# digitalSTROM Server JSON

digitalSTROM

Version: v1.3-branch*

August 19, 2015

*Revision: 3c451f5c0c98db7edb9555940b5215106499d5d1

digitalSTROM Alliance
Building Technology Park Zurich
Brandstrasse 33
CH-8952 Schlieren
Switzerland

# Contents

# 1 Introduction

All requests are sent using HTTP GET and parameters added to the query string url like:

```
/json/apartment/setName?name="My␣digitalStrom␣Server"&username=dssadmin&
    password=secret
```

If not properly authenticated the HTTP Status 403 is returned and the error response contains:

```
{
  "ok":false,
  "message":"not␣logged␣in"
}
```

If an unknown method is requested the error message "Unhandled Function" is returned:

```
{
   "ok": false,
   "message": "Unhandled␣function"
}
```

If a request has been successfully processed the JSON answer contains an "ok" and an optional "result" field. The result array is explained in the particular sections.

| | |
|---|---|
| ok | true |
| result | array of result values |

Where Group Names are allowed the following table lists the possible names.

| Name | Group Id | Description |
|---|---|---|
| yellow | 1 | Light |
| gray | 2 | Light |
| blue | 3 | Climate |

# 2 Apartment

## 2.1 Name

### 2.1.1 getName

Returns the user defined name of the installation.

**Synopsis**
HTTP GET /json/apartment/getName

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| name | identifier string for the installation |

**Sample**

```
GET /json/apartment/getName
{
    "ok":true,
    "result" :
    {
        "name" : "digitalStrom␣Installation␣Hans␣Mustermann"
    }
}
```

### 2.1.2 setName

Sets the installation name.

**Synopsis**
HTTP GET /json/apartment/setName

| Parameter | Description | Remarks |
|---|---|---|
| newName | identifier string for the installation | Mandatory |

**Parameter**

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/apartment/setName?newName="My␣dSS"
{
    "ok":true
}
```

## 2.2 Scene

### 2.2.1 callScene

Excutes the scene *sceneNumber* on a group of devices.

**Synopsis**
HTTP GET /json/apartment/callScene

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneNumber | Numerical value | Mandatory |
| groupID | Number of the target group | Optional |
| groupName | Name of the target group | Optional |
| force | Boolean value, if set a forced scene call is issued | Optional |

If the group parameters are omitted the command is sent as broadcast to all zones and all devices.

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/apartment/callScene?sceneNumber=65
{
    "ok":true
}
```

### 2.2.2 saveScene

Tells devices to store their current output values as a default for the scene *sceneNumber*.

**Synopsis**
HTTP GET /json/apartment/saveScene

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneNumber | Numerical value | Mandatory |
| groupID | Number of the target group | Optional |
| groupName | Name of the target group | Optional |

If the group parameters are omitted the command is sent as broadcast to all zones and all devices.

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/apartment/saveScene?sceneNumber=65
{
    "ok":true
}
```

### 2.2.3 undoScene

Tells devices to restore their output values to the previous state if the current scene matches the *sceneNumber*.

**Synopsis**
HTTP GET /json/apartment/undoScene

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneNumber | Numerical value | Mandatory |
| groupID | Number of the target group | Optional |
| groupName | Name of the target group | Optional |

If the group parameters are omitted the command is sent as broadcast to all zones and all devices.

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/apartment/undoScene?sceneNumber=65
{
    "ok":true
}
```

### 2.2.4 getLockedScenes

Retrieves scene numbers of scenes that are currently locked because of an update of device scene tables.

**Synopsis**
HTTP GET /json/apartment/getLockedScenes

**Parameter** None

**Response**
HTTP Status 200

| | |
|---|---|
| result.lockedScenes[] | array of scene numbers that are currently locked |

**Sample**

```
GET /json/apartment/getLockedScenes
{
    "ok" : true,
    "result" :
    {
        "lockedScenes" : []
    }
}
```

## 2.3 Value

### 2.3.1 Set Device Output Value

Set the output value of a group of devices to a given value.

> **Notice** Setting output values directly bypasses the group state machine and is unrecommended.

**Synopsis**
HTTP GET /json/apartment/setValue

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| value | Numerical value | Mandatory |
| groupID | Number of the target group | Optional |
| groupName | Name of the target group | Optional |

If the group parameters are omitted the command is sent as broadcast to all devices.

> **Notice** Setting output values without a group identification is strongly unrecommended.

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/apartment/setValue?value=0&groupID=2
{
    "ok":true,
}
```

## 2.4 Groups

### 2.4.1 getReachableGroups

Returns a list of groups for which are actuators actually present in the installation.

**Synopsis**
HTTP GET /json/apartment/getReachableGroups

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| result.zones | array of zones in the installation |
| result.zones[].groups | array of groups in a zone |

**Sample**

```
GET /json/apartment/getReachableGroups
{

    "ok": true,
    "result": {
        "zones": [
            {
                "zoneID": 1223,
                "name": "Wohnen",
                "groups": [
                    1,
                    2,
                    7
                ]
            },
            {
                "zoneID": 1241,
                "name": "Schlafen",
                "groups": [
                    1,
                    5,
                    7
                ]
            },
            {
                "zoneID": 1237,
                "name": "Essen",
                "groups": [
                    1,
                    6
                ]
            }
        ]
```

```
    }
}
```

## 2.5 Structure

### 2.5.1 getStructure

Returns an object containing the structure of the apartment. This includes detailed information about all zones, groups and devices.

**Synopsis**
HTTP GET /json/apartment/getStructure

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| result.apartment.zones | array of zone information |
| result.apartment.zones[].devices | array of device information in each zone |
| result.apartment.zones[].devices[].groups | group membership of each device |
| result.apartment.zones[].groups | array of group information in each zone |
| result.apartment.zones[].groups[].devices | array of devices per group in a zone |

**Sample**

```
GET /json/apartment/getStructure
{

    "ok": true,
    "result": {
        "apartment": {
            "zones": [
                {
                    "id": 0,
                    "name": "",
                    "isPresent": false,
                    "devices": [
                        {
                            "id": "3504175fe0000000000182f6",
                            "name": "Regalleuchte",
                            "functionID": 4152,
                            "productRevision": 49955,
                            "productID": 6344,
                            "hwInfo": "GE-SDS200",
                            "meterDSID": "3504175fe0000010000003dd",
                            "busID": 97,
                            "zoneID": 989,
                            "isPresent": false,
                            "lastDiscovered": "2012-10-24␣11:17:29",
```

```
        "firstSeen": "2012-10-22 16:22:02",
        "inactiveSince": "2012-10-22 16:22:02",
        "outputMode": 22,
        "buttonID": 0,
        "buttonActiveGroup": 1,
        "buttonInputMode": 0,
        "buttonInputIndex": 0,
        "buttonInputCount": 1,
        "groups": [
            "1"
        ]
    },
    {
        "id": "3504175fe00000000000439c",
        "name": "Stehlampe",
        "functionID": 4152,
        "productRevision": 789,
        "productID": 200,
        "hwInfo": "GE-KM200",
        "meterDSID": "3504175fe0000010000003dd",
        "busID": 153,
        "zoneID": 989,
        "isPresent": false,
        "lastDiscovered": "2012-10-24 11:17:29",
        "firstSeen": "2012-10-22 16:22:02",
        "inactiveSince": "2012-10-22 16:22:02",
        "outputMode": 22,
        "buttonID": 0,
        "buttonActiveGroup": 1,
        "buttonInputMode": 0,
        "buttonInputIndex": 0,
        "buttonInputCount": 1,
        "groups": [
            "1"
        ]
    },
    {
        "id": "3504175fe0000000000151fd",
        "name": "Fernseher",
        "functionID": 33041,
        "productRevision": 41761,
        "productID": 5320,
        "hwInfo": "SW-ZWS200",
        "meterDSID": "3504175fe0000010000003dd",
        "busID": 693,
        "zoneID": 989,
        "isPresent": false,
        "lastDiscovered": "2012-10-24 11:17:29",
        "firstSeen": "2012-10-22 16:22:02",
        "inactiveSince": "2012-10-22 16:22:02",
        "outputMode": 39,
        "buttonID": 0,
        "buttonActiveGroup": 5,
        "buttonInputMode": 0,
        "buttonInputIndex": 0,
        "buttonInputCount": 1,
        "groups": [
            "5",
            "8"
        ]
    },
    {
```

```
        "id": "3504175fe000000000001234",
        "name": "Wandlampe",
        "functionID": 4144,
        "productRevision": 789,
        "productID": 1234,
        "hwInfo": "GE-TKM210",
        "meterDSID": "3504175fe0000010000003dd",
        "busID": 782,
        "zoneID": 989,
        "isPresent": false,
        "lastDiscovered": "2012-10-24 11:17:29",
        "firstSeen": "2012-10-22 16:22:02",
        "inactiveSince": "2012-10-22 16:22:02",
        "outputMode": 22,
        "buttonID": 4,
        "buttonActiveGroup": 1,
        "buttonInputMode": 0,
        "buttonInputIndex": 0,
        "buttonInputCount": 1,
        "groups": [
            "1"
        ]
    },
    {
        "id": "3504175fe0000000000043a7",
        "name": "Deckenlicht",
        "functionID": 4152,
        "productRevision": 789,
        "productID": 200,
        "hwInfo": "GE-KM200",
        "meterDSID": "3504175fe0000010000003dd",
        "busID": 784,
        "zoneID": 1038,
        "isPresent": true,
        "lastDiscovered": "2012-10-26 15:36:30",
        "firstSeen": "2012-10-22 16:22:02",
        "inactiveSince": "1970-01-01 01:00:00",
        "outputMode": 22,
        "buttonID": 5,
        "buttonActiveGroup": 1,
        "buttonInputMode": 0,
        "buttonInputIndex": 0,
        "buttonInputCount": 1,
        "groups": [
            "1"
        ]
    },
    {
        "id": "3504175fe0000000000042dc",
        "name": "Paniktaster",
        "functionID": 24896,
        "productRevision": 790,
        "productID": 1225,
        "hwInfo": "RT-TKM201",
        "meterDSID": "3504175fe0000010000003dd",
        "busID": 785,
        "zoneID": 989,
        "isPresent": false,
        "lastDiscovered": "2012-10-24 11:17:29",
        "firstSeen": "2012-10-23 16:23:38",
        "inactiveSince": "2012-10-24 11:01:40",
        "outputMode": 0,
```

```
            "buttonID": 17,
            "buttonActiveGroup": 154,
            "buttonInputMode": 20,
            "buttonInputIndex": 0,
            "buttonInputCount": 0,
            "groups": [
                "6"
            ]
        }
    ],
    "groups": [
        {
            "id": 0,
            "name": "broadcast",
            "isPresent": false,
            "devices": [
                "3504175fe0000000000182f6",
                "3504175fe00000000000439c",
                "3504175fe0000000000151fd",
                "3504175fe00000000001234",
                "3504175fe0000000000043a7",
                "3504175fe0000000000042dc"
            ]
        },
        {
            "id": 1,
            "name": "yellow",
            "isPresent": true,
            "devices": [
                "3504175fe0000000000182f6",
                "3504175fe00000000000439c",
                "3504175fe00000000001234",
                "3504175fe0000000000043a7"
            ]
        },
        {
            "id": 2,
            "name": "gray",
            "isPresent": true,
            "devices": [ ]
        },
        {
            "id": 3,
            "name": "blue",
            "isPresent": true,
            "devices": [ ]
        },
        {
            "id": 4,
            "name": "cyan",
            "isPresent": true,
            "devices": [ ]
        },
        {
            "id": 5,
            "name": "magenta",
            "isPresent": true,
            "devices": [
                "3504175fe0000000000151fd"
            ]
        },
        {
```

```
                "id": 6,
                "name": "red",
                "isPresent": true,
                "devices": [
                    "3504175fe0000000000042dc"
                ]
            },
            {
                "id": 7,
                "name": "green",
                "isPresent": true,
                "devices": [ ]
            },
            {
                "id": 8,
                "name": "black",
                "isPresent": true,
                "devices": [
                    "3504175fe0000000000151fd"
                ]
            },
            {
                "id": 9,
                "name": "white",
                "isPresent": true,
                "devices": [ ]
            },
            {
                "id": 10,
                "name": "display",
                "isPresent": false,
                "devices": [ ]
            }
        ]
    },
    {
        "id": 989,
        "name": "Wohnen",
        "isPresent": true,
        "devices": [
            {
                "id": "3504175fe000000000182f6",
                "name": "Regalleuchte",
                "functionID": 4152,
                "productRevision": 49955,
                "productID": 6344,
                "hwInfo": "GE-SDS200",
                "meterDSID": "3504175fe0000010000003dd",
                "busID": 97,
                "zoneID": 989,
                "isPresent": false,
                "lastDiscovered": "2012-10-24␣11:17:29",
                "firstSeen": "2012-10-22␣16:22:02",
                "inactiveSince": "2012-10-22␣16:22:02",
                "outputMode": 22,
                "buttonID": 0,
                "buttonActiveGroup": 1,
                "buttonInputMode": 0,
                "buttonInputIndex": 0,
                "buttonInputCount": 1,
                "groups": [
                    "1"
```

```
        ]
    },
    {
        "id": "3504175fe00000000000439c",
        "name": "Stehlampe",
        "functionID": 4152,
        "productRevision": 789,
        "productID": 200,
        "hwInfo": "GE-KM200",
        "meterDSID": "3504175fe0000010000003dd",
        "busID": 153,
        "zoneID": 989,
        "isPresent": false,
        "lastDiscovered": "2012-10-24 11:17:29",
        "firstSeen": "2012-10-22 16:22:02",
        "inactiveSince": "2012-10-22 16:22:02",
        "outputMode": 22,
        "buttonID": 0,
        "buttonActiveGroup": 1,
        "buttonInputMode": 0,
        "buttonInputIndex": 0,
        "buttonInputCount": 1,
        "groups": [
            "1"
        ]
    },
    {
        "id": "3504175fe0000000000151fd",
        "name": "Fernseher",
        "functionID": 33041,
        "productRevision": 41761,
        "productID": 5320,
        "hwInfo": "SW-ZWS200",
        "meterDSID": "3504175fe0000010000003dd",
        "busID": 693,
        "zoneID": 989,
        "isPresent": false,
        "lastDiscovered": "2012-10-24 11:17:29",
        "firstSeen": "2012-10-22 16:22:02",
        "inactiveSince": "2012-10-22 16:22:02",
        "outputMode": 39,
        "buttonID": 0,
        "buttonActiveGroup": 5,
        "buttonInputMode": 0,
        "buttonInputIndex": 0,
        "buttonInputCount": 1,
        "groups": [
            "5",
            "8"
        ]
    },
    {
        "id": "3504175fe000000000001234",
        "name": "Wandlampe",
        "functionID": 4144,
        "productRevision": 789,
        "productID": 1234,
        "hwInfo": "GE-TKM210",
        "meterDSID": "3504175fe0000010000003dd",
        "busID": 782,
        "zoneID": 989,
        "isPresent": false,
```

```
            "lastDiscovered": "2012-10-24 11:17:29",
            "firstSeen": "2012-10-22 16:22:02",
            "inactiveSince": "2012-10-22 16:22:02",
            "outputMode": 22,
            "buttonID": 4,
            "buttonActiveGroup": 1,
            "buttonInputMode": 0,
            "buttonInputIndex": 0,
            "buttonInputCount": 1,
            "groups": [
                "1"
            ]
        },
        {
            "id": "3504175fe0000000000042dc",
            "name": "Paniktaster",
            "functionID": 24896,
            "productRevision": 790,
            "productID": 1225,
            "hwInfo": "RT-TKM201",
            "meterDSID": "3504175fe0000010000003dd",
            "busID": 785,
            "zoneID": 989,
            "isPresent": false,
            "lastDiscovered": "2012-10-24 11:17:29",
            "firstSeen": "2012-10-23 16:23:38",
            "inactiveSince": "2012-10-24 11:01:40",
            "outputMode": 0,
            "buttonID": 17,
            "buttonActiveGroup": 154,
            "buttonInputMode": 20,
            "buttonInputIndex": 0,
            "buttonInputCount": 0,
            "groups": [
                "6"
            ]
        }
    ],
    "groups": [
        {
            "id": 0,
            "name": "broadcast",
            "isPresent": false,
            "devices": [
                "3504175fe0000000000182f6",
                "3504175fe00000000000439c",
                "3504175fe0000000000151fd",
                "3504175fe000000000001234",
                "3504175fe0000000000042dc"
            ]
        },
        {
            "id": 1,
            "name": "yellow",
            "isPresent": true,
            "devices": [
                "3504175fe0000000000182f6",
                "3504175fe00000000000439c",
                "3504175fe000000000001234"
            ]
        },
        {
```

```
                "id": 2,
                "name": "gray",
                "isPresent": true,
                "devices": []
            },
            {
                "id": 3,
                "name": "blue",
                "isPresent": true,
                "devices": []
            },
            {
                "id": 4,
                "name": "cyan",
                "isPresent": true,
                "devices": []
            },
            {
                "id": 5,
                "name": "magenta",
                "isPresent": true,
                "devices": [
                    "3504175fe0000000000151fd"
                ]
            },
            {
                "id": 6,
                "name": "red",
                "isPresent": true,
                "devices": [
                    "3504175fe00000000000042dc"
                ]
            },
            {
                "id": 7,
                "name": "green",
                "isPresent": true,
                "devices": []
            },
            {
                "id": 8,
                "name": "black",
                "isPresent": true,
                "devices": [
                    "3504175fe0000000000151fd"
                ]
            },
            {
                "id": 9,
                "name": "white",
                "isPresent": true,
                "devices": []
            },
            {
                "id": 10,
                "name": "display",
                "isPresent": false,
                "devices": []
            }
        ]
    },
    {
```

```
"id": 1038,
"name": "Schlafen",
"isPresent": true,
"devices": [
    {
        "id": "3504175fe0000000000043a7",
        "name": "Deckenlicht",
        "functionID": 4152,
        "productRevision": 789,
        "productID": 200,
        "hwInfo": "GE-KM200",
        "meterDSID": "3504175fe0000010000003dd",
        "busID": 784,
        "zoneID": 1038,
        "isPresent": true,
        "lastDiscovered": "2012-10-26␣15:36:30",
        "firstSeen": "2012-10-22␣16:22:02",
        "inactiveSince": "1970-01-01␣01:00:00",
        "outputMode": 22,
        "buttonID": 5,
        "buttonActiveGroup": 1,
        "buttonInputMode": 0,
        "buttonInputIndex": 0,
        "buttonInputCount": 1,
        "groups": [
            "1"
        ]
    }
],
"groups": [
    {
        "id": 0,
        "name": "broadcast",
        "isPresent": true,
        "devices": [
            "3504175fe0000000000043a7"
        ]
    },
    {
        "id": 1,
        "name": "yellow",
        "isPresent": true,
        "devices": [
            "3504175fe0000000000043a7"
        ]
    },
    {
        "id": 2,
        "name": "gray",
        "isPresent": true,
        "devices": [ ]
    },
    {
        "id": 3,
        "name": "blue",
        "isPresent": true,
        "devices": [ ]
    },
    {
        "id": 4,
        "name": "cyan",
        "isPresent": true,
```

```
                        "devices": [ ]
                    },
                    {
                        "id": 5,
                        "name": "magenta",
                        "isPresent": true,
                        "devices": [ ]
                    },
                    {
                        "id": 6,
                        "name": "red",
                        "isPresent": true,
                        "devices": [ ]
                    },
                    {
                        "id": 7,
                        "name": "green",
                        "isPresent": true,
                        "devices": [ ]
                    },
                    {
                        "id": 8,
                        "name": "black",
                        "isPresent": true,
                        "devices": [ ]
                    },
                    {
                        "id": 9,
                        "name": "white",
                        "isPresent": true,
                        "devices": [ ]
                    },
                    {
                        "id": 10,
                        "name": "display",
                        "isPresent": false,
                        "devices": [ ]
                    }
                ]
            }
        ]
    }
}
```


### 2.5.2 getDevices

Returns an array containing all devices of the apartment.

**Synopsis**
HTTP GET /json/apartment/getDevices

**Parameter**
None

**Response**
HTTP Status 200

| result | array of devices |
|---|---|

**Sample**

GET /json/apartment/getDevices


```
{
    "ok": true,
    "result": [
        {
            "id": "3504175fe0000000000182f6",
            "name": "Regalleuchte",
            "functionID": 4152,
            "productRevision": 49955,
            "productID": 6344,
            "hwInfo": "GE-SDS200",
            "meterDSID": "3504175fe0000010000003dd",
            "busID": 97,
            "zoneID": 989,
            "isPresent": false,
            "lastDiscovered": "2012-10-24 11:17:29",
            "firstSeen": "2012-10-22 16:22:02",
            "inactiveSince": "2012-10-22 16:22:02",
            "outputMode": 22,
            "buttonID": 0,
            "buttonActiveGroup": 1,
            "buttonInputMode": 0,
            "buttonInputIndex": 0,
            "buttonInputCount": 1,
            "groups": [
                "1"
            ]
        },
        {
            "id": "3504175fe00000000000439c",
            "name": "Stehlampe",
            "functionID": 4152,
            "productRevision": 789,
            "productID": 200,
            "hwInfo": "GE-KM200",
            "meterDSID": "3504175fe0000010000003dd",
            "busID": 153,
            "zoneID": 989,
            "isPresent": false,
            "lastDiscovered": "2012-10-24 11:17:29",
            "firstSeen": "2012-10-22 16:22:02",
            "inactiveSince": "2012-10-22 16:22:02",
            "outputMode": 22,
            "buttonID": 0,
            "buttonActiveGroup": 1,
            "buttonInputMode": 0,
            "buttonInputIndex": 0,
            "buttonInputCount": 1,
            "groups": [
                "1"
            ]
```

```
    },
    {
        "id": "3504175fe0000000000151fd",
        "name": "Fernseher",
        "functionID": 33041,
        "productRevision": 41761,
        "productID": 5320,
        "hwInfo": "SW-ZWS200",
        "meterDSID": "3504175fe0000010000003dd",
        "busID": 693,
        "zoneID": 989,
        "isPresent": false,
        "lastDiscovered": "2012-10-24 11:17:29",
        "firstSeen": "2012-10-22 16:22:02",
        "inactiveSince": "2012-10-22 16:22:02",
        "outputMode": 39,
        "buttonID": 0,
        "buttonActiveGroup": 5,
        "buttonInputMode": 0,
        "buttonInputIndex": 0,
        "buttonInputCount": 1,
        "groups": [
            "5",
            "8"
        ]
    },
    {
        "id": "3504175fe000000000001234",
        "name": "Wandlampe",
        "functionID": 4144,
        "productRevision": 789,
        "productID": 1234,
        "hwInfo": "GE-TKM210",
        "meterDSID": "3504175fe0000010000003dd",
        "busID": 782,
        "zoneID": 989,
        "isPresent": false,
        "lastDiscovered": "2012-10-24 11:17:29",
        "firstSeen": "2012-10-22 16:22:02",
        "inactiveSince": "2012-10-22 16:22:02",
        "outputMode": 22,
        "buttonID": 4,
        "buttonActiveGroup": 1,
        "buttonInputMode": 0,
        "buttonInputIndex": 0,
        "buttonInputCount": 1,
        "groups": [
            "1"
        ]
    },
    {
        "id": "3504175fe0000000000043a7",
        "name": "Deckenlicht",
        "functionID": 4152,
        "productRevision": 789,
        "productID": 200,
        "hwInfo": "GE-KM200",
        "meterDSID": "3504175fe0000010000003dd",
        "busID": 784,
        "zoneID": 1038,
        "isPresent": true,
        "lastDiscovered": "2012-10-26 15:36:30",
```

```
            "firstSeen": "2012-10-22 16:22:02",
            "inactiveSince": "1970-01-01 01:00:00",
            "outputMode": 22,
            "buttonID": 5,
            "buttonActiveGroup": 1,
            "buttonInputMode": 0,
            "buttonInputIndex": 0,
            "buttonInputCount": 1,
            "groups": [
                "1"
            ]
        },
        {
            "id": "3504175fe0000000000042dc",
            "name": "Paniktaster",
            "functionID": 24896,
            "productRevision": 790,
            "productID": 1225,
            "hwInfo": "RT-TKM201",
            "meterDSID": "3504175fe0000010000003dd",
            "busID": 785,
            "zoneID": 989,
            "isPresent": false,
            "lastDiscovered": "2012-10-24 11:17:29",
            "firstSeen": "2012-10-23 16:23:38",
            "inactiveSince": "2012-10-24 11:01:40",
            "outputMode": 0,
            "buttonID": 17,
            "buttonActiveGroup": 154,
            "buttonInputMode": 20,
            "buttonInputIndex": 0,
            "buttonInputCount": 0,
            "groups": [
                "6"
            ]
        }
    ]
}
```


### 2.5.3 getCircuits

Returns an array containing all digitalSTROM Meters of the apartment.

**Synopsis**
HTTP GET /json/apartment/getCircuits

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| result.circuits | array of digitalSTROM Meters |

**Sample**


```
GET /json/apartment/getCircuits
{

    "ok": true,
    "result": {
        "circuits": [
            {
                "name": "dSM03DD-#1",
                "dsid": "3504175fe0000010000003dd",
                "hwVersion": 721409,
                "armSwVersion": 17498112,
                "dspSwVersion": 16908800,
                "apiVersion": 517,
                "hwName": "",
                "isPresent": true,
                "isValid": true
            },
            {
                "name": "dSM040E-#2",
                "dsid": "3504175fe00000100000040e",
                "hwVersion": 721409,
                "armSwVersion": 17498112,
                "dspSwVersion": 16908800,
                "apiVersion": 517,
                "hwName": "",
                "isPresent": true,
                "isValid": true
            }
        ]
    }

}
```


### 2.5.4 removeMeter

Removes an inactive digitalSTROM Meter object from the installation.

**Synopsis**
HTTP GET /json/apartment/removeMeter

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| dsid | dSID of the digitalSTROM Meter | Mandatory |

**Response**
HTTP Status 200

| result | array of digitalSTROM Meters |
|---|---|

**Sample**

```
GET /json/apartment/removeMeter?dsid=3504175fe00000100000040e
{
    "ok" : true
}
```

## 2.6 Heating

### 2.6.1 Get Temperature Control Status

Get the current status of temperature control in all zones.

**Synopsis**
HTTP GET /json/apartment/getTemperatureControlStatus

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| id | Id of the zone |
| name | Name of the zone |
| ControlMode | Control mode: 0=off; 1=pid-control; 2=zone-follower; 3=fixed-value; 4=manual |
| ControlState | Control state: 0=internal; 1=external; 2=exbackup; 3=emergency |
| ControlDSUID | dSUID of the meter or service that runs the control algorithm for this zone |
| OperationMode | Current operation mode of the control |
| Temperature | Current temperature of the zone |
| TemperatureTime | Timestamp of last temperature data update, seconds since epoch |
| NominalValue | Target temperature of this zone |
| NominalValueTime | Timestamp of last set point change, seconds since epoch |
| ControlValue | Current control value |
| ControlValueTime | Timestamp of last control value data update, seconds since epoch |
| IsConfigured | A boolean value indicating if a heating controller is setup |

**Sample**

```
GET /json/apartment/getTemperatureControlStatus
{
    "ok": true,
    "result":
        "zones": [
            {
                "id": 1,
                "name": "Living␣Room",
                "IsConfigured": true,
                "ControlMode": 1,
                "ControlState": 3,
                "ControlDSUID": 3504175fe0000000001000000006239100,
                "OperationMode": 4,
                "Temperature": 20.7,
                "NominalValue": 20.0,
                "ControlValue": 92.5,
                "TemperatureTime": 2014-10-08T18:21:05Z,
                "NominalValueTime": 2014-10-08T18:00:00Z,
                "ControlValueTime": 2014-10-08T18:22:00Z
            },
            {
                "id": 2,
                "name": "Kitchen",
                "IsConfigured": true,
                "ControlMode": 2,
                "ControlState": 1,
                "ControlDSUID": 3504175fe0000000001000000006239200,
                "ControlValue": 92.5,
                "ControlValueTime": 2014-10-08T18:19:00Z
            },
            {
                "id": 3,
                "name": "Corridor",
                "IsConfigured": true,
                "ControlMode": 3,
                "ControlState": 3,
                "ControlDSUID": 3504175fe0000000001000000006239300,
                "OperationMode": 2,
                "ControlValue": 80
            },
            {
                "id": 4,
                "name": "Garden",
                "IsConfigured": false
            }
        ]
    }
}
```

### 2.6.2 Get Temperature Control Configuration

Get the configuration of the temperature control settings for all zones.

**Synopsis**
HTTP GET /json/apartment/getTemperatureControlConfig

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| id | Id of this zone |
| name | Name of this zone |
| ControlDSUID | dSUID of the meter or service that runs the control algorithm for this zone |
| ControlMode | Control mode: 0=off; 1=pid-control; 2=zone-follower; 3=fixed-offset; 4=manual |
| ReferenceZone | Zone number of the reference zone (mode 2 only) |
| CtrlOffset | Static control value offset (mode 2 only) |
| EmergencyValue | Fixed control value in case of malfunction (mode 1 only) |
| CtrlKp | Control proportional factor |
| CtrlTs | Control sampling time |
| CtrlTi | Control integrator time constant |
| CtrlKd | Control differential factor |
| CtrlImin | Control minimum integrator value |
| CtrlImax | Control maximum integrator value |
| CtrlYmin | Control minimum control value |
| CtrlYmax | Control maximum control value |
| CtrlAntiWindUp | Control integrator anti wind up: 0=inactive, 1=active |
| CtrlKeepFloorWarm | Control mode with higher priority on comfort: 0=inactive, 1=active |
| IsConfigured | A boolean value indicating if a heating controller is setup |

**Sample**

```
GET /json/apartment/getTemperatureControlConfig
{
    "ok": true,
    "result":
        "zones": [
            {
                "id": 1,
                "name": "Living␣Room",
                "IsConfigured": true,
                "ControlDSUID": 3504175fe0000000001000000006239100,
                "ControlMode": 1,
                "EmergencyValue": 50,
                "CtrlKp": 5.2,
                "CtrlTs": 240,
                "CtrlTi": 1,
                "CtrlKd": 1,
```

```
                "CtrlImin": 600,
                "CtrlImax": 2400,
                "CtrlYmin": 0,
                "CtrlYmax": 100,
                "CtrlAntiWindUp": 1,
                "CtrlKeepFloorWarm": 0
            },
            {
                "id": 2,
                "name": "Kitchen",
                "IsConfigured": true,
                "ControlDSUID": 3504175fe0000000001000000006239200,
                "ControlMode": 2,
                "ReferenceZone": 0,
                "CtrlOffset": -10
            },
            {
                "id": 3,
                "name": "Corridor",
                "IsConfigured": true,
                "ControlDSUID": 3504175fe0000000001000000006239300,
                "ControlMode": 3
            },
            {
                "id": 4,
                "name": "Garden",
                "IsConfigured": false
            }
        ]
    }
}
```


### 2.6.3 Get Temperature Control Values

Returns a list of all temperature control preset values of all zones. Every control operation mode has up to 15 presets defined, where 6 of them are actually used by the system.

**Synopsis**
HTTP GET /json/apartment/getTemperatureControlValues

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| id | Id of this zone |
| name | Name of this zone |
| IsConfigured | A boolean value indicating if a heating controller is setup for this zone |
| ControlDSUID | dSUID of the meter or service that runs the control algorithm for this zone and stores the preset values |
| Off | Preset value for operation mode 0: "Off" |
| Comfort | Preset value for operation mode 1: "Comfort" |
| Economy | Preset value for operation mode 2: "Economy" |
| NotUsed | Preset value for operation mode 3: "Not Used" |
| Night | Preset value for operation mode 4: "Night" |
| Holiday | Preset value for operation mode 5: "Holiday" |

**Sample**

```
GET /json/apartment/getTemperatureControlValues
{
    "ok": true,
    "result":
        "zones": [
            {
                "id": 1,
                "name": "Living␣Room",
                "IsConfigured": true,
                "ControlDSUID": 3504175fe0000000001000000006239200,
                "Off": 6,
                "Comfort": 21,
                "Economy": 20,
                "NotUsed": 18,
                "Night": 16,
                "Holiday": 12
            },
            {
                "id": 972,
                "name": "",
                "IsConfigured": true,
                "ControlDSUID": 3504175fe0000000001000000006239200,
                "Off": 8,
                "Comfort": 22,
                "Economy": 20,
                "NotUsed": 18,
                "Night": 17,
                "Holiday": 16
            },
            {
                "id": 4,
                "name": "Garden",
                "IsConfigured": false
            }
    }
}
```

### 2.6.4 Get Assigned Sensors

Returns the list of assigned sensor devices in all zones.

**Synopsis**
HTTP GET /json/apartment/getAssignedSensors

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| id | Id of this zone |
| name | Name of this zone |
| sensorType | Numerical value of the sensor type |
| dsuid | dSUID of the source device |

**Sample**

```
GET /json/apartment/getAssignedSensors
{
    "ok":true,
    "result":
        "zones": [
            {
                "id": 1,
                "name": "Living␣Room",
                "sensors": [
                    {
                        "sensorType": 9,
                        "dsuid": 3504175fe00000000000000000016be700
                    },
                    {
                        "sensorType": 11,
                        "dsuid": 3504175fe00000000000000000016be700
                    }
                ]
            },
            {
                "id": 2,
                "name": "Kitchen",
                "sensors": [
                    {
                        "sensorType": 9,
                        "dsuid": 3504175fe00000000000000000001456700
                    }
                ]
            }
        ]
    }
}
```

### 2.6.5 Get Sensor Values

Returns a list of sensor relevant for the apartment.

For the apartment the temperature, humidity, and brightness are sensor types that are tracked. Additionally there is

For each zone the temperature, humidity, CO2 concentration and brightness are sensor types that are tracked. Typically there is one device as a zone reference for these values.

If there is no standard device defined for a sensor type or if no measurement is available there is neither the value or time field returned.

**Synopsis**
HTTP GET /json/apartment/getSensorValues

**Parameter**
None

**Response**
HTTP Status 200

The result object contains the following outdoor measurements:

| | |
|---|---|
| TemperatureValue | Outdoor temperature value |
| TemperatureValueTime | Timestamp of the temperature measurement |
| HumidityValue | Outdoor humidity value |
| HumidityValueTime | Timestamp of the humidity measurement |
| BrightnessValue | Outdoor brightness value |
| BrightnessValueTime | Timestamp of the brightness measurement |

If there is external weather service data available it will be provided as well:

| | |
|---|---|
| WeatherIconId | |
| WeatherConditionId | |
| WeatherServiceId | |
| WeatherServiceTime | |

The result object contains a ”zones” field with an array of all zones of the apartment and the relevant sensor data:

| | |
|---|---|
| TemperatureValue | Temperature value |
| TemperatureValueTime | Timestamp of the temperature measurement |
| HumidityValue | Humidity value |
| HumidityValueTime | Timestamp of the humidity measurement |
| CO2ConcentrationValue | CO2Concentration value |
| CO2ConcentrationValueTime | Timestamp of the CO2 concentration measurement |
| BrightnessValue | Brightness value |
| BrightnessValueTime | Timestamp of the brightness measurement |

### Sample

GET /json/apartment/getSensorValues


```
{

    "ok": true,
    "result": {
        "TemperatureValue": 17.0,
        "TemperatureValueTime": "2014-10-13T18:05:05.842+0200",
        "WeatherIconId": "",
        "WeatherConditionId": "",
        "WeatherServiceId": "",
        "WeatherServiceTime": "2014-10-12T16:42:31.106+0200",
        "zones": [
            {
                "id": 1142,
                "name": "Küche",
                "values": [ ]
            },
            {
                "id": 1168,
                "name": "Wohnzimmer",
                "values": [
                    {
                        "TemperatureValue": 22.55,
                        "TemperatureValueTime": "2014-10-13T18:07:24.528+0200"
                    },
                    {
                        "HumidityValue": 59.2,
                        "HumidityValueTime": "2014-10-13T18:07:24.638+0200"
                    },
                    {
                        "CO2ConcentrationValue": 1209.205182943208,
                        "CO2ConcentrationValueTime": "2014-10-13T11:45:53.756+0200"
                    }
                ]
            },
            {
                "id": 1191,
                "name": "Galerie",
                "values": [
                    {
                        "TemperatureValue": 21.75,
                        "TemperatureValueTime": "2014-10-13T18:04:13.382+0200"
                    },
                    {
                        "HumidityValue": 64.4,
                        "HumidityValueTime": "2014-10-13T18:04:13.480+0200"
                    }
                ]
            },
            {
                "id": 1192,
                "name": "Flur",
                "values": [
                    {
                        "TemperatureValue": 21.95000000000005,
                        "TemperatureValueTime": "2014-10-13T18:04:10.022+0200"
                    }
                ]
            },
```

```
            {
                "id": 3,
                "name": "Wintergarten",
                "values": [
                    {
                        "TemperatureValue": 19.67500000000001,
                        "TemperatureValueTime": "2014-10-13T18:06:07.659+0200"
                    },
                    {
                        "HumidityValue": 66,
                        "HumidityValueTime": "2014-10-13T18:06:07.790+0200"
                    }
                ]
            },
            {
                "id": 10,
                "name": "Terrasse",
                "values": []
            }
        ]
    }

}
```

# 3 Zone

## 3.1 Common

Every /json/zone/ function uses a common selection scheme for the zone to which the command refers to. Either the parameter "id" or "name" must be given to identify the zone. The special value zero for the "id" maybe used to send the command as broadcast to all zones.

| Parameter | Description | Remarks |
|---|---|---|
| id | Zone Number | Optional |
| name | Zone Name | Optional |

A missing zone identifier result in the following error message to be returned.

```
{
   "ok": false,
   "message": "Need␣parameter␣name␣or␣id␣to␣identify␣zone"

}
```

If a zone identifier does not match any actually known zone in the installation the following error message is returned.

```
{
    "ok": false,
    "message": "Could␣not␣find␣zone␣with␣id␣'1250'"
}
```

## 3.2 Name

### 3.2.1 getName

Returns the user defined name of the zone.

**Synopsis**
HTTP GET /json/zone/getName

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| name | identifier string for the zone |

**Sample**

```
GET /json/zone/getName?id=1237
{
    "ok":true,
    "result" :
    {
        "name" : "Wohnen"
    }
}
```

### 3.2.2 setName

Sets the zone name.

**Synopsis**
HTTP GET /json/zone/setName

| Parameter | Description | Remarks |
|---|---|---|
| newName | identifier string for the zone | Mandatory |

**Parameter**

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/zone/setName?id=1237&newName="Wohnen"
{
    "ok":true
}
```

## 3.3 Scene

### 3.3.1 callScene

Excutes the scene *sceneNumber* in a zone for a group of devices.

**Synopsis**
HTTP GET /json/zone/callScene

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneNumber | Numerical value | Mandatory |
| groupID | Number of the target group | Optional |
| groupName | Name of the target group | Optional |
| force | Boolean value, if set a forced scene call is issued | Optional |

If the group parameters are omitted the command is sent as broadcast to all devices in a zone.

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/zone/callScene?id=1237&groupID=1&sceneNumber=5&force=true
{
    "ok":true
}
```

### 3.3.2 saveScene

Tells devices to store their current output values as a default for the scene *sceneNumber*.

**Synopsis**
HTTP GET /json/zone/saveScene

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneNumber | Numerical value | Mandatory |
| groupID | Number of the target group | Optional |
| groupName | Name of the target group | Optional |

If the group parameters are omitted the command is sent as broadcast to all devices in a zone.

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/zone/saveScene?id=1237&groupID=2&sceneNumber=17
{
    "ok":true
}
```

### 3.3.3 undoScene

Tells devices to restore their output values to the previous state if the current scene matches the *sceneNumber*.

**Synopsis**
HTTP GET /json/zone/undoScene

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneNumber | Numerical value | Mandatory |
| groupID | Number of the target group | Optional |
| groupName | Name of the target group | Optional |

If the group parameters are omitted the command is sent as broadcast to all devices in the zone.

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/zone/undoScene?id=1237&sceneNumber=65
{
    "ok":true
}
```

### 3.3.4 sceneGetName

Get the user defined name for a scene *sceneNumber* within a group of a zone.

**Synopsis**
HTTP GET /json/zone/sceneGetName

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneNumber | Numerical value | Mandatory |
| groupID | Number of the target group | M/O |
| groupName | Name of the target group | M/O |

Either groupID or groupName must be supplied to this request.

**Response**
HTTP Status 200

| | |
|---|---|
| result.name | the user defined name of the scene |

**Sample**

```
GET /json/zone/sceneGetName?id=1237&sceneNumber=19&groupID=1
{
    "ok":true
    result":␣{
␣␣␣␣␣␣␣"name":␣"Fernsehen"
␣␣␣}
}
```

### 3.3.5 sceneSetName

Sets a user defined name for a scene *sceneNumber* within a group of a zone. This name is stored on the digitalSTROM Server only.

**Synopsis**
HTTP GET /json/zone/sceneSetName

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| newName | User defined name of the scene | Mandatory |
| sceneNumber | Numerical value | Mandatory |
| groupID | Number of the target group | M/O |
| groupName | Name of the target group | M/O |

Either groupID or groupName must be supplied to this request.

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/zone/sceneSetName?id=1237&sceneNumber=17&groupID=2&newName="Fernsehen"
{
    "ok":true
}
```

### 3.3.6 getReachableScenes

Returns a list of groups which can be controlled by pushbuttons which are actually present in the zone.

**Synopsis**
HTTP GET /json/zone/getReachableScenes

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| groupID | Number of the target group | Optional |
| groupName | Name of the target group | Optional |

If groupID or groupName are omitted the combined list for all groups is returned.

**Response**
HTTP Status 200

| result.reachableScens | array of scene numbers |
|---|---|

**Sample**

```
GET /json/zone/getReachableScenes?id=1237&groupID=1
{
    "ok": true,
    "result": {
        "reachableScenes": [
            0,
            1,
            5,
            6,
            17,
            18,
            19,
            29,
            30,
            31,
            38,
            39
        ]
    }
}
```

### 3.3.7 getLastCalledScene

Returns the *sceneNumber* which has been executed last for a group in a zone.

**Synopsis**
HTTP GET /json/zone/getLastCalledScene

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| groupID | Number of the target group | Optional |
| groupName | Name of the target group | Optional |

**Response**
HTTP Status 200

| | |
|---|---|
| result.scene | the number of the last called scene |

**Sample**

```
GET /json/zone/getLastCalledScene?id=1237&groupID=1
{
    "ok": true,
    "result": {
        "scene": 0
```

```
    }
}
```

## 3.4 Value

### 3.4.1 Set Output Value

Set the output value of a group of devices in a zone to a given value.

> **Notice** Setting output values directly bypasses the group state machine and is not recommended.

**Synopsis**
HTTP GET /json/zone/setValue

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| value | Numerical value | Mandatory |
| groupID | Number of the target group | Optional |
| groupName | Name of the target group | Optional |

If the group parameters are omitted the command is sent as broadcast to all devices in the selected zone.

> **Notice** Setting output values without a group identification is strongly unrecommended.

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/zone/setValue?id=1237&value=0&groupID=2
{
    "ok":true,
}
```

### 3.4.2 Blink

Executes the "blink" function on a group of devices in a zone for identification purposes.

**Synopsis**
HTTP GET /json/zone/blink

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| groupID | Number of the target group | Optional |
| groupName | Name of the target group | Optional |

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/zone/blink?id=1237&groupID=1
{
    "ok":true,
}
```

### 3.4.3 Send Sensor Value

Send a sensor value to a group of devices in a zone.

**Synopsis**
HTTP GET /json/zone/pushSensorValue

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| groupID | Number of the target group | Optional |
| sourceDSUID | DSUID of the originating device | Optional |
| sensorValue | Numerical value | Mandatory |
| sensorType | Numerical type of the sensor | Mandantory |

If the group parameter is omitted the command is sent as broadcast to all devices in the selected zone. The reference for the sensor type definitions can be found in the ds-basics document.

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/zone/pushSensorValue?id=1237&sensorType=51&sensorValue=100&groupID=48
{
    "ok":true,
}
```

## 3.5 Heating

### 3.5.1 Get Temperature Control Status

Get the current status of the zone temperature control.

**Synopsis**
HTTP GET /json/zone/getTemperatureControlStatus

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| ControlMode | Control mode: 0=off; 1=pid-control; 2=zone-follower; 3=fixed-value; 4=manual |
| ControlState | Control state: 0=internal; 1=external; 2=exbackup; 3=emergency |
| ControlDSUID | dSUID of the meter or service that runs the control algorithm for this zone, can be zero |
| OperationMode | Current operation mode of the control |
| Temperature | Current temperature of the zone |
| TemperatureTime | Timestamp of last temperature data update, seconds since epoch |
| NominalValue | Target temperature of this zone |
| NominalValueTime | Timestamp of last set point change, seconds since epoch |
| ControlValue | Current control value |
| ControlValueTime | Timestamp of last control value data update, seconds since epoch |
| IsConfigured | A boolean value indicating if a heating controller is setup |

**Sample**

```
GET /json/zone/getTemperatureControlStatus?id=1237
{
    "ok": true,
    "result":
    {
        "IsConfigured": true,
        "ControlMode": 1,
        "ControlState": 2,
        "ControlDSUID": 3504175fe000000000100000000623 9100,
        "OperationMode": 4,
        "Temperature": 20.7,
        "NominalValue": 20.0,
        "ControlValue": 92.5,
        "TemperatureTime": 2014-10-08T18:21:05Z,
        "NominalValueTime": 2014-10-08T18:00:00Z,
        "ControlValueTime": 2014-10-08T18:22:00Z
    }
}
```

### 3.5.2 Get Temperature Control Configuration

Get the configuration of the zone temperature control.

**Synopsis**
HTTP GET /json/zone/getTemperatureControlConfig

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| ControlDSUID | dSUID of the meter or service that runs the control algorithm for this zone, can be zero |
| ControlMode | Control mode: 0=off; 1=pid-control; 2=zone-follower; 3=fixed-value; 4=manual |
| ReferenceZone | Zone number of the reference zone (mode 2 only), can be zero |
| CtrlOffset | Control value offset (mode 2 only) |
| ManualValue | Fixed control value for manual mode (mode 4 only) |
| EmergencyValue | Fixed control value in case of malfunction (mode 1 only) |
| CtrlKp | Control proportional factor |
| CtrlTs | Control sampling time |
| CtrlTi | Control integrator time constant |
| CtrlKd | Control differential factor |
| CtrlImin | Control minimum integrator value |
| CtrlImax | Control maximum integrator value |
| CtrlYmin | Control minimum control value |
| CtrlYmax | Control maximum control value |
| CtrlAntiWindUp | Control integrator anti wind up: 0=inactive, 1=active |
| CtrlKeepFloorWarm | Control mode with higher priority on comfort: 0=inactive, 1=active |
| IsConfigured | A boolean value indicating if a heating controller is setup |

**Sample**

```
GET /json/zone/getTemperatureControlConfig?id=1237
{
    "ok": true,
    "result":
    {
        "IsConfigured": true,
        "ControlDSUID": 3504175fe000000000100000000623910 0,
        "ControlMode": 1,
        "EmergencyValue": 50,
        "CtrlKp": 5.2,
        "CtrlTs": 240,
        "CtrlTi": 1,
        "CtrlKd": 1,
        "CtrlImin": 600,
        "CtrlImax": 2400,
        "CtrlYmin": 0,
```

```
        "CtrlYmax": 100,
        "CtrlAntiWindUp": 1,
        "CtrlKeepFloorWarm": 0
    }
}
```

```
GET /json/zone/getTemperatureControlConfig?id=1237
{
    "ok": true,
    "result":
    {
        "IsConfigured": true,
        "ControlDSUID": 3504175fe0000000001000000006239100,
        "ControlMode": 2,
        "ReferenceZone": 0,
        "CtrlOffset": 10
    }
}
```

```
GET /json/zone/getTemperatureControlConfig?id=1237
{
    "ok": true,
    "result":
    {
        "IsConfigured": true,
        "ControlDSUID": 3504175fe0000000001000000006239100,
        "ControlMode": 3
    }
}
```

```
GET /json/zone/getTemperatureControlConfig?id=1237
{
    "ok": true,
    "result":
    {
        "IsConfigured": true,
        "ControlDSUID": 3504175fe0000000001000000006239100,
        "ManualValue": 87.5,
        "ControlMode": 4
    }
}
```


### 3.5.3 Set Temperature Control Configuration

Set the configuration of the zone temperature control.

**Synopsis**
HTTP GET /json/zone/setTemperatureControlConfig

**Parameter**

| | | |
|---|---|---|
| ControlDSUID | dSUID of the meter or service that runs the control algorithm for this zone | Optional |
| ControlMode | Control mode, can be one of: 0=off; 1=pid-control; 2=zone-follower; 3=fixed-value; 4=manual | Optional |
| ReferenceZone | Zone number of the reference zone | Optional for ControlMode 2 |
| CtrlOffset | Control value offset | Optional for ControlMode 2 |
| EmergencyValue | Fixed control value in case of malfunction | Optional for ControlMode 1 |
| ManualValue | Control value for manual mode | Optional for ControlMode 4 |
| CtrlKp | Control proportional factor | Mandatory for ControlMode 1 |
| CtrlTs | Control sampling time | Mandatory for ControlMode 1 |
| CtrlTi | Control integrator time constant | Mandatory for ControlMode 1 |
| CtrlKd | Control differential factor | Mandatory for ControlMode 1 |
| CtrlImin | Control minimum integrator value | Mandatory for ControlMode 1 |
| CtrlImax | Control maximum integrator value | Mandatory for ControlMode 1 |
| CtrlYmin | Control minimum control value | Optional for ControlMode 1 |
| CtrlYmax | Control maximum control value | Optional for ControlMode 1 |
| CtrlAntiWindUp | Control integrator anti wind up | Mandatory for ControlMode 1 |
| CtrlKeepFloorWarm | Control mode with higher priority on comfort | Mandatory for ControlMode 1 |

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/zone/setTemperatureControlConfig?id=1237&ControlMode=2&ReferenceZone=4327&
      CtrlOffset=-20
{
    "ok": true,
}
```

### 3.5.4 Get Temperature Control Values

Get the temperature control operation mode preset values for a zone. Every control operation mode has up to 15 presets defined.

**Synopsis**
HTTP GET /json/zone/getTemperatureControlValues

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| IsConfigured | A boolean value indicating if a heating controller is setup |
| Off | Preset value for operation mode 0: "Off" |
| Comfort | Preset value for operation mode 1: "Comfort" |
| Economy | Preset value for operation mode 2: "Economy" |
| NotUsed | Preset value for operation mode 3: "Not Used" |
| Night | Preset value for operation mode 4: "Night" |
| Holiday | Preset value for operation mode 5: "Holiday" |

**Sample**

```
GET /json/zone/getTemperatureControlValues?id=1237
{
    "ok": true,
    "result":
    {
        "Off": 22.5,
        "Comfort": 21,
        "Economy": 20,
        "NotUsed": 18,
        "Night": 16,
        "Holiday": 4
    }
}
```

### 3.5.5 Set Temperature Control Values

Set the temperature control operation mode preset values for a zone. Single values can be given and the others that do not change may be omitted.

**Notice** For operation mode "PID Control" the given values are nominal temperatures, and for operation mode "Fixed Values" the given values are absolute control values.

**Synopsis**
HTTP GET /json/zone/setTemperatureControlValues

**Parameter**

| | | |
|---|---|---|
| Off | Preset value for operation mode 0: "Off" | Optional |
| Comfort | Preset value for operation mode 1: "Comfort" | Optional |
| Economy | Preset value for operation mode 2: "Economy" | Optional |
| NotUsed | Preset value for operation mode 3: "Not Used" | Optional |
| Night | Preset value for operation mode 4: "Night" | Optional |
| Holiday | Preset value for operation mode 5: "Holiday" | Optional |

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/zone/setTemperatureControlValues?id=1237&Comfort=22.5&Night=21
{
    "ok": true
}
```

### 3.5.6 Set Sensor Source

Set the source of a sensor in a zone to a given device source address. For example one might have multiple temperature and humidity sensors in a a room and using this method he can select which one to use for room temperature control.

**Synopsis**
HTTP GET /json/zone/setSensorSource

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| dsid | dSID of the source device | Mandatory |
| sensorType | Numerical value of the sensor type | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/zone/setSensorSource?id=1237&sensorType=9&
      dsuid=3504175fe0000000000000000016be700
{
    "ok": true,
}
```

### 3.5.7 Clear Sensor Source

Remove all assignments for a particular sensor type in a zone.

**Synopsis**
HTTP GET /json/zone/clearSensorSource

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sensorType | Numerical value of the sensor type | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/zone/clearSensorSource?id=1237&sensorType=11
{
    "ok": true,
}
```

### 3.5.8 Get Assigned Sensors

Returns the list of assigned sensor devices in a zone.

**Synopsis**
HTTP GET /json/zone/getAssignedSensors

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| sensorType | Numerical value of the sensor type |
| dsid | dSID of the source device |

**Sample**

```
GET /json/zone/getAssignedSensors?id=1237
{
    "ok":true,
    "result": {
        "sensors": [
            {
                "sensorType": 9,
                "dsuid": 3504175fe00000000000000000016be700
            },
            {
                "sensorType": 11,
                "dsuid": 3504175fe00000000000000000016be700
            }
        ]
    }
}
```

### 3.5.9 Set Temperature Control State

Modify the internal state of the temperature control for a zone.

**Synopsis**
HTTP GET /json/zone/setTemperatureControlState

**Parameter**

| | | |
|---|---|---|
| ControlState | Control state: 0=internal; 1=external; 2=exbackup; 3=emergency | Mandatory |

| ok | true |
|---|---|

**Response**
HTTP Status 200

**Sample**

```
GET /json/zone/setTemperatureControlState?id=1237&ControlState=external
{
    "ok": true
}
```

### 3.5.10 Get Temperature Control Internals

Returns status information of the temperature control of a zone.

**Synopsis**
HTTP GET /json/zone/getTemperatureControlInternals

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| ControlDSUID | dSUID of the meter or service that runs the control algorithm for this zone, can be zero |
| ControlMode | Control mode: 0=off; 1=pid-control; 2=zone-follower; 3=fixed-value; 4=manual |
| ControlState | Control state: 0=internal; 1=external; 2=exbackup; 3=emergency |
| CtrlTRecent | Current room temperature |
| CtrlTReference | Control temperature |
| CtrlTError | Control temperature error, in 0.025K |
| CtrlTErrorPrev | Previous control temperature error, in 0.025K |
| CtrlIntegral | Control current integral value |
| CtrlYp | Current control value proportional portion |
| CtrlYi | Current control value integral portion |
| CtrlYd | Current control value differential portion |
| CtrlY | Current control value |
| CtrlAntiWindUp | Currently the anti wind up condition is active |
| IsConfigured | A boolean value indicating if a heating controller is setup |

**Sample**


```
GET /json/zone/getTemperatureControlInternals?id=1237
{
    "ok":true,
    "result":
    {
        "IsConfigured": true,
        "ControlDSUID": 3504175fe000000000100000000623910 0,
        "ControlMode": 1,
        "ControlState": 0,
        "CtrlTRecent": 20.50,
        "CtrlTReference": 21.00,
        "CtrlTError": 0.55,
        "CtrlTErrorPrev": 0.50,
        "CtrlIntegral": 82,
        "CtrlYp": 3.55,
        "CtrlYi": 23.1,
        "CtrlYd": 0,
        "CtrlY": 27,
        "CtrlAntiWindUp": 0
    }
}
```


### 3.5.11 Get Sensor Values

Returns a list of sensor measurements relevant for a zone. The temperature, humidity, CO2 concentration and brightness are sensor types that are tracked for a zone. Typically there is one device as a zone reference for these values.

If there is no standard device defined for a sensor type or if no measurement is available there is neither the value or time field returned.

**Synopsis**
HTTP GET /json/zone/getSensorValues

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| TemperatureValue | Temperature value |
| TemperatureValueTime | Timestamp of the temperature measurement |
| HumidityValue | Humidity value |
| HumidityValueTime | Timestamp of the humidity measurement |
| CO2ConcentrationValue | CO2Concentration value |
| CO2ConcentrationValueTime | Timestamp of the CO2 concentration measurement |
| BrightnessValue | Brightness value |
| BrightnessValueTime | Timestamp of the brightness measurement |

### Sample

```
GET /json/zone/getSensorValues?id=1237
{

    "ok": true,
    "result": {
        "id": 14236,
        "name": "Heizungsraum",
        "values": [
            {
                "TemperatureValue": 26.5,
                "TemperatureValueTime": "2014-10-13T17:57:21.445+0200"
            }
        ]
    }

}
```

# 4 Device

## 4.1 Common

Every /json/device/ function uses a common selection scheme for the device to which the command refers to. Either the parameter "dsid" or "name" must be given to identify the device.

| Parameter | Description | Remarks |
|---|---|---|
| dsid | Device dSID String | Optional |
| name | Device Name | Optional |
| category | Request Category | Optional |

A missing device identifier result in the following error message to be returned.

```
{ "ok": false, "message": "Need␣parameter␣name␣or␣dsid␣to␣identify␣device" }
```

If a device identifier does not match any actually known device in the installation the following error message is returned.

```
{ "ok": false, "message": "Could␣not␣find␣device␣named␣'Wandlampe␣am␣Eingang'" }
```

The category parameter has an influence on how particular requests are treated, the goal is to prevent scene calls from automated scripts in certain situations. Currently supported categories are:

- manual
- timer
- algoirthm

A missing category parameter is currently treated as manual category, this compatibility will be removed in release 1.8.

## 4.2 Name

### 4.2.1 getName

Returns the user defined name of a device.

**Synopsis**
HTTP GET /json/device/getName

**Parameter**
None

**Response**
HTTP Status 200

| result.name | identifier string for the device |
|---|---|

**Sample**

```
GET /json/device/getName?dsid=3504175fe000000000017ef3
{
    "ok":true,
    "result" :
    {
        "name" : "App-Taster"
    }
}
```

### 4.2.2 setName

Sets the device name.

**Synopsis**
HTTP GET /json/device/setName

| Parameter | Description | Remarks |
|---|---|---|
| newName | identifier string for the device | Mandatory |

**Parameter**

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setName?id=3504175fe000000000017ef3&newName="Wohnen"
{
    "ok":true
}
```

### 4.2.3 getSpec

Retrieves device and product information.

**Synopsis**
HTTP GET /json/device/getSpec

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| result.functionID | Function ID of the device |
| result.productID | Product ID of the device |
| result.revisionID | Revision ID of the device |

**Sample**

```
GET /json/device/getName?dsid=3504175fe000000000017ef3
{
    "ok": true,
    "result": {
        "functionID": 33027,
        "productID": 1224,
        "revisionID": 834
    }

}
```

## 4.3 First seen

### 4.3.1 getFirstSeen

Returns the timestamp where the device was registered.

**Synopsis**
HTTP GET /json/device/getFirstSeen

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| result.time | ISO8601 time where device was registered |

**Sample**

```
GET /json/device/getFirstSeen?dsid=3504175fe000000000017ef3
{
    "result" :
    {
        "time" : "2010-10-01T10:42:13Z"
    },
    "ok":true
}
```

### 4.3.2 setFirstSeen

Sets the timestamp where the device was registered.

**Synopsis**
HTTP GET /json/device/setFirstSeen

| Parameter | Description | Remarks |
|---|---|---|
| time | ISO8601 time where device was registered | Mandatory |

**Parameter**

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setFirstSeen?id=3504175fe000000000017ef3&time=2010-06-14T10:42:13Z
{
    "ok":true
}
```

## 4.4 Groups

### 4.4.1 getGroups

Returns a list of groups the device is registered in.

**Synopsis**
HTTP GET /json/device/getGroups

**Parameter**
None

**Response**
HTTP Status 200

| result.groups | array of groups of the device |
|---|---|

**Sample**

```
GET /json/device/getGroups
{
    "ok": true,
    "result": {
        "groups": [
            {
                "id": 3,
                "name": "blue"
            },
            {
                "id": 8,
                "name": "black"
            }
        ]
    }
}
```

## 4.5 Scene

### 4.5.1 callScene

Excutes the scene *sceneNumber* on a devices.

**Synopsis**
HTTP GET /json/device/callScene

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneNumber | Numerical value | Mandatory |
| force | Boolean value, if set a forced scene call is issued | Optional |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/callScene?dsid=3504175fe000000000017ef3&sceneNumber=13
{
    "ok":true
}
```

### 4.5.2 saveScene

Tells the device to store the current output values as a default for the scene *sceneNumber*.

**Synopsis**
HTTP GET /json/device/saveScene

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneNumber | Numerical value | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/saveScene?dsid=3504175fe000000000017ef3&sceneNumber=5
{
    "ok":true
}
```

### 4.5.3 undoScene

Tells devices to restore the output values to the previous state if the current scene matches the *sceneNumber*.

**Synopsis**
HTTP GET /json/device/undoScene

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneNumber | Numerical value | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/undoScene?dsid=3504175fe000000000017ef3&sceneNumber=18
{
    "ok":true
}
```

### 4.5.4 turnOn

Tells devices to execute the scene MAX.

**Synopsis**
HTTP GET /json/device/turnOn

**Parameter**
None

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/turnOn?dsid=3504175fe000000000017ef3
{
    "ok":true
}
```

### 4.5.5 turnOff

Tells devices to execute the scene MIN.

**Synopsis**
HTTP GET /json/device/turnOff

**Parameter**
None

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/turnOff?dsid=3504175fe000000000017ef3
{
    "ok":true
}
```

### 4.5.6 increaseValue

Tells devices to execute the scene INC.

**Synopsis**
HTTP GET /json/device/increaseValue

**Parameter**
None

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/increaseValue?dsid=3504175fe000000000017ef3
{
    "ok":true
}
```

### 4.5.7 decreaseValue

Tells devices to execute the scene DEC.

**Synopsis**
HTTP GET /json/device/decreaseValue

**Parameter**
None

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/decreaseValue?dsid=3504175fe000000000017ef3
{
    "ok":true
}
```

## 4.6 Value

### 4.6.1 Set Value

Set the primary output value of a device to a given value.

> **Notice** Setting output values directly bypasses the group state machine and is unrecommended.

**Synopsis**
HTTP GET /json/device/setValue

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| value | Numerical 8 bit value, in the range from 0 to 255 | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setValue?dsid=3504175fe000000000017ef3&value=127
{
    "ok":true
}
```

### 4.6.2 Set Output Value

Set a output channel value of a device to a given value. The available output parameter ranges and channels depend on the feature of the hardware components.

> **Notice** Setting output values directly bypasses the group state machine and is unrecommended.

**Synopsis**
HTTP GET /json/device/setOutputValue

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| value | Numerical Value | Mandatory |
| offset | Output Channel Offset | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/device/setOutputValue?dsid=3504175fe000000000017ef3&value=5345&offset=0
{
    "ok":true
}
```

### 4.6.3 Get Output Value

Get the current output channel status of a device. The available output channels depend on the feature of the hardware components.

> **Notice** Getting output values directly from the device takes a noticeable amount of time. This request is subject of limitations in the systems certification rules.

**Synopsis**
HTTP GET /json/device/getOutputValue

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| offset | Output Channel Offset | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| result.offset | the given offset from the request |
| result.value | Numerical value of the selected output channel queried from the device |

**Sample**

```
GET /json/device/getOutputValue?dsid=3504175fe000000000017ef3&offset=0
{
    "ok": true, "result": { "offset": 0, "value": 5345 }
}
```

### 4.6.4 Get Scene Value

Retrieves the device value of the given scene.

**Synopsis**
HTTP GET /json/device/getSceneValue

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneID | Numerical value | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| result.value | value of the device |
| result.angle | angle value of the device, field available only for for GR-KL |

**Sample**

```
GET /json/device/getSceneValue?dsid=3504175fe000000000017ef3&sceneID=72
{
    "ok":true,"result":{"value":65535,"angle":255}
}
```

### 4.6.5 Set Scene Value

Retrieves the device value of the given scene.

**Synopsis**
HTTP GET /json/device/setSceneValue

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneID | Numerical value | Mandatory |
| value | Numerical value | Mandatory |
| angle | Numerical value, only applicable for GR-KL | Optional |

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/device/setSceneValue?dsid=3504175fe000000000016c4f&sceneID=72&value=26987
{
    "ok":true
}
```

### 4.6.6 Blink

Executes the "blink" function on a device for identification purposes.

**Synopsis**
HTTP GET /json/device/blink

**Parameter**
None

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/blink?dsid=3504175fe000000000017ef3
{
    "ok":true
}
```

### 4.6.7 Get Output Channel Value

Retrieve the value of one or more output channels of the device.

**Synopsis**
HTTP GET /json/device/getOutputChannelValue

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| channels | Semicolon separated list of channel names | Mandatory |

Currently supported channels are:

| Channel | Description | Minimum Value | Maximum Value | Unit |
|---|---|---|---|---|
| brightness | brightness | 0 | 100 | percent |
| hue | hue | 0 | 358,6 | degrees |
| saturation | saturation | 0 | 100 | percent |
| colortemp | color temperature | 100 | 1000 | mired |
| x | CIE color model | 0.0 | 1.0 | |
| y | CIE color model | 0.0 | 1.0 | |
| verticalpos | vertical position | 0 | 100 | percent |
| horizontalpos | horizontal position | 0 | 100 | percent |
| openinganglepos | opening angle position | 0 | 100 | percent |
| permeability | | 0 | 100 | percent |

**Response**
HTTP Status 200

| | |
|---|---|
| result.channels | array of channels and their values |
| result.channels[x].channel | output channel name |
| result.channels[x].value | output channel value |

**Sample**

```
GET /json/device/getOutputChannelValue?dsid=3504175fe000000000016c4f&
    channels=brightness;saturation

{ "ok":true, result: { channels: [ { channel: "brightness", value: 50 }, { channel: "saturation", value: 80
    } ] } }
```

### 4.6.8 Set Output Channel Value

Set the value of one or more output channels of the device.

**Synopsis**
HTTP GET /json/device/setOutputChannelValue

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| channelvalues | Semicolon separated list of channel names and their values | Mandatory |
| applyNow | Immediately apply the new values to the channe outputs | Optional, 1 (true) by default |

See getOutputChannelValue description for a list of output channel names and their value ranges.

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setOutputChannelValue?dsid=3504175fe000000000016c4f&
      channelvalues=brightness=10;saturation=100&applyNow=1
{ "ok":true }
```

### 4.6.9 Set Output Channel Don't Care Flag

Set don't care flag for one or more output channels.

**Synopsis**
HTTP GET /json/device/setOutputChannelDontCareFlag

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| channels | Semicolon separated list of channel names | Mandatory |
| dontCare | Don't care flag value, boolean | Mandatory (0 or 1) |
| sceneNumber | Scene number for which the flag will be set | Mandatory |

See getOutputChannelValue description for a list of output channel names.

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setOutputChannelDontCareFlag?dsid=3504175fe000000000016c4f&
      channels=brightness;saturation&dontCare=1&sceneNumber=1

{ "ok":true }
```

### 4.6.10 Get Output Channel Don't Care Flags

Get don't care flags for one or more output channels.

**Synopsis**
HTTP GET /json/device/getOutputChannelDontCareFlags

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| channels | Semicolon separated list of channel names | Mandatory |
| sceneNumber | Scene number for which the flag will be set | Mandatory |

See getOutputChannelValue description for a list of output channel names.

**Response**
HTTP Status 200

| | |
|---|---|
| result.channels | array of channels and their values |
| result.channels[x].channel | output channel name |
| result.channels[x].dontCare | output channel "don't care" flag value |

**Sample**

```
GET /json/device/getOutputChannelDontCareFlags?dsid=3504175fe000000000016c4f&
    channels=brightness;saturation&dontCare=1&sceneNumber=1

{ "ok":true, result: { channels: [ { channel: "brightness", dontCare: 0 }, { channel: "saturation",
    dontCare: 1 } ] } }
```

### 4.6.11 Get Output Channel Scene Value

Get scene value of one or more output channels.

**Synopsis**
HTTP GET /json/device/getOutputChannelSceneValue

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| channels | Semicolon separated list of channel names | Mandatory |
| sceneNumber | Number of the scene for which the values should be returned | Mandatory |

See getOutputChannelValue description for a list of output channel names.

**Response**
HTTP Status 200

| | |
|---|---|
| result.sceneID | Scene number for which the values are returned |
| result.channels | array of channels and their values |
| result.channels[x].channel | output channel name |
| result.channels[x].value | output channel value for the requested scene |

**Sample**

```
GET /json/device/getOutputChannelSceneValue?dsid=3504175fe000000000016c4f&
    channels=brightness;saturation&sceneNumber=1

{ "ok":true, result: { sceneID: 1, channels: [ { channel: "brightness", value: 40 }, { channel:
    "saturation", value: 20 } ] } }
```

### 4.6.12 Set Output Channel Scene Value

Set scene value of one or more output channels.

**Synopsis**
HTTP GET /json/device/setOutputChannelSceneValue

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| channelvalues | Semicolon separated list of channel names and their values | Mandatory |
| sceneNumber | Number of the scene for which the values should be set | Mandatory |

See getOutputChannelValue description for a list of output channel names.

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/device/setOutputChannelSceneValue?dsid=3504175fe000000000016c4f&
      channelvalues=brightness=10;saturation=100&sceneNumber=1

{ "ok":true }
```

## 4.7 Configuration

### 4.7.1 setButtonID

Sets the button ID of a device. For details about the push button configuration see the ds-basics reference document.

**Synopsis**
HTTP GET /json/device/setButtonID

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| buttonID | Button number to set | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setButtonID?dsid=3504175fe000000000016be7&buttonID=5
{
    "ok": true
}
```

### 4.7.2 setButtonInputMode

Sets the button input mode of a device. For details about the push button configuration see the ds-basics reference document.

**Synopsis**
HTTP GET /json/device/setButtonInputMode

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| modelID | Numerical value of the button mode to set | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setButtonInputMode?dsid=3504175fe000000000016be7&modelID=0
{
    "ok": true
}
```

### 4.7.3 setOutputMode

Sets the output mode of a device.

**Synopsis**
HTTP GET /json/device/setOutputMode

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| modelID | Numerical value of the output mode to set | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setOutputMode?dsid=3504175fe000000000016be7&modelID=0
{
    "ok": true
}
```

### 4.7.4 setJokerGroup

Sets the color group of a Joker device.

**Synopsis**
HTTP GET /json/device/setJokerGroup

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| groupID | Group number to set | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setJokerGroup?dsid=3504175fe000000000016be7&groupID=2
{
    "ok": true
}
```

### 4.7.5 setButtonActiveGroup

Sets the user group of a push button device.

**Synopsis**
HTTP GET /json/device/setButtonActiveGroup

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| groupID | Group number to set | Mandatory, value range between 0 and 63,use 0xff to reset |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setButtonActiveGroup?dsid=3504175fe000000000016be7&groupID=20
{
    "ok": true
}
```

### 4.7.6 getSceneMode

Reads the device configuration flags for a given *sceneID*. For details about the scene configuration see the ds-basics reference document.

**Synopsis**
HTTP GET /json/device/getSceneMode

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneID | Scene number for which the configuration is requested | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| sceneID | Scene number which has been requested |
| dontCare | Don't Care Flag |
| localPrio | Local Prio Flag |
| specialMode | Special Mode Flag |
| flashMode | Flashing Mode Flag |
| ledconIndex | Index of the LED configuration register |
| dimmTimeIndex | Index of the transition configuration register |

**Sample**

```
GET /json/device/getSceneMode?dsid=3504175fe000000000016be7&sceneID=5
{
    "ok": true,
    "result":
    {
        "sceneID": 5,
        "dontCare": false,
        "localPrio": false,
        "specialMode": false,
        "flashMode": false,
        "ledconIndex": 1,
```

```
        "dimtimeIndex": 1
    }
}
```

### 4.7.7 setSceneMode

Sets the device configuration flags for a given *sceneID*. For details about the scene configuration see the ds-basics reference document.

**Synopsis**
HTTP GET /json/device/setSceneMode

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sceneID | Scene number which has been requested | Mandatory |
| dontCare | Don't Care Flag | Optional |
| localPrio | Local Prio Flag | Optional |
| specialMode | Special Mode Flag | Optional |
| flashMode | Flashing Mode Flag | Optional |
| ledconIndex | Index of the LED configuration register | Optional |
| dimmtimeIndex | Index of the transition configuration register | Optional |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setSceneMode?dsid=3504175fe000000000016be7&sceneID=5&dimtimeIndex=2&
    dontCare=true { "ok": true }
```

### 4.7.8 getTransitionTime

Reads the device transition time configuration for a given register set. For details about the transition time configuration see the ds-basics reference document.

**Synopsis**
HTTP GET /json/device/getTransitionTime

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| dimtimeIndex | Index of the transition configuration register | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| dimmtimeIndex | Index of the transition configuration register |
| up | Ramptime up in Milliseconds |
| down | Ramptime down in Milliseconds |

**Sample**

```
GET /json/device/getTransitionTime?dsid=3504175fe000000000016be7&dimtimeIndex=2
{
    "ok": true,
    "result":
    {
        "dimtimeIndex": 2,
        "up": 600,
        "down": 55
    }
}
```

### 4.7.9 setTransitionTime

Sets the device transition time configuration for a given register set. For details about the transition time configuration see the ds-basics reference document.

**Synopsis**
HTTP GET /json/device/setTransitionTime

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| dimtimeIndex | Index of the transition configuration register | Mandatory |
| up | Ramptime up in Milliseconds | Mandatory |
| down | Ramptime down in Milliseconds | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setTransitionTime?dsid=3504175fe000000000016be7&dimtimeIndex=2&up=600&
      down=600
{
    "ok": true
}
```

### 4.7.10 setConfig

Write a configuration value of a config class parameter to the device.

> **Notice** Writing configuration parameters directly to the device may lead to malfunctions including complete failure of the whole device. Do not write parameters or values unless you are sure that the device supports it.

**Synopsis**
HTTP GET /json/device/setConfig

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| class | Configuration Class | Mandatory |
| index | Parameter Index | Mandatory |
| value | Parameter Value | Mandatory |

**Response**
HTTP Status 200

| class | the class parameter from the request |
|---|---|
| index | the index parameter from the request |
| value | parameter value |

**Sample**

```
GET /json/device/setConfig?dsid=3504175fe000000000016be7&class=3&index=0&value=33
{
    "ok": true
}
```

### 4.7.11 getConfig

Reads a 8 bit parameter value of a config class from the device.

> **Notice** Getting parameter values directly from the device takes a noticeable amount of time. This request is subject of limitations in the systems certification rules.

**Synopsis**
HTTP GET /json/device/getConfig

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| class | Configuration class | Mandatory |
| index | Parameter index | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| class | the class parameter from the request |
| index | the index parameter from the request |
| value | parameter value |

**Sample**

```
GET /json/device/getConfig?dsid=3504175fe000000000016be7&class=1&index=2
{
    "ok": true,
    "result":
    {
        "class": 1,
```

```
        "index": 2,
        "value": 231
    }
}
```

### 4.7.12 getConfigWord

Reads a 16 bit parameter value of a config class from the device.

> **Notice** Getting parameter values directly from the device takes a noticeable amount of time. This request is subject of limitations in the systems certification rules.

**Synopsis**
HTTP GET /json/device/getConfigWord

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| class | Configuration class | Mandatory |
| index | Parameter index, even | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| class | the class parameter from the request |
| index | the index parameter from the request |
| value | parameter value |

**Sample**

```
GET /json/device/getConfigWord?dsid=3504175fe000000000016be7&class=3&index=2
{
    "ok": true,
    "result":
    {
        "class": 3,
        "index": 2,
        "value": 65280
    }
}
```

### 4.7.13 setCardinalDirection

Write the cardinal direction of the device.

**Synopsis**
HTTP GET /json/device/setCardinalDirection

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| direction | the cardinal direction of this device. Allowed values:<br>none<br>north<br>north east<br>east<br>south east<br>south<br>south west<br>west<br>north west | |

**Response**
HTTP Status 200

**Sample**

```
GET /json/device/setCardinalDirection?dsid=3504175fe000000000016be7&direction=south\%20west
{
    "ok": true
}
```

### 4.7.14 getCardinalDirection

Read the configured cardinal direction of the device.

**Synopsis**
HTTP GET /json/device/getCardinalDirection

**Response**
HTTP Status 200

| | |
|---|---|
| direction | the cardinal direction of this device. Allowed values:<br>none<br>north<br>north east<br>east<br>south east<br>south<br>south west<br>west<br>north west |

**Sample**

```
GET /json/device/getCardinalDirection?dsid=3504175fe000000000016be7
{
    "ok": true,
    "result":
    {
        "direction": "south␣west"
    }
}
```

### 4.7.15 setWindProtectionClass

Write the wid protection class of the device.

**Synopsis**
HTTP GET /json/device/setWindProtectionClass

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| class | the wind protection class of this device. | |

**Response**
HTTP Status 200

**Sample**

```
GET /json/device/setWindProtectionClass?dsid=3504175fe000000000016be7&class=4
{
    "ok": true
}
```

### 4.7.16 getWindProtectionClass

Read the the wid protection class of the device.

**Synopsis**
HTTP GET /json/device/getWindProtectionClass

**Response**
HTTP Status 200

| | |
|---|---|
| class | the wind protection class of this device. |

**Sample**

```
GET /json/device/getWindProtectionClass?dsid=3504175fe000000000016be7
{
    "ok": true,
    "result":
    {
        "class": 2
    }
}
```

### 4.7.17 setFloor

Write floor number where the device is installed.

**Synopsis**
HTTP GET /json/device/setFloor

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| floor | the floor number where the device is installed. | |

**Response**
HTTP Status 200

**Sample**

```
GET /json/device/setFloor?dsid=3504175fe000000000016be7&floor=14
{
    "ok": true
}
```

### 4.7.18 getFloor

Read floor number where the device is installed.

**Synopsis**
HTTP GET /json/device/getFloor

**Response**
HTTP Status 200

| | |
|---|---|
| floor | the floor number where the device is installed. |

**Sample**

```
GET /json/device/getFloor?dsid=3504175fe000000000016be7
{
    "ok": true,
    "result":
    {
        "floor": 14
    }
}
```

## 4.8 Sensor

### 4.8.1 Get Sensor Value

Ready a sensor measurement from a device.

**Synopsis**
HTTP GET /json/device/getSensorValue

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sensorIndex | Numerical value, in the range from 0 to 14 | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| sensorIndex | the index parameter from the request |
| sensorValue | the actual measurement read from the device |

**Sample**

```
GET /json/device/getSensorValue?dsid=3504175fe000000000017ef3&sensorIndex=4
{
    "ok": true,
    "result": {
        "sensorIndex": 4,
        "sensorValue": 0
    }
}
```

### 4.8.2 Get Sensor Type

Ready the sensor type description from a device. For details about sensor types see the ds-basics reference document.

**Synopsis**
HTTP GET /json/device/getSensorType

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| sensorIndex | Numerical value, in the range from 0 to 14 | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| sensorIndex | the index parameter from the request |
| sensorType | the sensor type read from the device |

**Sample**

```
GET /json/device/getSensorType?dsid=3504175fe000000000017ef3&sensorIndex=4
{
    "ok": true,
    "result": {
        "sensorIndex": 4,
        "sensorType": 6
    }
}
```

### 4.8.3 getSensorEventTableEntry

Reads the device event configuration for a given index. For details about the event table configuration see the ds-basics reference document.

**Synopsis**
HTTP GET /json/device/getSensorEventTableEntry

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| eventIndex | Index of the event configuration entry | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| eventIndex | Index of the event configuration register |
| eventName | User defined name of this event |
| sensorIndex | Sensor index on which this entry operates |
| action | Action value |
| value | Threshold value |
| test | Comparison operator |
| hysteresis | Hysteresis value |
| validity | Enabled Flag |

**Sample**

```
GET /json/device/getSensorEventTableEntry?dsid=3504175fe00000000001540c&eventIndex=0
{
    "ok": true,
    "result": {
        "eventIndex": 0,
        "eventName": "",
        "sensorIndex": 2,
        "test": 2,
        "action": 0,
        "value": 35,
        "hysteresis": 0,
        "validity": 2
    }
}
```

### 4.8.4 setSensorEventTableEntry

Sets the device event configuration for a given index. For details about the event table configuration see the ds-basics reference document.

**Synopsis**
HTTP GET /json/device/setSensorEventTableEntry

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| eventIndex | Index of the event configuration register | Mandatory |
| eventName | User defined name of this event | Mandatory |
| sensorIndex | Sensor index on which this entry operates | Mandatory |
| action | Action value | Mandatory |
| value | Threshold value | Mandatory |
| test | Comparison operator | Mandatory |
| hysteresis | Hysteresis value | Mandatory |
| validity | Enabled Flag | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setSensorEventTableEntry?dsid=3504175fe00000000001540c&eventIndex=0&
    eventName="TV␣turned␣on"&sensorIndex=2&test=2&action=0&value=50&hysteresis=25&
    validity=2
{
   "ok": true
}
```

## 4.9 Programming

### 4.9.1 Set Programming Mode

Enabled or disabled the programming mode on a device.

**Synopsis**
HTTP GET /json/device/setProgMode

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| mode | mode value, either enabled or disabled | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setProgMode?dsid=3504175fe000000000017ef3&mode=disabled
{
    "ok": true
}
```

### 4.9.2 Add To Area

Modify the device scene table configuration and activate the area scene.

**Synopsis**
HTTP GET /json/device/addToArea

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| areaScene | either the area-on or area-off scenes | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/addToArea?dsid=3504175fe000000000017ef3&areaScene=7
{
    "ok": true
}
```

### 4.9.3 Remove From Area

Modify the device scene table configuration and deactivate the area scene.

**Synopsis**
HTTP GET /json/device/removeFromArea

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| areaScene | either the area-on or area-off scenes | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/removeFromArea?dsid=3504175fe000000000017ef3&areaScene=7
{
    "ok": true
}
```

### 4.9.4 Get Transmission Quality

Sends test commands to a device to evaluate the actual transmission quality.

**Synopsis**
HTTP GET /json/device/getTransmissionQuality

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| upstream | a numerical value in the range of 0 to 62, 62 meaning best quality |
| downstream | a numerical value in the range 0 to 6, 0 meaning best quality |

**Sample**

```
GET /json/device/getTransmissionQuality?dsid=3504175fe000000000017ef3
{
    "ok": true,
    "result": {
        "upstream": 61,
        "downstream": 0
    }
}
```

## 4.10 Heating and valve actuators

### 4.10.1 setHeatingGroup

Sets the standard color group of a heating actuator. Some actuators support operation with different connected hardware equipment, therefore the terminal blocks support operation in different zone groups, for example in heating, cooling or ventilation.

**Synopsis**
HTTP GET /json/device/setHeatingGroup

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| groupID | New group Id | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/device/setHeatingGroup?dsuid=3504175fe0000000000000000016be700&groupID=48
{
    "ok": true
}
```

### 4.10.2 getValvePwmState

Reads the device status of a valve PWM actuator.

**Synopsis**
HTTP GET /json/device/getValvePwmState

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| pwmValue | Current PWM control value in percent |
| pwmPriorityMode | Current operating state value |

**Sample**

```
GET /json/device/getValvePwmState?dsuid=3504175fe000000000000000000016be700
{
    "ok": true,
    "result":
    {
        "pwmValue": 60,
        "pwmPriorityMode": 0
    }
}
```

### 4.10.3 getValvePwmMode

Reads the device configuration of a valve PWM actuator.

**Synopsis**
HTTP GET /json/device/getValvePwmMode

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| pwmPeriod | Length of PWM period in seconds |
| pwmMinX | Minimum set point or threshold |
| pwmMaxX | Maximum set point or threshold |
| pwmMinY | Minimum output value at min set point |
| pwmMaxY | Maximum output value at max set point |
| pwmOffset | Set point offset |

**Sample**

```
GET /json/device/getValvePwmMode?dsuid=3504175fe000000000000000000016be700
{
    "ok": true,
    "result":
    {
        "pwmPeriod": 900,
        "pwmMinX": 0,
        "pwmMaxX": 10,
        "pwmMinY": 12,
        "pwmMaxY": 28,
        "pwmOffset": -20
    }
}
```

### 4.10.4 setValvePwmMode

Sets the device configuration of a valve PWM actuator.

**Synopsis**
HTTP GET /json/device/setValvePwmMode

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| pwmPeriod | Length of PWM period in seconds, 0 .. 64k | Optional |
| pwmMinX | Minimum set point or threshold, 0 .. 255 | Optional |
| pwmMaxX | Maximum set point or threshold, 0 .. 255 | Optional |
| pwmMinY | Minimum output value at min set point, 0 .. 255 | Optional |
| pwmMaxY | Maximum output value at max set point, 0 .. 255 | Optional |
| pwmOffset | Set point offset, -128 .. 127 | Optional |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/device/setValvePwmMode?dsuid=3504175fe000000000000000000016be700&pwmMaxX=80&
      pwmOffset=-20
{
  "ok": true
}
```

### 4.10.5 getValveControlMode

Reads the device configuration of a valve PWM actuator.

**Synopsis**
HTTP GET /json/device/getValveControlMode

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| ctrlNONC | Configure normally closed (false) or normally open (true) output behavior |
| ctrlClipMaxHigher | PWM value over maximum control value to 100 percent |
| ctrlClipMinLower | PWM lower than maximum control value to 0 percent |
| ctrlClipMinZero | Control value of zero maps to 0 percent PWM |

**Sample**

```
GET /json/device/getValveControlMode?dsuid=3504175fe0000000000000000016be700
{
    "ok": true,
    "result":
    {
        "ctrlNONC": true,
        "ctrlClipMaxHigher": false,
        "ctrlClipMinLower": false,
        "ctrlClipMinZero": false
    }
}
```

### 4.10.6 setValveControlMode

Sets the device configuration of a valve PWM actuator.

**Synopsis**
HTTP GET /json/device/setValveControlMode

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| ctrlClipMinZero | Control value of zero maps to 0 percent PWM | Optional |
| ctrlClipMinLower | PWM lower than minimum forces control value to 0 percent | Optional |
| ctrlClipMaxHigher | PWM value over maximum forces control value to 100 percent | Optional |
| ctrlNONC | Configure normally open or normally closed output behavior | Optional |

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/device/setValveControlMode?dsuid=3504175fe0000000000000000016be700&
        ctrlNONC=false
{
  "ok": true
}
```

### 4.10.7 getValveTimerMode

Reads the timer device configuration settings of a valve PWM actuator.

**Synopsis**
HTTP GET /json/device/getValveTimerMode

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| valveProtectionTimer | Duration of the valve protection period in seconds |
| emergencyValue | Fixed output value in percent in emergency mode |
| emergencyTimer | Duration in seconds until emergency mode is activated |

**Sample**

```
GET /json/device/getValveTimerMode?dsuid=3504175fe0000000000000000016be700
{
    "ok": true,
    "result":
    {
        "valveProtectionTimer": 0,
        "emergencyValue": 75,
        "emergencyTimer": 7200
    }
}
```

### 4.10.8 setValveTimerMode

Sets the timer device configuration of a valve PWM actuator.

**Synopsis**
HTTP GET /json/device/setValveTimerMode

### Parameter

| Parameter | Description | Remarks |
|---|---|---|
| valveProtectionTimer | Duration of the valve protection period in seconds | Optional |
| emergencyValue | Fixed output value in percent in emergency mode | Optional |
| emergencyTimer | Duration in seconds until emergency mode is activated | Optional |

### Response

HTTP Status 200

| | |
|---|---|
| ok | true |

### Sample

```
GET /json/device/setValveTimerMode?dsuid=3504175fe000000000000000016be700&
      valveProtectionTimer=600
{
  "ok": true
}
```

# 5 Circuit

## 5.1 Common

Every /json/circuit/ function uses a common selection scheme for the digitalSTROM Meter to which the command refers to. The parameter "id" must be given to identify the dSM which is a string value of the dSID.

| Parameter | Description | Remarks |
|---|---|---|
| id | dSID Number of the digitalSTROM Meter | Mandatory |

A missing dsid identifier result in the following error message to be returned.

```
{
    "ok": false,
    "message": "Missing parameter id"

}
```

If a dSID identifier does not match any actually known digitalSTROM Meter in the installation the following error message is returned.

```
{
     "ok": false,
     "message": "Could not find dSMeter with given dsid"
}
```

## 5.2 Name

### 5.2.1 getName

Returns the user defined name of the zone.

**Synopsis**
HTTP GET /json/circuit/getName

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| name | identifier string for the digitalSTROM Meter |

**Sample**

```
GET /json/circuit/getName?id=3504175fe0000010000004d5
{
    "ok":true,
    "result" : {
        "name" : "Wohnen/Flur/Eingang"
    }
}
```

### 5.2.2 setName

Sets the zone name.

**Synopsis**
HTTP GET /json/circuit/setName

| Parameter | Description | Remarks |
|---|---|---|
| newName | identifier string for the digitalSTROM Meter | Mandatory |

**Parameter**

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/circuit/setName?id=3504175fe0000010000004d5&newName="Wohnen"
{
    "ok":true
}
```

## 5.3 Energy Meter

### 5.3.1 getConsumption

Returns the current measurent of the power consumption on this circuit.

**Synopsis**
HTTP GET /json/circuit/getConsumption

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| consumption | Current power consumption [W] |

**Sample**

```
GET /json/circuit/getConsumption?id=3504175fe0000010000004d5
{
    "ok": true,
    "result" : {
        "consumption": 725
    }
}
```

### 5.3.2 getEnergyMeterValue

Returns the current measurent of the power consumption on this circuit.

**Synopsis**
HTTP GET /json/circuit/getEnergyMeterValue

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| meterValue | Energy Meter Value [Ws] |

**Sample**

```
GET /json/circuit/getEnergyMeterValue?id=3504175fe0000010000004d5
{
    "ok": true,
    "result": {
        "meterValue": 1438467
    }
}
```

# 6 Structure

## 6.1 Zone

### 6.1.1 addZone

Adds a zone with the given Id. The zone is added to the digitalSTROM Server data model only and initially does not have any devices associated.

**Synopsis**
HTTP GET /json/structure/addZone

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| zoneID | unique numerical identifier for the new zone | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/structure/addZone?zoneID=1
{
    "ok":true
}
```

### 6.1.2 removeZone

Removes the zone with the give Id from the installation. A zone can only be removed if it has no associated devices.

**Synopsis**
HTTP GET /json/structure/removeZone

| Parameter | Description | Remarks |
|---|---|---|
| zoneID | unique numerical identifier | Mandatory |

**Parameter**

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/structure/removeZone?zoneID=1234
{
    "ok":true
}
```

## 6.2 Group

### 6.2.1 addGroup

Adds a user group to the zone with the given Id.

**Synopsis**
HTTP GET /json/structure/addGroup

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| zoneID | unique numerical identifier for the new zone | Mandatory |
| groupID | numerical identifier for the new group | Mandatory if groupAutoSelect is not given |
| groupAutoSelect | group type 'user' or 'global' | Mandatory if groupID is not given |

**Response**
HTTP Status 200

| | |
|---|---|
| result.groupID | numeric identifier for the new group |
| result.zoneID | numeric identifier for the zone |
| result.groupName | string, name of the new group |
| result.groupColor | numeric identifier, color of the new group |

**Sample**

```
GET /json/structure/addGroup?zoneID=0&groupAutoSelect=global&groupColor=5&groupName=test
{
    "ok":true,
    "result":
    {
        "groupID":18,
        "zoneID":0,
        "groupName":"test",
        "groupColor":5
    }
}
```

### 6.2.2 removeGroup

Removes a user group to the zone with the given Id.

**Synopsis**
HTTP GET /json/structure/removeGroup

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| zoneID | unique numerical identifier for the zone | Mandatory |
| groupID | numerical identifier for the group | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| ok | true |

**Sample**

```
GET /json/structure/removeGroup?zoneID=0&groupID=18
{
    "ok":true
}
```

### 6.2.3 groupAddDevice

Adds a device to the user group. Only active devices can be added to additional groups.

**Synopsis**
HTTP GET /json/structure/groupAddDevice

| Parameter | Description | Remarks |
|---|---|---|
| deviceID | DSID of the device | Mandatory |
| groupID | unique numerical group identifier | Mandatory |

**Parameter**

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/structure/groupAddDevice?deviceID=3504175fe000000000005854&groupID=16
{
    "ok":true
}
```

### 6.2.4 groupSetName

Rename a group.

**Synopsis**
HTTP GET /json/structure/groupSetName

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| zoneID | unique numerical identifier for the zone | Mandatory |
| groupID | numerical identifier for the group | Mandatory |
| newName | string, new name for the group | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/structure/groupSetName?zoneID=0&groupID=18&newName=test
{
    "ok":true
}
```

### 6.2.5 groupSetColor

Change color of the group.

**Synopsis**
HTTP GET /json/structure/groupSetColor

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| zoneID | unique numerical identifier for the zone | Mandatory |
| groupID | numerical identifier for the group | Mandatory |
| newColor | numerical identifier of the color for the group | Mandatory |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/structure/groupSetColor?zoneID=0&groupID=18&newColor=4
{
    "ok":true
}
```

## 6.3 Device

### 6.3.1 zoneAddDevice

Associates a device with a new zone. A device is automatically removed from the old zone. Only active devices can be moved to a new zone because the zone configuration has to be synchronized with the device itself.

**Synopsis**
HTTP GET /json/structure/zoneAddDevice

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| deviceID | DSID of the device to move | Mandatory |
| zone | unique numerical identifier for the new zone | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| movedDevices | array of devices which have been moved |

In the case of a failure various different error messages may occur.

**Sample**

```
GET /json/structure/zoneAddDevice?deviceID= &zone=1
{
    ok: true
    result: {
        movedDevices: [
            {
                id: "3504175fe000000000005854"
                name: ""
                functionID: 4144
                productRevision: 788
                productID: 1234
                hwInfo: "GE-TKM210"
                meterDSID: "3504175fe0000010000004d9"
                busID: 241
                zoneID: 1
                isPresent: true
                lastDiscovered: "2012-11-22 10:35:05"
                firstSeen: "2012-11-19 14:34:02"
                inactiveSince: "1970-01-01 01:00:00"
                outputMode: 16
                buttonID: 12
                buttonActiveGroup: 1
                buttonInputMode: 0
                buttonInputIndex: 0
                buttonInputCount: 1
                groups: [
                    "1"
                ]
            }
        ]
    }
}
```

### 6.3.2 removeDevice

Removes a device from the data model. Only inactive devices can be removed.

**Synopsis**
HTTP GET /json/structure/removeDevice

| Parameter | Description | Remarks |
|---|---|---|
| deviceID | DSID of the device to be removed | Mandatory |

**Parameter**

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/structure/removeDevice?deviceID=3504175fe000000000005854
{
    ok: false
    message: "Cannot␣remove␣present␣device"
}
```

# 7 Event

## 7.1 Raise

### 7.1.1 raise

Raises an event and appends it to the digitalSTROM Server event queue. Details of the digitalSTROM Server event processing can be found in the system-interfaces document.

> **Notice** System events should be treated as reserved and must not be raised by external applications. In this term system events are events which originate from the digitalSTROM system lower layers.

**Synopsis**
HTTP GET /json/event/raise

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| name | identifier string for event | Mandatory |
| parameter | list of key=value pairs, seperated with semicolons | Optional |

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/event/raise?name=highlevelevent&parameter=id=1026;value=0;index=1
{
    "ok":true
}
```

## 7.2 Subscription

### 7.2.1 subscribe

Subscribe to an event with the given name and registers the callers subscriptionId. A unique subscriptionId can be selected by the subscriber. It is possible to subscribe to several events reusing the same subscriptionId.

**Synopsis**
HTTP GET /json/event/subscribe

| Parameter | Description | Remarks |
|---|---|---|
| name | identifier string for the event | Mandatory |
| subscriptionID | numerical unique value | Mandatory |

**Parameter**

**Response**
HTTP Status 200

| ok | true |
|---|---|

**Sample**

```
GET /json/event/subscribe?name=deviceSensorEvent&subscriptionID=42
{
    "ok":true
}
```

### 7.2.2 unsubscribe

Unsubscribes for the previously registered events by giving the event name and the unique subscriptionId.

**Synopsis**
HTTP GET /json/event/unsubscribe

| Parameter | Description | Remarks |
|---|---|---|
| name | identifier string for the event | Mandatory |
| subscriptionID | numerical unique value | Mandatory |

**Parameter**

**Response**
HTTP Status 200

| ok | true |
|---|---|

If there is no registered session for the given event name the following error message is returned.

```
{
    ok: false
    message: "Event␣"callScene"␣is␣not␣subscribed␣in␣this␣session"
}
```

If the subscriptionId is unknown to the digitalSTROM Server the following error message is returned.

```
{
    ok: false
    message: "Token␣not␣found!"
}
```

**Sample**

```
GET /json/event/unsubscribe?name=callScene&subscriptionID=42
{
    ok: true
}
```

### 7.2.3 get

Get event and context information for an event subscription. All events subscribed with the given Id will be handled by this call. An optional timeout value in milliseconds can be specified and will block the call until either an event or the timeout occurs. If the timeout value is zero or missing the call will not timeout.

**Synopsis**
HTTP GET /json/event/get

| Parameter | Description | Remarks |
|---|---|---|
| subscriptionID | numerical unique value | Mandatory |
| timeout | numerical value, timeout in milli seconds | Optional |

**Parameter**

**Response**
HTTP Status 200

| | |
|---|---|
| events | array of events |

### Sample

```
GET /json/event/get?subscriptionID=42&timeout=60000
{
    ok: true
    result: {
        events: [ ]
    }
}
```

```
GET /json/event/get?subscriptionID=42&timeout=60000
{
    ok: true
    result: {
        events: [
            {
                name: "callScene"
                properties: {
                    groupID: "1"
                    sceneID: "8"
                    zoneID: "1241"
                    originDeviceID: "3504175fe000000000005854"
                }
            }
        ]
    }
}
```

# 8 Metering

## 8.1 Metering

### 8.1.1 getResolution

Returns a list of time-series metering data resolutions stored on the digitalSTROM Server.

**Synopsis**
HTTP GET /json/metering/getResolutions

**Parameter**
None

**Response**
HTTP Status 200

| Parameter | Description |
|---|---|
| ok | boolean result of the call |
| result.resolutions | a list of supported resolutions |
| result.resolutions[...].resolution | step size in thei resolution in seconds |

**Sample**

```
GET /json/metering/getResolutions
{
    "ok": true,
    "result": {
        "resolutions": [
            {
                "resolution": 1
            },
            {
                "resolution": 60
            },
            {
                "resolution": 900
            },
            {
                "resolution": 86400
            },
            {
                "resolution": 604800
            },
            {
                "resolution": 2592000
            }
        ]
    }
}
```

### 8.1.2 getSeries

Returns a list of all metering series stored on the digitalSTROM Server.
Three types of series are available:

**energy** An energy meter counter.

**energyDelta** The total energy consumed during the previous time slot.

**consumption** The average power used during the previous time slot.

**Synopsis**
HTTP GET /json/metering/getSeries

**Parameter**
None

**Response**
HTTP Status 200

| Parameter | Description |
|---|---|
| ok | boolean result of the call |
| result.series | a list of available time series |
| result.series[...].dsid | dSID of the digitalSTROM Meter for this series |
| result.series[...].type | the series type |

**Sample**

```
GET /json/metering/getSeries
{
    "ok": true,
    "result": {
        "series": [
            {
                "dsid": "3504175fe00000100000053e",
                "type": "energy"
            },
            {
                "dsid": "3504175fe0000010000006b4",
                "type": "energyDelta"
            },
            {
                "dsid": "3504175fe0000010000008a5",
                "type": "consumption"
            },
        ]
    }
}
```

### 8.1.3 getValues

Returns a time series of metering values with the specified properties.

All times are integers that represent UNIX timestamps (seconds since 1970-01-01).

The (optional) window selection parameters can be used in different combinations. Only two of the three options can be used together in a call. The following table details the available combinations:

**startTime** return all available values starting at startTime until now.

**endTime** return all available values from the oldest available until endTime.

**valueCount** return the valueCount newest values

**startTime and valueCount** return valueCount values starting from startTime.

**endTime and valueCount** return valueCount values ending at endTime

**startTime and endTime** return the values between startTime and endTime.

**Synopsis**
HTTP GET /json/metering/getValues

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| dsid | request the data for this digitalSTROM Meter | Mandatory |
| type | series type (according to the getSeries call) | Mandatory |
| resolution | series resolution (the digitalSTROM Server will adjust the resolution to the closest multiple of the available resolutions according to the getResolutions call) | Mandatory |
| unit | (only relevant for types "energy" and "energyDelta") unit of the returned metering values. Options are "Wh" and "Ws". Defaults to "Wh". | Optional |
| startTime | start time (UNIX timestamp) | Optional |
| endTime | ent time (UNIX timestamp) | Optional |
| valueCount | number of values (UNIX timestamp) | Optional |

**Response**
HTTP Status 200

| Parameter | Description |
|---|---|
| ok | boolean result of the call |
| result.meterID | dSID of the digitalSTROM Meter |
| result.type | same as Request |
| result.resolution | actual resolution of the data, might differ from the requested resolution if it was not available. |
| result.values | array of time-value pairs |

**Sample**

```
GET /json/metering/getValues?dsid=3504175fe00000100000063a&type=energy&resolution=60&
      unit=Ws&valueCount=5
{
    "ok": true,
    "result": {
        "meterID": "3504175fe00000100000063a",
        "type": "energy",
        "unit": "Ws",
        "resolution": "60",
        "values": [
            [
                1352906040,
                47562600
            ],
            [
                1352906100,
                47562600
```

```
            ],
            [
                1352906160,
                47562600
            ],
            [
                1352906220,
                47562600
            ],
            [
                1352906280,
                47562600
            ]
        ]
    }
}
```

### 8.1.4 getLatest

Returns the latest available metering values.

**Synopsis**
HTTP GET /json/metering/getLatest

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| from | the dSID of the requested digitalSTROM Meters. It uses a Set-Syntax: ".meters(dsid1,dsid2,...)" and ".meters(all)" | Mandatory |
| type | series type (according to the getSeries call) | Mandatory |
| unit | (only relevant for types "energy" and "energyDelta") unit of the returned metering values. Options are "Wh" and "Ws". Defaults to "Wh". | Optional |

**Response**
HTTP Status 200

| Parameter | Description |
|---|---|
| ok | boolean result of the call |
| result.values | array of results |
| result.values[...].dsid | dSID of the digitalSTROM Meter |
| result.values[...].value | the latest metering value |
| result.values[...].date | date and time when the latest metering value was recorded |

### Sample

```
GET
        /json/metering/getLatest?from=.meters(3504175fe00000100000063a,3504175fe0000010000008c4)&
        type=energy&unit=Ws
```


```
{
    "ok": true,
    "result": {
        "values": [
            {
                "dsid": "3504175fe00000100000063a",
                "value": 49414887,
                "date": "2012-11-19␣13:49:41"
            },
            {
                "dsid": "3504175fe0000010000008c4",
                "value": 151215631,
                "date": "2012-11-19␣10:29:29"
            }
        ]
    }
}
```

# 9 System

## 9.1 System Information

### 9.1.1 version

Returns the version of the digitalSTROM Server software.

**Synopsis**
HTTP GET /json/system/version

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| version | the dSS application version |
| distroVersion | the host platform firmware release (since v1.10) |
| Hardware | the host platform hardware identifier (since v1.10) |
| Revision | the host platform hardware revision number (since v1.10) |
| Serial | the host platform hardware serial number (since v1.10) |
| Ethernet | the host platform IEEE Mac address (since v1.10) |
| MachineID | the host system unique id (since v1.10) |
| Kernel | the host system Linux kernel release string (since v1.10) |

**Sample**

```
GET /json/system/version
{
    "ok":true,
    result":
␣␣␣{
␣␣␣␣␣␣"version":␣"dSS v1.31.1 (git:2acc82fe90f273a788fb573b07419f29f369a02a) (oebuild@builder)",
␣␣␣␣␣␣"distroVersion":␣"Angstrom 2010.4-devel-20111031",
␣␣␣␣␣␣"Hardware":␣" AIZO digitalSTROM Server",
␣␣␣␣␣␣"Revision":␣" 0000",
␣␣␣␣␣␣"Serial":␣" 0000000000000000",
␣␣␣␣␣␣"EthernetID":␣"a8:99:5c:c0:00:27",
␣␣␣␣␣␣"MachineID":␣"603a932537518b121da4ffad00000037",
␣␣␣␣␣␣"Kernel":␣"Linux version 2.6.32.8 (jin@vsrv-pilot-feedback) (gcc version 4.3.3 (GCC) ) #1
        Mon Jan 31 18:55:47 CET 2011"
␣␣}
}
```

### 9.1.2 time

Gets the installation time.

**Synopsis**
HTTP GET /json/system/time

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| time | number of seconds since the Epoch, 1970-01-01 00:00:00 +0000 (UTC) |

**Sample**

```
GET /json/system/time
{
  "ok":true,
  "result" :
  {
    "time" : 1353510820
  }
}
```

### 9.1.3 getDSID

Returns the dSUID and dSID of the digitalSTROM Server.

**Synopsis**
HTTP GET /json/system/getDSID

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| dSID | dSID = SGTIN-96 of the dSS |
| dSUID | dSUID of the dSS |

**Sample**

```
GET /json/systme/getDSID
{
  "ok":true,
  "result" :
```

```
  {
    "dSID": "303505d7f800182000c00027",
    "dSUID": "303505d7f80000000000182000c0002700"
  }
}
```

## 9.2 Authentication

### 9.2.1 login

Creates a new session using the provided credentials.

**Synopsis**
HTTP GET /json/system/login

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| user | user name string | Mandatory |
| password | password string | Mandatory |

**Response**
HTTP Status 200

| result.token | session token as string |
|---|---|

**Sample**

```
GET /json/system/login?user=dssadmin\&password=dssadmin

{
  "ok" : true,
  "result" : { "token" : "cea026b6f9d69e57e030736076285da77dbf117d24dbec69e349b2fb4ab7425e" }
}
```

### 9.2.2 logout

Destroys the session and signs out the user.

**Synopsis**
HTTP GET /json/system/logout

**Parameter**
None

**Response**
HTTP Status 200

**Sample**

```
GET /json/system/logout { "ok" : true }
```

### 9.2.3 loggedInUser

Returns the name of the currently logged in user.

**Synopsis**
HTTP GET /json/system/loggedInUser

**Parameter**
None

**Response**
HTTP Status 200

| | |
|---|---|
| result.name | name of the currently logged in user |

Note: if noone is currently logged in, the result will be empty, i.e. name will be missing.

**Sample**

```
GET /json/system/loggedInUser

{ "ok" : true, "result" : { "name" : "dssadmin" } }
```

### 9.2.4 setPassword

Changes the password of the currently logged in user.

**Synopsis**
HTTP GET /json/system/setPassword

| Parameter | Description | Remarks |
|---|---|---|
| password | new password | Mandatory |

**Parameter**

**Response**
HTTP Status 200

**Sample**

```
GET /json/system/setPassword

{ "ok" : true, "message" : "Password␣changed,␣have␣a␣nice␣day" }
```

### 9.2.5 requestApplicationToken

Returns a token for paswordless login. The token will need to be approved by a user first, the caller must not be logged in.

**Synopsis**
HTTP GET /json/system/requestAppplicationToken

| Parameter | Description | Remarks |
|---|---|---|
| applicationName | name of the application that requests the token | Mandatory |

**Parameter**

**Response**
HTTP Status 200

| | |
|---|---|
| result.applicationToken | application token as string |

**Sample**

```
GET /json/system/requestApplicationToken?applicationName=Example

{
    "ok" : true,
    "result" :
    {
        "applicationToken" :
            "4fa07386c77d7f32260066c83b58aece5814698376bd03f0e3b5764e58f0ec1a"
    }
}
```

### 9.2.6 enableToken

Enables an application token, caller must be logged in.

**Synopsis**
HTTP GET /json/system/enableToken

| Parameter | Description | Remarks |
|---|---|---|
| applicationToken | application token as string | Mandatory |

**Parameter**

**Response**
HTTP Status 200

**Sample**

```
GET
    /json/system/enableToken?applicationToken=4fa07386c77d7f32260066c83b58aece5814698376bd03f0e3b5764e58f0ec1a

{ "ok" : true }
```

### 9.2.7 revokeToken

Revokes an application token, caller must be logged in.

**Synopsis**
HTTP GET /json/system/revokeToken

| Parameter | Description | Remarks |
|---|---|---|
| applicationToken | application token as string | Mandatory |

**Parameter**

**Response**
HTTP Status 200

**Sample**

```
GET
    /json/system/revokeToken?applicationToken=4fa07386c77d7f32260066c83b58aece5814698376bd03f0e3b5764e58f0ec1a

{ "ok" : true }
```

### 9.2.8 loginApplication

Creates a new session using the registered application token.

**Synopsis**
HTTP GET /json/system/loginApplication

| Parameter | Description | Remarks |
|---|---|---|
| loginToken | application token as string | Mandatory |

**Response**
HTTP Status 200

| result.token | session token as string |
|---|---|

**Sample**

```
GET
      /json/system/loginApplication?loginToken=4fa07386c77d7f32260066c83b58aece5814698376bd03f0e3b5764e58f0ec1a

{
    "ok" : true,
    "result" : { "token" : "a84bfd1219512078d5537a4a5cd1c78084e6c4d3f8b0ef2ae3a2c81dff638822" }
}
```

# 10 Property Tree

## 10.1 Basic Property Tree Operations

### 10.1.1 getString

Returns the string value of the property, this call will fail if the property is not of type 'string'.

**Synopsis**
HTTP GET /json/property/getString

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| path | path of the property | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| result.value | string value of the property |

**Sample**

```
GET /json/property/getString?path=/system/version/version

{ "ok" : true, "result" : { "value" : "1.17.3" } }
```

### 10.1.2 setString

Sets the string value of the property, this call will fail if the property is not of type 'string'.

**Synopsis**
HTTP GET /json/property/setString

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| path | path of the property | Mandatory |
| value | string value to set | Mandatory |

**Response**
HTTP Status 200

**Sample**

```
GET /json/property/setString?path=/testpath/teststring\&value=testvalue

{ "ok" : true }
```

### 10.1.3 getInteger

Returns the integer value of the property, this call will fail if the property is not of type 'integer'.

**Synopsis**
HTTP GET /json/property/getInteger

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| path | path of the property | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| result.value | integer value of the property |

**Sample**

```
GET /json/property/getInteger?path=/system/uptime

{ "ok" : true, "result" : { "value" : 7539 } }
```

### 10.1.4 setInteger

Sets the integer value of the property, this call will fail if the property is not of type 'integer'.

**Synopsis**
HTTP GET /json/property/setInteger

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| path | path of the property | Mandatory |
| value | integer value of the property | Mandatory |

**Response**
HTTP Status 200

**Sample**

```
GET /json/property/setInteger?path=/testpath/testint\&value=1

{ "ok" : true }
```

### 10.1.5 getBoolean

Returns the boolean value of the property, this call will fail if the property is not of type 'boolean'.

**Synopsis**
HTTP GET /json/property/getBoolean

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| path | path of the property | Mandatory |

**Response**
HTTP Status 200

| result.value | boolean value of the property |
|---|---|

**Sample**

```
GET /json/property/getBoolean?path=/config/subsystems/Metering/enabled

{ "ok" : true, "result" : { "value" : true } }
```

### 10.1.6 setBoolean

Returns the boolean value of the property, this call will fail if the property is not of type 'boolean'.

**Synopsis**
HTTP GET /json/property/setBoolean

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| path | path of the property | Mandatory |
| value | boolean value of the property | Mandatory |

**Response**
HTTP Status 200

**Sample**

```
GET /json/property/setBoolean?path=/testpath/testbool\&value=true

{ "ok" : true }
```

### 10.1.7 getChildren

Returns an array of child nodes.

**Synopsis**
HTTP GET /json/property/getChildren

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| path | path of the node | Mandatory |

**Response**
HTTP Status 200

| result[] | result is an array of child nodes |
|---|---|

**Sample**

```
GET /json/property/getChildren?path=/system/host/interfaces/lo

{
    "ok" : true,
    "result":
    [
        { "name" : "mac", "type" : "string"},
        { "name" : "ip", "type" : "string"},
        { "name" : "netmask", "type" : "string"}
    ]
}
```

### 10.1.8 getType

Returns the type of the property, this can be "none", "string", "integer" or "boolean".

**Synopsis**
HTTP GET /json/property/getType

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| path | path of the property | Mandatory |

**Response**
HTTP Status 200

| result.type | type of the property |
|---|---|

**Sample**

```
GET /json/property/getType?path=/system/host/interfaces/lo/mac

{ "ok" : true, "result" : { "type" : "string" } }
```

### 10.1.9 getFlags

Returns the flag values of a property.

**Synopsis**
HTTP GET /json/property/getFlags

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| path | path of the property | Mandatory |

**Response**
HTTP Status 200

| | |
|---|---|
| result.READABLE | information about the READABLE flag |
| result.WRITEABLE | information about the WRITEABLE flag |
| result.ARCHIVE | information about the ARCHIVE flag |

**Sample**

```
GET /json/property/getFlags?path=/system/host/interfaces/lo/mac

{ "ok" : true, "result" : { "READABLE" : true, "WRITEABLE" : true, "ARCHIVE" : false } }
```

### 10.1.10 setFlag

Sets a given flag of a property.

**Synopsis**
HTTP GET /json/property/setFlag

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| path | path of the property | Mandatory |
| flag | flag identifier | Mandatory |
| value | boolean flag value | Mandatory |

**Response**
HTTP Status 200

**Sample**

```
GET /json/property/setFlag?path=/system/host/interfaces/lo/mac\&flag=WRITEABLE\&value=true

{ "ok" : true }
```

### 10.1.11 remove

Removes a property node.

**Synopsis**
HTTP GET /json/property/remove

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| path | path of the property | Mandatory |

**Response**
HTTP Status 200

**Sample**

```
GET /json/property/remove?path=/testpath

{ "ok" : true }
```

## 10.2 Property Query

### 10.2.1 query

Returns a part of the tree specified by query. All queries start from the root.
The properties to be included have to be put in parentheses. A query to get
all device from zone4 would look like this: '/apartment/zones/zone4/*(ZoneID,name)'.
More complex combinations (see example below) are also possible.

**Synopsis**
HTTP GET /json/property/query

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| query | query string | Mandatory |

**Response**
HTTP Status 200

| result.value | string value of the property |
|---|---|

**Sample**

```
GET
      /json/property/query?query=/apartment/zones/{*}(ZoneID,scenes)/groups/{*}(group,name)/scenes/{*}(scene,name)

{
    "ok":true,
    "result":
    {
        "zones":
        [
            {
                "ZoneID":3663,
                "groups":
                [
                    {
                        "group":1,
                        "name":"yellow",
                        "scenes":
                        [
                            {
                                "scene":5,
                                "name":"demo␣scene"
                            }
                        ]
                    },
                    {
                        "group":2,
                        "name":"gray",
                        "scenes":[]
                    }
                ]
            }
        ]
    }
}
```

### 10.2.2 query2

Differs from query(1) only in the format of the the returned json struct.

**Synopsis**
HTTP GET /json/Property/query2?query=/Folder1(Property1,Property2)/Folder2(Property1)

HTTP GET /json/Property/query2?query=/Folder1/Folder2/Folder3(Property1)
HTTP GET /json/Property/query2?query=/Folder1/*(*)/*(*)/*(*)

*Folder* selects the nodes to descend, *Property* declares which attributes we are extracting from the current node. If no properties are declared for a folder, nothing is extracted, and the node will not show up in the resulting json structure.

**Parameter**

| Parameter | Description | Remarks |
|---|---|---|
| query | query string | Mandatory |

**Response**
HTTP Status 200

| result.value | string value of the property |
|---|---|

**Sample**

```
GET '/json/property/query2?query=/apartment/zones/*(scenes)/groups/*(name)/scenes/*(name)'


{
  "ok":true,
  "result":{
    "zone0":{
      ...
    },
    "zone6268":{
      "group0":{
        "name":"broadcast"
      },
      "group1":{
        "name":"yellow",
        "scene5":{
          "name":"dining"
        },
        "scene6":{
          "name":"TV"
        },
      },
      "group2":{
        "name":"gray",
        "scene6":{
          "name":"TV"
        },
        "scene17":{
          "name":"blinds␣15%"
        },
```

```
        },
      }
      ...
    }
}
```

The difference to query1 format is, that zones/groups/scenes are not returned as arrays of elements, but each element individually as a named property. This more closely matches the query format and facilitates accessing a specific element, e.g. zone6268.group1.scene6

Mind that the zones/groups folders are not part of the resulting json structure since no attributes is extracted from them. We could re-add them to the output using the wildcard (*) property match

Different from query1, we are not extracting the zoneid and scene name attribute, since that information is already contained in the element name. Neither are there any empty scene arrays. This makes the resulting json structure quite a bit smaller and easier read by a human. Potentially the json structure uses less memory is faster to generate, transfer, parse and render by a web application