=Permundo;timestamp=2015-10-16T17:41:37.05
```

Actuator has been switched on:

```
telegram;deviceId=019498D3;friendlyId=Permundo;timestamp=2015-10-16T17:41:42.228+0200;direction=from;dimValue=100[%]
```





**- A -**

Activation of the encryption 73, 463

**- B -**

Basic Authentication 72, 462

**- C -**

Certificate 73, 463

**- D -**

default 492  
default value 494  
Device functions 56

**- G -**

General Guidelines 9

**- H -**

Help functions 56

**- L -**

Last States 58

**- M -**

Motivation 9

**- P -**

Profile functions 56

**- R -**

Response Codes 259

**- S -**

System functions 56

**- T -**

Transmission functions 56