WEINZIERL

BAOS REST-Services API v3.3

Weinzierl Engineering GmbH
Achatz 3
Burgkirchen, 84508
DE
info@weinzierl.de

1. Introduction

This document details the BAOS RESTful service API. The KNX IP BAOS 777 is the first device in our BAOS line that implements this new service API. The BAOS 777 then has two web APIs, the web services API implemented by BAOS 771/772 and the new RESTful API. The major differences are:

- the urls describe now describe access to resources, and we use HTTP primitimes such as GET, PUT and POST to act on these resources
- where possible we use consistent parameter naming, such as start and count, rather than service specific parameters such as ItemStart, ItemCount and DatapointStart, DatapointCount etc.
- We have added a WebSocket notification system (server push)
- The services have been extended to include support for structured database interpretation (rooms, function types etc)

2. Why REST and what is it ?

REST stands for Representational State Transfer. It relies on a stateless, client-server, cacheable communications protocol.

REST is an architecture style for designing networked applications. The idea is that, rather than using complex mechanisms such as CORBA, RPC or SOAP to connect between machines, simple HTTP is used to make calls between machines.

In many ways, the World Wide Web itself, based on HTTP, can be viewed as a REST-based architecture.

RESTful applications use HTTP requests to post data (create and/or update), read data (e.g., make queries), and delete data. Thus, REST uses HTTP and its methods for all four CRUD (Create/Read/Update/Delete) operations.

REST is a lightweight alternative to mechanisms like RPC (Remote Procedure Calls) and Web Services (SOAP, WSDL, et al.).

Despite being simple, REST is fully-featured; there's basically nothing you can do in Web Services that can't be done with a RESTful architecture.

3. Conventions

- In this document the IP address 10.0.0.103 is used to represent the KNX IP BAOS 777. When using the samples from this document don't forget to change this to the IP address of your BAOS device.
- Placeholders in endpoints or parameters are always all uppercase: for example **`Token token=TOKEN`** TOKEN must be replaced with actual value.
- If not further specified all indices are 1 based

3.1. Standard query parameters

Where possible we try to use the same parameter names across services. Common parameters (which are typically optional) are:

Table 1. Parameters (optional)

start	the start ID of the first resource	
end	the end ID of the last resource (included)	
count	the amount of resources to obtain	
page	for paged access combine with count	
format	true for formatted responses, false for RAW Bytes	boolean

3.2. Security

The BAOS 777 supports both HTTP and HTTPS protocols in addition to the websocket ws and wss protocols for asynchronous indications from the server.

The device certificate is automatically generated and self signed which means you will have to add an exception for the certification in your web browser. The certificate uses the IP address as the common name so it is recommended to assign a static IP address, either by reserving an address in the DHCP server or by configuring the IP address as "manual" on the device directly.

3.3. Supported methods

All services implement HEAD, OPTIONS and GET (read) unless otherwise indicated. Some services implement PUT (update), POST (create) or DELETE (delete) as required.

4. Supported services

A GET request to the endpoint **`/rest`** returns the list of supported services.

Table 2. Implemented services

Service	Description	Notes
device	Returns device information	This service is available to unauthenticated users
login	Authenticates a user and returns session token	This service is available to unauthenticated users
logout	Ends user session	
serveritems	Manages the server items (device configuration)	
datapoints	Manages the datapoints configured via the ETS	
parameters	Returns the user parameter block downloaded via the ETS	
structured	Interprets the structured database information for groups and functions	
timers	Manages timers	
system	System level services, such as restart	

Example 1. rest

Request

Method	URL
GET	http://10.0.0.103/rest

Response

Statuscode	200

Content Response


```
{
    "services" : [
        {
            "service" : "device",
            "url" : "http:\/\/10.0.0.103\/rest\/device"
        },
        {
            "service" : "login",
            "url" : "http:\/\/10.0.0.103\/rest\/login"
        },
        {
            "service" : "logout",
            "url" : "http:\/\/10.0.0.103\/rest\/logout"
        },
        {
            "service" : "serveritems",
            "url" : "http:\/\/10.0.0.103\/rest\/serveritems"
        },
        {
            "service" : "datapoints",
            "url" : "http:\/\/10.0.0.103\/rest\/datapoints"
        },
        {
            "service" : "parameters",
            "url" : "http:\/\/10.0.0.103\/rest\/parameters"
        },
        {
            "service" : "structured",
            "url" : "http:\/\/10.0.0.103\/rest\/structured"
        },
        {
            "service" : "timers",
            "url" : "http:\/\/10.0.0.103\/rest\/timers"
        },
        {
            "service" : "system",
            "url" : "http:\/\/10.0.0.103\/rest\/system"
        }
    ],
    "version" : 33
}
```


5. Authentication

All services, unless otherwise described, are only available to **authenticated users**.

Table 3. unauthorized accessible

/rest/login
/rest/device

Currently there is one user, identified by a username and a password. The username and password can be set in the ETS. Authentication is performed with the *login* and *logout* services.

5.1. Login

NOTE : While sending a login request via http the username and password are send in plain text. This may pose a security risk. To avoid this, the login POST request can be send to ***https://10.0.0.100/rest/login*** and after that both http and https are usable.

Example 2. rest/login

Request

Method	URL
POST	http://10.0.0.103/rest/login

Content Request

```
{
    "password": "admin",
    "username": "admin"
}
```

Response

Statuscode	200

Content Response

```
3c8b531737cbd849bccf15bb9ef09d9c
```

The response is the token which must be send for every authenticated request in the form of a cookie

```
user=%22YOURTOKEN%22
```

This token is IP address based. Resulting in the fact, that every request from the same IP, e.g different browser, different application, that sends this token, is considered authorized.

5.2. Logout

Example 3. rest/logout

Request

Method	URL
POST	http://10.0.0.103/rest/logout

Response

Statuscode	204

6. Credentials

To change either the username, password or both the endpoint **`/rest/device/password`** is available. Every new ETS download will overwrite the values set via this endpoint.

Example 4. rest/device/password

Request

Method	URL
POST	http://10.0.0.103/rest/device/password

Content Request

```
{
    "credentials": {
        "current": {
            "password": "admin",
            "username": "admin"
        },
        "new": {
            "password": "newpassword",
            "username": "newuser"
        }
    }
}
```

Response

Statuscode	204

6.1. Authorization for a request

Currently two ways are supported.

1. via cookie **`user=%22TOKEN%22`**
2. via Authorization header **`Token token=TOKEN`**

6.1.1. Javascript and CORS

6.1.1.1. What is CORS

The following is from wikipedia:

> Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources (e.g. fonts) on a web page to be requested from another domain outside the domain from which the resource originated.
>
> A web page may freely embed images, stylesheets, scripts, iframes, videos and some plugin content (such as Adobe Flash) from any other domain. However embedded web fonts and AJAX (XMLHttpRequest) requests have traditionally been limited to accessing the same domain as the parent web page (as per the same-origin security policy). "Cross-domain" AJAX requests are forbidden by default because of their ability to perform advanced requests (POST, PUT, DELETE and other types of HTTP requests, along with specifying custom HTTP headers) that introduce many cross-site scripting security issues.
>
> CORS defines a way in which a browser and server can interact to safely determine whether or not to allow the cross-origin request. It allows for more freedom and functionality than purely same-origin requests, but is more secure than simply allowing all cross-origin requests. It is a recommended standard of the W3C.

6.1.1.2. Authorization

We have added an exception for the websocket when using Javascript and CORS. As neither cookies or autherization headers can be set on a websocket connection we have added a query parameter to accept the Token. See *websocket* for more information.

Also, we have a number of basic Javscript examples which show how to implement ajax access to the KNX IP BAOS 777 with CORS. These examples are freely available from the *http://www.weinzierl.de* website

7. Device

Example 5. rest/device

Request

Method	URL
GET	http://10.0.0.103/rest/device

Response

Statuscode	200

Content Response

```
{
    "device" : {
        "build_version" : "379",
        "name" : "NEW NAME"
    }
}
```

7.1. Diagnostics

The log files are downloadable as application/x-gzip package via:

Example 6. rest/device/diagnostics

Request

Method	URL
GET	http://10.0.0.103/rest/device/diagnostics

Response

Statuscode	200

Note

This file is encrypted and you can send it to *support@weinzierl.de*[1] for analysis

8. System

Method	URL
POST	*http://10.0.0.100/rest/system/restart*

There are two options here

- reboot the whole device with reboot set to True
- restart only the application on the device with reboot set to False

[1] mailto:support@weinzierl.de

Parameter

{'reboot':True/False}

Note

There will be no response, since the restart is immediately invoked

9. Serveritems

Serveritems represent properties of the BAOS-Object-Server. A list of all serveritems is available at *Section 15.4, "Serveritems list"*

9.1. Overview

A list of available serveritems and their names is obtainable under **`/rest/serveritems`**

Table 4. Parameters (optional)

start	the start ID of the first serveritems
count	the amount of serveritems to obtain

Example 7. rest/serveritems

Request

Method	URL
GET	http://10.0.0.103/rest/serveritems?count=3&start=8

Response

Statuscode	200

Content Response

```
{
    "serveritems" : [
        {
            "id" : 8,
            "name" : "SerialNumber",
            "url" : "http:\/\/10.0.0.103\/rest\/serveritems\/8"
        },
        {
            "id" : 9,
            "name" : "TimeSinceReset",
            "url" : "http:\/\/10.0.0.103\/rest\/serveritems\/9"
        },
        {
            "id" : 10,
            "name" : "BusConnectionState",
            "url" : "http:\/\/10.0.0.103\/rest\/serveritems\/10"
        }
    ]
}
```

9.2. Values and Attributes

To gain more information about serveritems and access their value, the server exposes different endpoints

Table 5. Parameters (optional)

start	the start ID of the first serveritems	integer
count	the amount of serveritems to obtain	integer

format	true for formatted responses, false for RAW Bytes	boolean

9.2.1. Ranged Access

To access more than one item at once the following endpoints can be used

Example 8. rest/serveritems/values

Request

Method	URL
GET	http://10.0.0.103/rest/serveritems/values?count=1&start=8&format=False

Response

Statuscode	200

Content Response

```
{
    "serveritems_values" : [
        {
            "id" : 8,
            "name" : "SerialNumber",
            "read_only" : true,
            "value" : [
                0,
                197,
                0,
                0,
                0,
                0
            ]
        }
    ]
}
```

Example 9. rest/serveritems/values

Request

Method	URL
GET	http://10.0.0.103/rest/serveritems/values?count=1&start=8&format=True

Response

Statuscode	200

Content Response

```
{
    "serveritems_values" : [
        {
            "id" : 8,
            "name" : "SerialNumber",
            "read_only" : true,
            "value" : "00 C5 00 00 00 00"
        }
    ]
}
```

For those serveritems with **read_only** flag set to **`false`**, a new value can be set via a PUT request

Example 10. rest/serveritems/values

Request

Method	URL
PUT	http://10.0.0.103/rest/serveritems/values

Content Request

```
{
    "serveritems_values": [
        {
            "id": 15,
            "value": true
        },
        {
            "id": 20,
            "value": "5.3.52"
        },
        {
            "id": 37,
            "value": "NEW NAME"
        }
    ]
}
```

Response

Statuscode	204

9.2.2. Accessing single items

This can be achieved with **`/rest/serveritems/ITEMID`** or **`/rest/serveritems/ITEMNAME`**

Example 11. rest/serveritems/15

Request

Method	URL
GET	http://10.0.0.103/rest/serveritems/15

Response

Statuscode	200

Content Response

```
{
    "id" : 15,
    "name" : "ProgrammingMode",
    "read_only" : false,
    "value" : true
}
```

Example 12. rest/serveritems/ProgrammingMode

Request

Method	URL
GET	http://10.0.0.103/rest/serveritems/ProgrammingMode

Response

Statuscode	200

Content Response

```
{
    "id" : 15,
    "name" : "ProgrammingMode",
    "read_only" : false,
    "value" : true
}
```

For those serveritems with **read_only** flag set to **`false`**, a new value can be set via a PUT request

Example 13. rest/serveritems/ProgrammingMode

Request

Method	URL
PUT	http://10.0.0.103/rest/serveritems/ProgrammingMode

Content Request

```
{
    "value": true
}
```

Response

Statuscode	204

9.3. Indications

After establishing a *Section 11, "Websocket"* connection, changes on certain serveritems cause the server to send a message to the client.

A List of serveritems which are able to send indications can be found here: *Section 15.4, "Serveritems list"*

Indication

```
{
    "indications" : {
        "type" : "serveritem_ind",
        "values" : [
            {
                "id" : 15,
                "name" : "ProgrammingMode",
                "read_only" : false,
                "value" : false
            }
        ]
    }
}
```

10. Datapoints

10.1. Overview

BAOS devices support a list of datapoints, with unique datapoint id's that range from 1 to a maximum value, such as 1000 or 2500. These datapoints are freely configurable in the ETS (less so if a structured database) and it is possible to end up with a non-contiguous range of active datapoints. For example, we could end up with datapoints 1, 2, 20 and 198 being active. To discover which datapoints are active you can use this service, which simply returns the datapoint id along with a url for the corresponding datapoint resource.

The list of active datapoints, i.e. a discovery service to determine which datapoints are active is accessible via **`/rest/datapoints`**.

Table 6. Parameters (optional)

start	the start ID of the first datapoint	integer
end	the datapoint ID of the last datapoint (included)	integer
count	the amount of datapoints to obtain	integer
page	for paged access combine wih count	integer
filter	filters the datapoints	all,valid,updated

Example 14. rest/datapoints

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints?start=2&end=10

Response

Statuscode	200

Content Response

```
{
    "datapoints" : [
        {
            "id" : 4,
            "url" : "http:\/\/10.0.0.103\/rest\/datapoints\/4"
        },
        {
            "id" : 7,
            "url" : "http:\/\/10.0.0.103\/rest\/datapoints\/7"
        },
        {
            "id" : 10,
            "url" : "http:\/\/10.0.0.103\/rest\/datapoints\/10"
        }
    ]
}
```

Example 15. rest/datapoints

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints?count=3&start=7

Response

Statuscode	200

Content Response

```
{
    "datapoints" : [
        {
            "id" : 7,
            "url" : "http:\/\/10.0.0.103\/rest\/datapoints\/7"
        },
        {
            "id" : 10,
            "url" : "http:\/\/10.0.0.103\/rest\/datapoints\/10"
        },
        {
            "id" : 11,
            "url" : "http:\/\/10.0.0.103\/rest\/datapoints\/11"
        }
    ]
}
```

Example 16. rest/datapoints

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints?count=2&page=2

Response

Statuscode	200

Content Response

```
{
    "datapoints" : [
        {
            "id" : 7,
            "url" : "http:\/\/10.0.0.103\/rest\/datapoints\/7"
        },
        {
            "id" : 10,
            "url" : "http:\/\/10.0.0.103\/rest\/datapoints\/10"
        }
    ]
}
```

10.2. Descriptions

The datapoints descriptions service accesses datapoint descriptions of one or more datapoint objects. Only GET is supported as descriptions are read only and are configured by the ETS.

Table 7. Parameters (optional)

start	the start ID of the first datapoint	integer

end	the datapoint ID of the last datapoint (included)	integer
count	the amount of datapoints to obtain	integer
page	for paged access combine wih count	integer

Example 17. rest/datapoints/descriptions

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/descriptions?start=2&end=10

Response

Statuscode	200

Content Response


```
{
    "datapoints_descriptions" : [
        {
            "datapoint_type" : "11.001",
            "flags" : {
                "is_communication" : true,
                "is_read" : false,
                "is_read_on_init" : true,
                "is_transmit" : false,
                "is_update" : true,
                "is_write" : true
            },
            "id" : 4,
            "name" : "",
            "size" : {
                "bits" : 24,
                "bytes" : 3
            }
        },
        {
            "datapoint_type" : "20.102",
            "flags" : {
                "is_communication" : true,
                "is_read" : false,
                "is_read_on_init" : true,
                "is_transmit" : true,
                "is_update" : true,
                "is_write" : true
            },
            "id" : 7,
            "name" : "",
            "size" : {
                "bits" : 8,
                "bytes" : 1
            }
        },
        {
            "datapoint_type" : "1.002",
            "flags" : {
                "is_communication" : true,
                "is_read" : false,
                "is_read_on_init" : false,
                "is_transmit"
 ... output omitted
```

Example 18. rest/datapoints/descriptions

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/descriptions?count=3&start=7

Response

Statuscode	200

Content Response


```
{
    "datapoints_descriptions" : [
        {
            "datapoint_type" : "20.102",
            "flags" : {
                "is_communication" : true,
                "is_read" : false,
                "is_read_on_init" : true,
                "is_transmit" : true,
                "is_update" : true,
                "is_write" : true
            },
            "id" : 7,
            "name" : "",
            "size" : {
                "bits" : 8,
                "bytes" : 1
            }
        },
        {
            "datapoint_type" : "1.002",
            "flags" : {
                "is_communication" : true,
                "is_read" : false,
                "is_read_on_init" : false,
                "is_transmit" : true,
                "is_update" : false,
                "is_write" : false
            },
            "id" : 10,
            "name" : "",
            "size" : {
                "bits" : 1,
                "bytes" : 1
            }
        },
        {
            "datapoint_type" : "1.002",
            "flags" : {
                "is_communication" : true,
                "is_read" : false,
                "is_read_on_init" : true,
                "is_transmit"
 ... output omitted
```

Example 19. rest/datapoints/descriptions

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/descriptions?count=2&page=2

Response

Statuscode	200

Content Response


```
{
    "datapoints_descriptions" : [
        {
            "datapoint_type" : "20.102",
            "flags" : {
                "is_communication" : true,
                "is_read" : false,
                "is_read_on_init" : true,
                "is_transmit" : true,
                "is_update" : true,
                "is_write" : true
            },
            "id" : 7,
            "name" : "",
            "size" : {
                "bits" : 8,
                "bytes" : 1
            }
        },
        {
            "datapoint_type" : "1.002",
            "flags" : {
                "is_communication" : true,
                "is_read" : false,
                "is_read_on_init" : false,
                "is_transmit" : true,
                "is_update" : false,
                "is_write" : false
            },
            "id" : 10,
            "name" : "",
            "size" : {
                "bits" : 1,
                "bytes" : 1
            }
        }
    ]
}
```


10.3. Values

10.3.1. Ranged Access

The datapoints values service accesses datapoint values of one or more datapoint objects. Both GET and PUT are supported. Once you have the datapoint descriptions you typically only need the datapoint values, which can be obtained with this service, rather than accessing each datapoint individually.

Table 8. Parameters (optional)

start	the start ID of the first datapoint	
end	the datapoint ID of the last datapoint (included)	
count	the amount of datapoints to obtain	
page	for paged access combine with count	

filter	filters the datapoints	all,valid,updated
format	true for formatted responses, false for RAW Bytes	boolean

Example 20. rest/datapoints/values

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/values?start=2&end=10

Response

Statuscode	200

Content Response


```
{
    "datapoints_values" : [
        {
            "Format" : "DPT11",
            "id" : 4,
            "state" : {
                "is_update" : false,
                "is_valid" : false
            },
            "value" : {
                "Day" : 0,
                "Month" : 0,
                "Year" : 2000
            }
        },
        {
            "Format" : "DPT20",
            "id" : 7,
            "state" : {
                "is_update" : false,
                "is_valid" : true
            },
            "value" : 4
        },
        {
            "Format" : "DPT1",
            "id" : 10,
            "state" : {
                "is_update" : false,
                "is_valid" : true
            },
            "value" : false
        }
    ]
}
```

Example 21. rest/datapoints/values

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/values?count=3&start=7

Response

Statuscode	200

Content Response

```
{
    "datapoints_values" : [
        {
            "Format" : "DPT20",
            "id" : 7,
            "state" : {
                "is_update" : false,
                "is_valid" : true
            },
            "value" : 4
        },
        {
            "Format" : "DPT1",
            "id" : 10,
            "state" : {
                "is_update" : false,
                "is_valid" : true
            },
            "value" : false
        },
        {
            "Format" : "DPT1",
            "id" : 11,
            "state" : {
                "is_update" : false,
                "is_valid" : false
            },
            "value" : false
        }
    ]
}
```

Example 22. rest/datapoints/values

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/values?count=2&page=2

Response

Statuscode	200

Content Response

```
{
    "datapoints_values" : [
        {
            "Format" : "DPT20",
            "id" : 7,
            "state" : {
                "is_update" : false,
                "is_valid" : true
            },
            "value" : 4
        },
        {
            "Format" : "DPT1",
            "id" : 10,
            "state" : {
                "is_update" : false,
                "is_valid" : true
            },
            "value" : false
        }
    ]
}
```

10.3.1.1. Setting new value

A new value can be set via a PUT request

See *Section 15.3, "Commands"* for possible values for command parameter.

Example 23. rest/datapoints/values

Request

Method	URL
PUT	http://10.0.0.103/rest/datapoints/values

Content Request

```
{
    "command": 3,
    "datapoints_values": [
        {
            "id": 103,
            "value": {
                "B": 0,
                "G": 0,
                "R": 0
            }
        }
    ]
}
```

Response

Statuscode	204

10.3.2. Accessing single items

Accesses an individual datapoint resource. A datapoint has a number of attributes including: value, description and state. Additional services are available to access the datapoint value or the datapoint description.

Example 24. rest/datapoints/7

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/7

Response

Statuscode	200

Content Response

```
{
    "Format" : "DPT20",
    "description" : {
        "datapoint_type" : "20.102",
        "flags" : {
            "is_communication" : true,
            "is_read" : false,
            "is_read_on_init" : true,
            "is_transmit" : true,
            "is_update" : true,
            "is_write" : true
        },
        "name" : "",
        "size" : {
            "bits" : 8,
            "bytes" : 1
        }
    },
    "id" : 7,
    "state" : {
        "is_update" : false,
        "is_valid" : true
    },
    "value" : 4
}
```

10.3.2.1. Setting new value

A new value can be set via a PUT request.

See *Section 15.3, "Commands"* for possible values for command parameter.

Example 25. rest/datapoints/7

Request

Method	URL
PUT	http://10.0.0.103/rest/datapoints/7

Content Request

```
{
    "command": 3,
    "value": 2
}
```

Response

Statuscode	204

10.4. Indications

After establishing a *Section 11, "Websocket"* connection, changes to datapoint values and states cause the server to send a message to the client.

Note

The format of the values list elements matches the format of the datapoint values *Section 10.3, "Values"*

Indication

```
{
    "indications" : {
        "type" : "datapoint_ind",
        "values" : [
            {
                "Format" : "DPT20",
                "id" : 7,
                "state" : {
                    "is_update" : true,
                    "is_valid" : true
                },
                "value" : 3
            }
        ]
    }
}
```

10.5. History

An addition to viewing the current value, the BAOS is able to store previous values within a database and present them later.

Example 26. rest/datapoints/history

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/history

Response

Statuscode	200

Content Response

```
{
    "history" : [
        {
            "name" : "values",
            "url" : "http:\/\/10.0.0.103\/rest\/datapoints\/history\/values"
        },
        {
            "name" : "count",
            "url" : "http:\/\/10.0.0.103\/rest\/datapoints\/history\/count"
        },
        {
            "name" : "state",
            "url" : "http:\/\/10.0.0.103\/rest\/datapoints\/history\/state"
        }
    ]
}
```

10.5.1. History Count

To determine how many records are stored within a given interval for a given array of datapoints, this service can be used.

Table 9. Parameters

Parameter	Description	Notes
start	The start datapoint id	Optional, defaults to 1
end	The end datapoint id	Optional, defaults to the last datapoint in database
count	The number of datapoints to return	Optional, defaults to max objects
start_timestamp	Date/Time filter, returns all datapoints that happen after the start timestamp	Optional
end_timestamp	Date/Time filter, returns all datapoints that happen before the end timestamp	Optional
format	Boolean value to indicate if the timestamp are unix timestamps (false) or ISO formatted string (true)	Optional, default is true

Example 27. rest/datapoints/history/count

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/history/count?count=2&start=3&start_timestamp=2015-11-09+10%3A00%3A00+&end_timestamp=2015-11-12+11%3A11%3A11

Response

Statuscode	200

Content Response

```
{
    "datapoints_history_count" : {
        "count" : 0,
        "datapoints" : [
            4,
            7
        ]
    }
}
```

10.5.2. History Values

Table 10. Parameters

Parameter	Description	Notes
start	The start datapoint id	Optional, defaults to 1
end	The end datapoint id	Optional, defaults to the last datapoint in database
max_ids	how many ids should be returned maximal	Optional
count	The number of records to return or if combined with page the number of records per page	Optional, defaults to 1000
start_timestamp	Date/Time filter, returns all datapoints that happen after the start timestamp	Optional
end_timestamp	Date/Time filter, returns all datapoints that happen before the end timestamp	Optional
format	Boolean value to determine if the value is formatted or a byte array	Optional, default is true

Parameter	Description	Notes
page	which page should be returned, starts with 1	Optional

Note

The start_timestamp and end_timestamp change there typ according to the format value. This means they represent UNIX time-stamps (seconds since 1.1.1970) in integer form if **`format`** was set to false and must be ISO 8601 Date /Time strings "YYYY-MM-DD HH:MM:SS" if format it set to true

Important

The amount of history values returned by one query is limited to **1000**. To access more than those the query parameters (like page, count…) can be used

Example 28. rest/datapoints/history/values

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/history/values?start=6&end=8

Response

Statuscode	200

Content Response


```
{
    "datapoints_history_values" : [
        {
            "Format" : "DPT20",
            "id" : 7,
            "timestamp" : "2018-02-05 11:21:17",
            "value" : 0
        },
        {
            "Format" : "DPT20",
            "id" : 7,
            "timestamp" : "2018-02-05 11:21:18",
            "value" : 1
        },
        {
            "Format" : "DPT20",
            "id" : 7,
            "timestamp" : "2018-02-05 11:21:19",
            "value" : 2
        },
        {
            "Format" : "DPT20",
            "id" : 7,
            "timestamp" : "2018-02-05 11:21:19",
            "value" : 4
        },
        {
            "Format" : "DPT20",
            "id" : 7,
            "timestamp" : "2018-02-05 11:21:20",
            "value" : 3
        },
        {
            "Format" : "DPT20",
            "id" : 7,
            "timestamp" : "2018-02-05 11:21:21",
            "value" : 4
        }
    ]
}
```

Example 29. rest/datapoints/history/values

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/history/values?start=3&max_ids=2

Response

Statuscode	200

Content Response


```
{
    "datapoints_history_values" : [
        {
            "Format" : "DPT20",
            "id" : 7,
            "timestamp" : "2018-02-05 11:21:17",
            "value" : 0
        },
        {
            "Format" : "DPT20",
            "id" : 7,
            "timestamp" : "2018-02-05 11:21:18",
            "value" : 1
        },
        {
            "Format" : "DPT20",
            "id" : 7,
            "timestamp" : "2018-02-05 11:21:19",
            "value" : 2
        },
        {
            "Format" : "DPT20",
            "id" : 7,
            "timestamp" : "2018-02-05 11:21:19",
            "value" : 4
        },
        {
            "Format" : "DPT20",
            "id" : 7,
            "timestamp" : "2018-02-05 11:21:20",
            "value" : 3
        },
        {
            "Format" : "DPT20",
            "id" : 7,
            "timestamp" : "2018-02-05 11:21:21",
            "value" : 4
        }
    ]
}
```

Example 30. rest/datapoints/history/values

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/history/values?count=4&start=3&page=2

Response

Statuscode	200

Content Response

```
{
    "datapoints_history_values" : [
        {
            "Format" : "DPT1",
            "id" : 10,
            "timestamp" : "2018-02-05 11:21:18",
            "value" : true
        },
        {
            "Format" : "DPT18",
            "id" : 142,
            "timestamp" : "2018-02-05 11:21:18",
            "value" : {
                "Control" : true,
                "Scene" : 1
            }
        },
        {
            "Format" : "DPT20",
            "id" : 7,
            "timestamp" : "2018-02-05 11:21:19",
            "value" : 2
        },
        {
            "Format" : "DPT1",
            "id" : 10,
            "timestamp" : "2018-02-05 11:21:19",
            "value" : false
        }
    ]
}
```

10.5.3. History state

Controls the history capture for one or more datapoints. By default, the capture of history for a datapoint is disabled.

Returns the history capture state for one or more datapoints.

Table 11. Parameters

Parameter	Description	Notes
start	The start id lower bound	Optional, defaults to 1
count	The number of datapoints to return	Optional, defaults to max objects

Example 31. rest/datapoints/history/state

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/history/state?count=2&start=3

Response

Statuscode	200

Content Response

```
{
    "datapoints_history_state" : [
        {
            "active" : false,
            "count" : 0,
            "id" : 4
        },
        {
            "active" : false,
            "count" : 6,
            "id" : 7
        }
    ]
}
```

10.5.3.1. Setting the State

Sets the capture state for one or more datapoints. We expect an array of History State Objects with id and active set. We ignore the count attribute if it is set.

See *Section 15.2, "History States"* for possible values for the **`active`** parameter.

Example 32. rest/datapoints/history/state

Request

Method	URL
PUT	http://10.0.0.103/rest/datapoints/history/state

Content Request

```
{
    "datapoints_history_state": [
        {
            "active": 2,
            "id": 7
        }
    ]
}
```

Response

Statuscode	204

10.6. Addresses

A list of group addresses, assigned by the ETS can be viewed via the `/rest/datapoints/addresses" endpoint

Table 12. Parameters (optional)

start	the start ID of the first datapoint	integer

end	the datapoint ID of the last datapoint (included)	integer
count	the amount of datapoints to obtain	integer
page	for paged access combine wih count	integer

Example 33. rest/datapoints/addresses

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/addresses?start=2&end=10

Response

Statuscode	200

Content Response

```
{
    "datapoints_addresses" : [
        {
            "addresses" : [
                "0\/0\/4"
            ],
            "id" : 4
        },
        {
            "addresses" : [
                "0\/0\/7"
            ],
            "id" : 7
        },
        {
            "addresses" : [
                "0\/0\/10"
            ],
            "id" : 10
        }
    ]
}
```

Example 34. rest/datapoints/addresses

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/addresses?count=3&start=7

Response

Statuscode	200

Content Response

```
{
    "datapoints_addresses" : [
        {
            "addresses" : [
                "0\/0\/7"
            ],
            "id" : 7
        },
        {
            "addresses" : [
                "0\/0\/10"
            ],
            "id" : 10
        },
        {
            "addresses" : [
                "0\/0\/11"
            ],
            "id" : 11
        }
    ]
}
```

Example 35. rest/datapoints/addresses

Request

Method	URL
GET	http://10.0.0.103/rest/datapoints/addresses?count=2&page=2

Response

Statuscode	200

Content Response

```
{
    "datapoints_addresses" : [
        {
            "addresses" : [
                "0\/0\/7"
            ],
            "id" : 7
        },
        {
            "addresses" : [
                "0\/0\/10"
            ],
            "id" : 10
        }
    ]
}
```

11. Websocket

The device allows a websocket connection on **`ws://10.0.0.100/websocket`** or **`wss://10.0.0.100/websocket`**

For this three ways of authorization are implemented:

1. via cookie **`user=%22TOKEN%22`**
2. via Authorization header **`Token token=TOKEN`**
3. via query parameter for example ws://10.0.0.100/websocket?token=TOKEN

Note

if implementing cross-side access via javascript the query parameter must be used since the default websocket API of javascript does not allow setting header or cookies.

12. Timers

At the moment we support two type of timers.

- An One time timer (set a time and date) and this timer fires once at this given point
- An Interval timer (fires between start and end time/date every time a time of x has passed)

12.1. Jobs

At the moment only one jobtype is implemented: setting a datapoint value

12.1.1. set_datapoint_value

Table 13. parameters

id	the id of the datapoint to set	integer
value	the new value of the datapoint	integer
command	allowed values *Section 15.3, “Commands”*	optional, defaults to Set and send = 3

12.2. Getting Timers

12.2.1. Ranged Access

Table 14. Parameters (optional)

start	the start ID of the first datapoint	integer
count	the amount of datapoints to obtain	integer

Example 36. rest/timers

Request

Method	URL
GET	http://10.0.0.103/rest/timers?count=3&start=2

Response

Statuscode	200

Content Response


```
{
    "timers" : [
        {
            "configuration" : {
                "description" : "timer name",
                "job" : {
                    "parameters" : {
                        "id" : 7,
                        "value" : 4
                    },
                    "type" : "set_datapoint_value"
                },
                "trigger" : {
                    "parameters" : {
                        "days" : 0,
                        "end_date" : "2018-02-10 11:21:47",
                        "hours" : 0,
                        "minutes" : 0,
                        "seconds" : 6,
                        "start_date" : "2018-02-05 11:26:47",
                        "weeks" : 0
                    },
                    "type" : "interval"
                }
            },
            "id" : 5
        }
    ]
}
```

12.2.2. Single Access

Example 37. rest/timers/5

Request

Method	URL
GET	http://10.0.0.103/rest/timers/5

Response

Statuscode	200

Content Response


```
{
    "configuration" : {
        "description" : "timer name",
        "job" : {
            "parameters" : {
                "id" : 7,
                "value" : 4
            },
            "type" : "set_datapoint_value"
        },
        "trigger" : {
            "parameters" : {
                "days" : 0,
                "end_date" : "2018-02-10 11:21:47",
                "hours" : 0,
                "minutes" : 0,
                "seconds" : 6,
                "start_date" : "2018-02-05 11:26:47",
                "weeks" : 0
            },
            "type" : "interval"
        }
    },
    "id" : 5
}
```

12.3. Setting Timer

12.3.1. One Time Timer

Example 38. rest/timers

Request

Method	URL
PUT	http://10.0.0.103/rest/timers

Content Request

```
{
    "timers": [
        {
            "configuration": {
                "description": "timer name",
                "job": {
                    "parameters": {
                        "command": 2,
                        "id": 7,
                        "value": 2
                    },
                    "type": "set_datapoint_value"
                },
                "trigger": {
                    "parameters": {
                        "run_date": "2015-08-10 15:30:39"
                    },
                    "type": "date"
                }
            },
            "id": 1
        }
    ]
}
```

Response

Statuscode	204

12.3.2. Interval Timer

For interval timer the following trigger parameter are available:

```
interval : {
    weeks (int) // number of weeks to wait
    days (int) // number of days to wait
    hours (int) // number of hours to wait
    minutes (int) // number of minutes to wait
    seconds (int) // number of seconds to wait
    start_date (str) // starting point for the interval calculation
    end_date (str) // latest possible date/time to trigger on
}
```

Example 39. rest/timers

Request

Method	URL
PUT	http://10.0.0.103/rest/timers

Content Request

```
{
    "timers": [
        {
            "configuration": {
                "description": "timer name",
                "job": {
                    "parameters": {
                        "id": 7,
                        "value": 4
                    },
                    "type": "set_datapoint_value"
                },
                "trigger": {
                    "parameters": {
                        "days": 0,
                        "end_date": "2015-08-11 15:45:34",
                        "hours": 0,
                        "minutes": 0,
                        "seconds": 6,
                        "start_date": "2015-08-10 15:45:34",
                        "weeks": 0
                    },
                    "type": "interval"
                }
            },
            "id": 2
        }
    ]
}
```

Response

Statuscode	204

12.4. Deleting Timer

12.4.1. Ranged Removal

Example 40. rest/timers

Request

Method	URL
DELETE	http://10.0.0.103/rest/timers

Content Request

```
{
    "timers": [
        {
            "id": 2
        },
        {
            "id": 1
        }
    ]
}
```

Response

Statuscode	204

12.4.2. Remove single Timer

Example 41. rest/timers/5

Request

Method	URL
DELETE	http://10.0.0.103/rest/timers/5

Response

Statuscode	204

12.5. Timer configuration

Timers are described with a JavaScript Object Notation (json) configuration format.

Timers have three components.

Attribute	Description	Required
trigger	A scheduled event that will cause the job to be scheduled	M
job	A job, for example writing the value of a datapoint object	M
description	User defined description of the timer.	O

```
{
 'trigger': { 'type': 'date', 'parameters': { 'run_date': '2015-08-27 16:30:05' } },
 'job': { 'type': 'set_datapoint_value', 'parameters': { 'id': 1, 'value': True } },
 'description': 'some user description string'
}
```

12.5.1. Trigger

A trigger is a scheduled timer event that will cause the job to be executed. Currently there are two types of triggers: date and interval.

A trigger object requires two attributes.

Attribute	Description	Required
type	Type is one of 'date' or 'interval'.	M
parameters	Parameters will vary depending on the type and are described below.	M

12.5.1.1. Date Trigger

A date trigger is a one-shot event that will schedule the job to be run when the date (and time) is reached. The date trigger takes a single parameter, run_date, which indicates the date/time to run the job at.

Parameter	Description	Required
run_date	The run_date is given in the ISO 8601 format.	M

```
'trigger': { 'type': 'date', 'parameters': { 'run_date': '2015-08-27 16:30:05' } }
```

12.5.1.2. Interval Trigger

This trigger schedules jobs to be run periodically, on selected intervals.

You can also specify the starting date and ending dates for the schedule through the start_date and end_date parameters, respectively. They can be given as text (in the ISO 8601 format).

If the start date is in the past, the trigger will not fire many times retroactively but instead calculates the next run time from the current time, based on the past start time.

Attribute	Description	Required
start_date	The run_date is given in the ISO 8601 format.	O
weeks (int)	Number of weeks to wait	O
days (int)	Number of days to wait	O
hours (int)	Number of hours to wait	O
minutes (int)	Number of minutes to wait	O
seconds (int)	Number of seconds to wait	O
start_date (str)	Starting point for the interval calculation	O
end_date (str)	Latest possible date/time to trigger on	O

```
'trigger': { 'type': 'interval', 'parameters': { 'start_date': '2015-08-20 13:00:00', 'seconds' : 1, 'end_date': '2015-09-20
 13:00:00' } },
```

12.5.2. Job

The timer job is currently limited to writing the value of a datapoint object.

Attribute	Description	Required
type	set_datapoint_value	M
parameters	The same parameters used in the rest services for setting the datapoint value	M

```
'job': { 'type': 'set_datapoint_value', 'parameters': { 'id': 1, 'value': True } }
```

12.5.3. Description

This is a user-defined description and is not used by the timers.

```
'description': 'some user description string'
```

13. Parameter Bytes

Table 15. Parameters (optional)

start	the start ID of the first datapoint	integer
count	the amount of datapoints to obtain	integer

Example 42. rest/parameters

Request

Method	URL
GET	http://10.0.0.103/rest/parameters?count=3&start=7

Response

Statuscode	200

Content Response

```
{
    "parameters" : [
        105,
        110,
        103
    ]
}
```

14. Structured Information

If a structured ETS database is loaded into the device, the BAOS 777 is able to interpret the given information and present it via REST endpoints. Structural information contains the combination of data points to function blocks and grouping those in rooms. A Dimming actuator in your installation may have two input group objects, one DPT1 (on /off) and one for relative dimming, and one output as DPT 5.001 to represent the current brightness. Those three can be combined into one function block "Dimming actuator with value state".

Furthermore those function blocks can be grouped into rooms.

14.1. Views

To present this information different views are available.

A list of supported views can be accessed via the endpoint **`/rest/structured/views`**

- a room based view
- a function type based view

Example 43. rest/structured/views

Request

Method	URL
GET	http://10.0.0.103/rest/structured/views

Response

Statuscode	200

Content Response

```
{
    "views" : [
        {
            "name" : "rooms",
            "url" : "http:\/\/10.0.0.103\/rest\/structured\/views\/rooms"
        },
        {
            "name" : "function_types",
            "url" : "http:\/\/10.0.0.103\/rest\/structured\/views\/function_types"
        }
    ]
}
```

14.1.1. Rooms view

14.1.1.1. Overview

Example 44. rest/structured/views/rooms

Request

Method	URL
GET	http://10.0.0.103/rest/structured/views/rooms

Response

Statuscode	200

Content Response


```
{
    "groups" : [
        {
            "function_count" : 16,
            "id" : 1,
            "name" : "Building",
            "url" : "http:\/\/10.0.0.103\/rest\/structured\/views\/rooms\/1",
            "view" : "rooms"
        },
        {
            "function_count" : 16,
            "id" : 2,
            "name" : "Room 1",
            "url" : "http:\/\/10.0.0.103\/rest\/structured\/views\/rooms\/2",
            "view" : "rooms"
        },
        {
            "function_count" : 16,
            "id" : 3,
            "name" : "Room 2",
            "url" : "http:\/\/10.0.0.103\/rest\/structured\/views\/rooms\/3",
            "view" : "rooms"
        }
    ]
}
```


14.1.1.2. Details for rooms

For a complete list of supported function types and their dpts see *Function Types*

Example 45. rest/structured/views/rooms/2

Request

Method	URL
GET	http://10.0.0.103/rest/structured/views/rooms/2

Response

Statuscode	200

Content Response

```
{
    "functions" : [
        {
            "function_datapoints" : [
                49
            ],
            "id" : 17,
            "name" : "SWITCH_ACT",
            "room" : 2,
            "type" : 1
        },
        {
            "function_datapoints" : [
                52,
                53
            ],
            "id" : 18,
            "name" : "SWITCH_ACT_STATE",
            "room" : 2,
            "type" : 2
        },
        {
            "function_datapoints" : [
                55,
                56,
                57
            ],
            "id" : 19,
            "name" : "DIMMING_ACT",
            "room" : 2,
            "type" : 3
        },
        {
            "function_datapoints" : [
                58,
                59,
                60

 ... output omitted
```

14.1.1.2.1. Values

Table 16. Parameters

format	true for formatted responses, false for RAW Bytes	boolean
filter	filters the datapoints	all,valid,updated

Example 46. rest/structured/views/rooms/2/values

Request

Method	URL
GET	http://10.0.0.103/rest/structured/views/rooms/2/values

Response

Statuscode	200

Content Response

```
{
    "function_values" : [
        {
            "datapoints_values" : [
                {
                    "Format" : "DPT1",
                    "id" : 49,
                    "state" : {
                        "is_update" : false,
                        "is_valid" : false
                    },
                    "value" : false
                }
            ],
            "id" : 17
        },
        {
            "datapoints_values" : [
                {
                    "Format" : "DPT1",
                    "id" : 52,
                    "state" : {
                        "is
 ... output omitted
```

14.1.2. Function types view

14.1.2.1. Overview

Example 47. rest/structured/views/function_types

Request

Method	URL
GET	http://10.0.0.103/rest/structured/views/function_types

Response

Statuscode	200

Content Response


```
{
    "groups" : [
        {
            "function_count" : 4,
            "id" : 1,
            "name" : "Switch Act",
            "url" : "http:\/\/10.0.0.103\/rest\/structured\/views\/function_types\/1",
            "view" : "function_types"
        },
        {
            "function_count" : 4,
            "id" : 2,
            "name" : "Switch Act State",
            "url" : "http:\/\/10.0.0.103\/rest\/structured\/views\/function_types\/2",
            "view" : "function_types"
        },
        {
            "function_count" : 4,
            "id" : 3,
            "name" : "Dimming Act",
            "url" : "http:\/\/10.0.0.103\/rest\/structured\/views\/function_types\/3",
            "view" : "function_types"
        },
        {
            "function_count" : 4,
            "id" : 4,
            "name" : "Dimming Act Switch State",
            "url" : "http:\/\/10.0.0.103\/rest\/structured\/views\/function_types\/4",
            "view" : "function_types"
        },
        {
            "function_count" : 4,
            "id" : 5,
            "name" : "Dimming Act Value State",
            "url" : "http:\/\/10.0.0.103\/rest\/structured\/views\/function_types\/5",

 ... output omitted
```


14.1.2.2. Details for function type

For a complete list of supported function types and their dpts see *Function Types*

Example 48. rest/structured/views/function_types/1

Request

Method	URL
GET	http://10.0.0.103/rest/structured/views/function_types/1

Response

Statuscode	200

Content Response


```
{
    "functions" : [
        {
            "function_datapoints" : [
                25
            ],
            "id" : 9,
            "name" : "SWITCH_ACT",
            "room" : 1,
            "type" : 1
        },
        {
            "function_datapoints" : [
                49
            ],
            "id" : 17,
            "name" : "SWITCH_ACT",
            "room" : 2,
            "type" : 1
        },
        {
            "function_datapoints" : [
                79
            ],
            "id" : 27,
            "name" : "SWITCH_ACT",
            "room" : 2,
            "type" : 1
        },
        {
            "function_datapoints" : [
                106
            ],
            "id" : 36,
            "name" : "SWITCH_ACT",
            "room" : 3,
            "type" : 1
        }
    ],
    "id" : 1,
    "name" : "Switch Act",
    "view" : "function_types"
}
```


14.1.2.2.1. Values

Table 17. Parameters

format	true for formatted responses, false for RAW Bytes	boolean
filter	filters the datapoints	all,valid,updated

Example 49. rest/structured/views/function_types/1/values

Request

Method	URL
GET	http://10.0.0.103/rest/structured/views/function_types/1/values

Response

Statuscode	200

Content Response


```
{
    "function_values" : [
        {
            "datapoints_values" : [
                {
                    "Format" : "DPT1",
                    "id" : 25,
                    "state" : {
                        "is_update" : false,
                        "is_valid" : false
                    },
                    "value" : false
                }
            ],
            "id" : 9
        },
        {
            "datapoints_values" : [
                {
                    "Format" : "DPT1",
                    "id" : 49,
                    "state" : {
                        "is_update" : false,
                        "is_valid" : false
                    },
                    "value" : false
                }
            ],
            "id" : 17
        },
        {
            "datapoints_values" : [
                {
                    "Format" : "DPT1",
                    "id" : 79,
                    "state" : {
                        "is_update" : false,
                        "is_valid" : false
                    },
                    "value" : false
                }
            ],
            "id" : 27
        },
        {
            "datapoints_va
 ... output omitted
```


14.2. Functions

with this endpoints a flat list of configured functions can be viewed

14.2.1. Overview

Table 18. Parameters (optional)

start	the start ID of the first function	integer

count	the amount of functions to obtain	integer
filter	filters the functions	all,valid,updated

Example 50. rest/structured/functions

Request

Method	URL
GET	http://10.0.0.103/rest/structured/functions?count=2&start=7

Response

Statuscode	200

Content Response

```
{
    "functions" : [
        {
            "function_datapoints" : [
                19
            ],
            "id" : 7,
            "name" : "WIND_ALERT",
            "room" : 1,
            "type" : 28
        },
        {
            "function_datapoints" : [
                22
            ],
            "id" : 8,
            "name" : "OUTDOOR_TEMPERATURE",
            "room" : 1,
            "type" : 29
        }
    ]
}
```

14.2.2. Values

Table 19. Parameters (optional)

start	the start ID of the first function	integer
count	the amount of functions to obtain	integer
filter	filters the functions	all,valid,updated
format	true for formatted responses, false for RAW Bytes	boolean

Example 51. rest/structured/functions/values

Request

Method	URL
GET	http://10.0.0.103/rest/structured/functions/values?count=2&start=10

Response

Statuscode	200

Content Response


```
{
    "function_values" : [
        {
            "datapoints_values" : [
                {
                    "Format" : "DPT1",
                    "id" : 28,
                    "state" : {
                        "is_update" : false,
                        "is_valid" : false
                    },
                    "value" : false
                },
                {
                    "Format" : "DPT1",
                    "id" : 29,
                    "state" : {
                        "is_update" : false,
                        "is_valid" : false
                    },
                    "value" : false
                }
            ],
            "id" : 10
        },
        {
            "datapoints_values" : [
                {
                    "Format" : "DPT1",
                    "id" : 31,
                    "state" : {
                        "is_update" : false,
                        "is_valid" : false
                    },
                    "value" : false
                },
                {
                    "Format" : "DPT3",
                    "id" : 32,
                    "state" : {
                        "is_update" : false,

 ... output omitted
```

14.2.3. Single Access

Example 52. rest/structured/functions/1

Request

Method	URL
GET	http://10.0.0.103/rest/structured/functions/1

Response

Statuscode	200

Content Response

```
{
    "function_datapoints" : [
        1
    ],
    "id" : 1,
    "name" : "TIME",
    "room" : 1,
    "type" : 21
}
```

15. Apendix

15.1. Datapoints formats

for details regarding the DPT types see KNX specification chapter 3 datapoint types

DPT type	Raw Bytes	Formatted
1	[0]	False
2	[3]	{'Control': True, 'Value': True}
3	[3]	{'Control': False, 'StepCode': 3}
4	[103]	g
5	[95]	95
6	[136]	-120
7	[234, 96]	60000
8	[158, 88]	-25000
9	[161, 227]	-250.4
10	[0, 0, 0]	{'Second': 0, 'Hour': 0, 'Minute': 0, 'Week-day': 'Unknown'}
11	[0, 0, 0]	{'Month': 0, 'Day': 0, 'Year': 2000}
12	[186, 44, 43, 21]	3123456789
13	[128, 0, 0, 18]	-2147483630
14	[64, 73, 14, 86]	3.1415
16	[72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 0, 0, 0]	Hello World
18	[0]	{'Control': False, 'Scene': 0}
20	[2]	2
232	[2, 3, 4]	{'B': 4, 'R': 2, 'G': 3}

15.2. History States

Table 20. States

Decimal	Description
1	Clears the previous recorded data
2	Starts the recording of data
3	Clears values and afterwards start recording
4	Stops the recording,it can be resumed via start
5	Stops recording and clears previous data

15.3. Commands

Table 21. Commands

Binary	Decimal	Description
0000	0	No command
0001	1	Set new value
0010	2	Send value on bus
0011	3	Set new value and send on bus
0100	4	Read new value via bus
0101	5	Clear datapoint transmission state

15.4. Serveritems list

Table 22. Serveritems

ID	Name	Example value	Writeable	Sends Indication
1	HardwareType	00 00 C5 07 00 08		
2	HardwareVersion	16		
3	FirmwareVersion	8		
4	KnxManufacturerCod-eDev	197		
	Remarks: Manufacturer code for the device			
5	KnxManufactur-erCodeApp	197		
	Remarks: Manufacturer code for the application			
6	ApplicationId	1804		
7	ApplicationVersion	17		
8	SerialNumber	00 C5 00 00 00 00		
9	TimeSinceReset	7		
	Remarks: Dor the unit see serveritem TimeSinceResetUnit			
10	BusConnectionState	True		True
	Remarks: Is KNX bus connected or not			
11	MaximalBufferSize	4096		
12	LengthOfDescrip-tionString	0		
	Remarks: Every datapoint has an optional descrptions string, this represents its max length. For structured databases its always 0			
13	Baudrate	0		
	Remarks: Only for legacy support			
14	CurrentBufferSize	4096	True	True

ID	Name	Example value	Writeable	Sends Indication
15	ProgrammingMode	False	True	True
	Remarks: State of programming mode			
16	ProtocolVersion	33		
	Remarks: The BAOS binary protocol version			
17	IndicationSending	True	True	True
18	ProtocolVersionWebSer-vices	33		
19	RestServiceProtocolVer-sion	21		
20	IndividualAddress	5.3.52	True	True
	Remarks: The individual KNX address of the device			
21	MacAddress	01:02:03:04:05:06		
22	TunnellingEnabled	True	True	True
	Remarks: KNXnet/IP tunneling active or not			
23	BaosBinaryEnabled	True	True	True
	Remarks: Access via BAOS Binary connection available or not			
24	BaosWebEnabled	True	True	True
	Remarks: Web Services active or not			
25	BaosRestEnabled	True	True	True
	Remarks: REST services active or not			
26	HttpFileEnabled	True	True	True
	Remarks: Webserver active or not			
27	SearchRequestEnabled	True	True	True
	Remarks: Device responds to search requests(yes / no)			
28	IsStructured	True		
	Remarks: Indicates if the current loaded database is structured			
29	MaxManagementClients	1		
	Remarks: Max amount of available Management connections			
30	ConnectedManagement-Clients	0		
31	MaxTunnellingClients	8		
32	ConnectedTunnelling-Clients	0		
33	MaxBaosUdpClients	10		
34	ConnectedBaosUdp-Clients	0		
35	MaxBaosTcpClients	10		
36	ConnectedBaosTcp-Clients	0		
37	DeviceFriendlyName	NEW NAME	True	True
38	MaxDatapoints	144		
39	ConfiguredDatapoints	99		
40	MaxParameterBytes	1071		
41	DownloadCounter	0		
42	IPAssignment	DHCP	True	True

ID	Name	Example value	Writeable	Sends Indication
	Remarks: DHCP or Manual			
43	IPAddress	10.0.0.100	True	True
44	SubnetMask	255.255.255.0	True	True
45	DefaultGateway	10.0.0.1	True	True
46	TimeSinceResetUnit	s	True	True
	Remarks: x=ms, s=seconds, m=minutes ,h= hours			
47	SystemTime	2015-11-24 11:35:24	True	True
48	SystemTimezoneOffset	0	True	True
	Remarks: Reserverd			
49	MenuEnabled	True	True	True
	Remarks: Can values be edited in the device menu			

15.5. Function types

Table 23. Function types

ID	Name	DPT type	DPT description	Input / Output
1	Switching Control	1.001	On/Off	OUT
2	Switching Control with State	1.001	On/Off	OUT
		1.001	State	IN
3	Dimming Control	1.001	On/Off	OUT
		3.007	Relative	OUT
		5.001	Value	OUT
4	Dimming Control with State (On/Off)	1.001	On/Off	OUT
		3.007	Relative	OUT
		1.001	Switch State	IN
5	Dimming Control with State (%)	1.001	On/Off	OUT
		3.007	Relative	OUT
		5.001	Value State	IN
6	Jalousie Control	1.008	Up/Down	OUT
		1.007	Step/Stop	OUT
7	Jalousie Control with State	1.008	Up/Down	OUT
		1.007	Step/Stop	OUT
		5.001	Value State	IN
8	Shutter Control	1.008	Up/Down	OUT
		1.010	Stop	OUT
9	Shutter Control with State	1.008	Up/Down	OUT
		1.010	Stop	OUT
		5.001	Value State	IN
10	Temperature	9.001	State	IN
11	Temperature with Set-point	9.001	State	IN
		9.001	Setpoint	IN / OUT
12	Scene Control	18.001	Number	OUT
13	Presence	1.002	State	IN
14	Window Contact	1.019	State	IN

ID	Name	DPT type	DPT description	Input / Output
15	Door Contact	1.019	State	IN
16	Smoke Alert	1.002	State	IN
17	Water Alert	1.002	State	IN
18	RGB Control with State	232.600	Control	OUT
		232.600	State	IN
21	Time	10.001	State	IN
22	Date	11.001	State	IN
23	HVAC Mode	20.102	Control	IN / OUT
26	Burglary Alert	1.002	State	IN
27	Rain Alert	1.002	State	IN
28	Wind Alert	1.002	State	IN
29	Outdoor Temperature	9.001	State	IN
33	Univ. 1 Bit State	1.001	State	IN
35	Univ. Scaling Control	5.001	Control	OUT
36	Univ. Scaling State	5.001	State	IN
37	Univ. Scaling Control State	5.001	Control	OUT
		5.001	State	IN
38	Univ. 2 Byte Floatvalue State	9.001	State	IN
39	Univ. 4 Byte Floatvalue State	14.001	State	IN
40	Door bell / opener	1.009	Control	OUT
		1.001	State	IN