ezlo innovation

mios vera fortrezz centralite

Table of contents

ezlo innovation
mios vera fortrezz centralite

API guide

This guide is designed to explain how to set up the Linux based controller to network control and how to control it through API calls.

First steps

To start controlling the Hub locally is required to register it to the network, to do, after connecting the Hub to power, the user must:

- Connect to Hub
- Query the list of available networks
- Connect the Hub to an available network
- Query connection status
- Close connection with the Hub
- Open WebSocket connection
- Change access point configuration through WeSocket connection

Find controller in the local network

Linux controllers use mDNS protocol for broadcasting his main information in the local network. You can use avahi-browse for searching controllers in your network:

```
avahi-browse _ezlo._tcp --resolve
```

Result will be like that:
= enp0s25 IPv4 eZLO g150 controller (46154962) _ezlo._tcp local
hostname = [HUB46154962.local]
address = [192.168.11.133]
port = [17000]
txt = ["Hub Type=g150" "Vendor=eZLO" "Firmware Version=1.0.13" "Serial=46154962"]
address - it's ip of controller in your network
port - port for connecting to the controller
txt.Hub Type - type of controller
txt.Serial - it's serial number of your controller

Open WebSocket connection

Once the IP is known is possible to control the Hub with the calls described in this guide, using a WebSocket client configured in port 17000

Messages structure

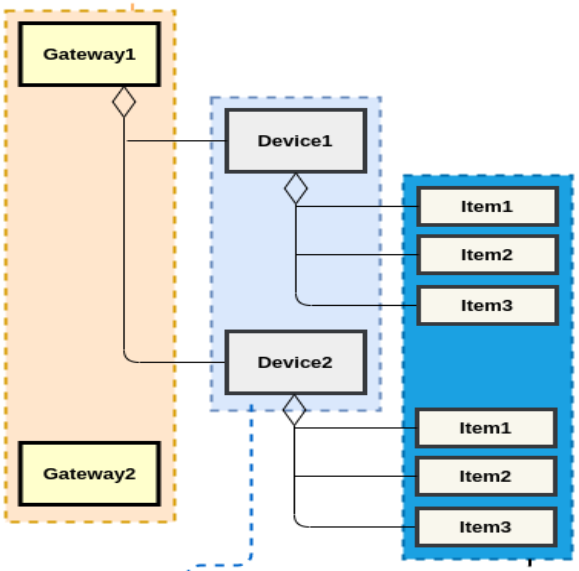
All interaction between the client and Hub must be done exchanging messages in JSON format, and the content will depend on the sender of the message:

- All messages sent by the client to the Hub will contain a “method” parameter to identify the intention of the client. The common methods used to interact with the Hub will be defined in this guide.
- All messages sent by the client to the Hub will contain an “id” parameter to identify the reply of the Hub. This parameter is any string defined by the client just to match the reply with the request performed.
- All messages generated by the Hub without any request, or as part of interaction for some processes will have in its body the key-value "id": "ui_broadcast". These messages can be seen when the Hub is periodically reporting the state of devices linked or actions triggered by the user e.g. temperature, motion detected, system errors, pairing/remove flow, etc…

General considerations

To properly understand the content of this guide there are some considerations to keep in mind:

- The Hub exposes three kinds of components:
    - Gateways - list of supported protocols:
    - Devices: Represent physical components of hardware.
    - Items: represent the minimum unit of interaction with the Hub that are mapped as devices or services.



Implementation of smart protocol(ZWave, BLE, etc)

Requests:

New Name	Description
hub.gateways.list	get all gateways on hub

Events:

New Name	Description
hub.gateway.added	New gateway was installed
hub.gateway.removed	Gateway was removed
hub.gateway.updated	Gateway property was changed

Fields:

Name	Description
_id	Unique id of gateway within the hub
name	Unique name of gateway within the hub
ready	Could be new operation executed or not
pluginId	Id of parent plugin
operations	Supported operation(adding and removing device)

Smart Device

Example:

Device Type	Description
light.bulb	Light Bulb
dimmer.inwall	In-wall Dimmer

Requests:

New Name	Description
hub.devices.list	get all devices on the hub
hub.device.name.set	Rename device

Events:

New Name	Description
hub.device.added	Device was added
hub.device.removed	Device was removed
hub.device.updated	Device property was changed

Example of fields:

Name	Description
_id	Unique id of device within the hub
gatewayId	Id of parent gateway
name	Name of device
batteryPowered	Is it battery device or not

Minimum unit of device functionality.

Example:

Item Name	Description
switch	Switch on / Switch Off device
thermostat_mode	Thermostat mode
electric_meter_amper	Amper Meter

Requests:

New Name	Description
hub.items.list	get all items on hub
hub.item.value.set	set value for item

Events:

New Name	Description
hub.item.added	Item was added
hub.item.removed	Item was removed
hub.item.updated	Item property was changed

Fields:

Name	Description
deviceId	Id of parent device
name	Name of item
value	Value of item

Add a ZWave device

The Hub supports the devices listed on the compatibility list. Other devices out of the list may be added following the same procedure, however its behaviour is not guaranteed.

To add a device it is necessary to put the Hub in inclusion mode, this is done with the following call:

```
{
   "method": "hub.extensions.plugin.run",
   "id": "_ID_",
   "params": {
       "script": "HUB:zwave/scripts/start_include"
   }
}
```

The following message should appear to indicate the inclusion mode state:

```
Answer received: { "method": "hub.extensions.plugin.run", "result": {}, "error": null,
"id": "_ID_", "sender": { "conn_id": "aa136e6f-ac1d-40da-824f-fd3bc6ec946c", "type":
"ui" } }

Answer received: { "id": "ui_broadcast", "msg_subclass":
"hub.extensions.plugin.ui_broadcast", "result": { "event": "include_invoked",
"plugin": "zwave" } }
```

Once “include_invoked” event appears is time to put the device in inclusion mode as well. This configuration is unique for each device and must be provided by the manufacturer.

The Hub will start an exchange of commands with the device until the process is done.

The following code snippet is an example of the messages that will appear during the pairing process, they may be different for each device:

```
Answer received: { "id": "ui_broadcast", "msg_subclass":
"hub.extensions.plugin.ui_broadcast", "result": { "event": "include_started",
"plugin": "zwave" } }

Answer received: { "id": "ui_broadcast", "msg_subclass": "hub.device.added", "result":
{ "_id": "Z6622669B", "deviceTypeId": "134_259_75", "parentDeviceId": "", "category":
```

```
"switch", "subcategory": "interior_plugin", "gatewayId": "zwave", "name": "Smart
Switch Gen5", "type": "dimmer.outlet", "batteryPowered": false, "reachable": true,
"persistent": false, "serviceNotification":false, "roomId": "" } }

Answer received: { "id": "ui_broadcast", "msg_subclass": "hub.item.added", "result": {
"_id": "switch475F9370", "deviceId": "Z6622669B", "hasGetter": true, "hasSetter":
true, "name": "switch", "show": true, "valueType": "bool", "value": false } }

Answer received: { "id": "ui_broadcast", "msg_subclass": "hub.item.added", "result": {
"_id": "electric_meter_kwh643D9171", "deviceId": "Z6622669B", "hasGetter": true,
"hasSetter": true, "name": "electric_meter_kwh", "show": true, "valueType": "float",
"value": 0 } }

Answer received: { "id": "ui_broadcast", "msg_subclass": "hub.item.added", "result": {
"_id": "electric_meter_watt72DC7934", "deviceId": "Z6622669B", "hasGetter": true,
"hasSetter": true, "name": "electric_meter_watt", "show": true, "valueType": "float",
"value": 0 } }

Answer received: { "id": "ui_broadcast", "msg_subclass": "hub.item.added", "result": {
"_id": "electric_meter_volt1FF3A209", "deviceId": "Z6622669B", "hasGetter": true,
"hasSetter": true, "name": "electric_meter_volt", "show": true, "valueType": "float",
"value": 0 } }

Answer received: { "id": "ui_broadcast", "msg_subclass": "hub.item.added", "result": {
"_id": "electric_meter_amper722354A4", "deviceId": "Z6622669B", "hasGetter": true,
"hasSetter": true, "name": "electric_meter_amper", "show": true, "valueType": "float",
"value": 0 } }

Answer received: { "id": "ui_broadcast", "msg_subclass": "hub.item.added", "result": {
"_id": "meter_reset5A0037E9", "deviceId": "Z6622669B", "hasGetter": false,
"hasSetter": false, "name": "meter_reset", "show": true, "valueType": "float",
"value": 0 } }

Answer received: { "id": "ui_broadcast", "msg_subclass":
"hub.extensions.plugin.ui_broadcast", "result": { "event": "include_finished",
"plugin": "zwave" } }
```


At the end of the process, an event message of "***include_finished***" should indicate that everything went well, in the case of "***include_finished_timeout***" the process must be restarted.

In case of failure during the process, the Hub will send an error message (like "***include_finished_error***"). In this case, the remove sequence must be applied to the device and try again the add device process.

Remove a device

In order to remove a device, both, Hub and device must be in exclusion mode. For the Hub the following call must be performed:

```
{
   "method": "hub.extensions.plugin.run",
   "id": "_ID_",
   "params": {
       "script": "HUB:zwave/scripts/start_exclude"
   }
}
```

The following messages must appear to indicate that exclusion process begin:

```
Answer received: { "method": "hub.extensions.plugin.run", "result": {}, "error": null,
"id": "_ID_", "sender": { "conn_id": "aa136e6f-ac1d-40da-824f-fd3bc6ec946c", "type":
"ui" } }
Answer received: { "id": "ui_broadcast", "msg_subclass":
"hub.extensions.plugin.ui_broadcast", "result": { "event": "exclude_invoked",
"plugin": "zwave" } }
```

Then the device must be set on exclusion mode. This must be specified in the user/installation guide provided by the manufacturer.

At the end of the process the Hub must confirm that exclusion ended as expected:

```
Answer received: {
   "id": "ui_broadcast",
   "msg_subclass": "hub.extensions.plugin.ui_broadcast",
   "result": {
       "event": "exclude_finished",
       "plugin": "zwave"
   }
}
```

In case of error event messages, the process must be restarted.

Get devices list

The following call allows querying about the devices added to the Hub. Some devices are multi-sensors and may appear as several devices for a single hardware piece:

```
{
   "method": "hub.devices.list",
   "id": "_ID_",
   "params": {}
}
```

The Hub will reply with the following structure (as big as devices added):

```
Answer received: {
   "method": "hub.devices.list",
   "result": {
       "devices": [
           {
               "_id": "ZFD0894A6",
               "deviceTypeId": "134_259_75",
               "parentDeviceId": "",
               "category": "switch",
               "subcategory": "interior_plugin",
               "gatewayId": "zwave",
               "name": "Switch 1",
               "type": "dimmer.outlet",
               "batteryPowered": false,
               "reachable": true,
               "persistent": false,
               "serviceNotification": false,
               "roomId": "0"
           }
       ]
   },
   "error": null,
   "id": "_ID_",
   "sender": {
       "conn_id": "aa136e6f-ac1d-40da-824f-fd3bc6ec946c",
       "type": "ui"
   }
```

```
}
```

Get items list

Provides a list of registered items on the Hub:

```
{
   "method": "hub.items.list",
   "id": "_ID_",
   "params": {}
}
```

The Hub will reply with the list of items:

```
Answer received: {
   "method": "hub.items.list",
   "result": {
       "items": [
           {
               "_id": "switchDB1FCA84",
               "deviceId": "ZFD0894A6",
               "hasGetter": true,
               "hasSetter": true,
               "name": "switch",
               "show": true,
               "valueType": "bool",
               "value": true
           },
           {
               "_id": "electric_meter_kwhD03F7BB4",
               "deviceId": "ZFD0894A6",
               "hasGetter": true,
               "hasSetter": true,
               "name": "electric_meter_kwh",
               "show": true,
               "valueType": "float",
               "value": 0
           }
       ]
   },
   "error": null,
   "id": "_ID_",
```

```
    "sender": {
        "conn_id": "8e6f7eee-aea4-480e-9a02-b6ac3d7a9804",
        "type": "ui"
    }
}
```

This call is required to understand the structure of the items in the Hub, the following information can be exposed:

Field	Type	Required	Description
_id	string	yes	id of the item
deviceId	string	yes	id of a device this item belongs to
enum	array	no	Finite array of possible token values
hasGetter	bool	yes	Whether the item provides an ability to get a value
hasSetter	bool	yes	whether the item provides an ability to set a value
name	string	yes	A name(type) of the item
show	bool	yes	Whether to show the item (on the UI) or not
scale	string	no	A name of measurement units
valueType	string	yes	A type of an item's value
valueFormatted	string	yes	An item formatted value
value	object	yes	An item value
valueMin	object	no	Lower limit of item's value field
valueMax	object	no	Upper limit of item's value field

Send commands to items

The following call allows to change the state of the items:

```
{
    "method": "hub.item.value.set",
    "id": "_ID_",
    "params": {
        "_id": "switchDB1FCA84",
```

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

The Hub will reply on success:

```
Answer received: {
    "method": "hub.item.value.set",
    "result": {},
    "error": null,
    "id": "_ID_",
    "sender": {
        "conn_id": "8e6f7eee-aea4-480e-9a02-b6ac3d7a9804",
        "type": "ui"
    }
}
Answer received: {
    "id": "ui_broadcast",
    "msg_subclass": "hub.item.updated",
    "result": {
        "_id": "switchDB1FCA84",
        "deviceId": "ZFD0894A6",
        "deviceName": "Switch 1",
        "deviceCategory": "switch",
        "deviceSubcategory": "interior_plugin",
        "serviceNotification": false,
        "roomName": "",
        "userNotification": false,
        "notifications": null,
        "name": "switch",
        "value": true
    }
}
```

Otherwise will reply an error in case of bad item requested or network error.

Get house modes

The Hub implement several house modes to apply a group of configuration to all devices with a single call, to know what house modes are implemented and get details about them the following call must be performed:

```
{
   "method": "hub.modes.get",
   "id": "_ID_",
   "params": {}
}
```

The Hub will reply with the modes:

```
Answer received: {
   "method": "hub.modes.get",
   "result": {
       "current": "1",
       "switchTo": "",
       "switchToDelay": 0,
       "modes": [
           {
               "_id": "1",
               "name": "Home",
               "description": "",
               "switchToDelay": 0,
               "alarmDelay": 0,
               "notifications": null,
               "disarmedDefault": true,
               "disarmedDevices": [],
               "alarmsOffDevices": []
           },
           {
               "_id": "2",
               "name": "Away",
               "description": "",
               "switchToDelay": 30,
               "alarmDelay": 30,
               "notifications": null,
               "disarmedDefault": true,
               "disarmedDevices": [],
               "alarmsOffDevices": []
           },
           {
               "_id": "3",
               "name": "Night",
               "description": "",
               "switchToDelay": 30,
```

```
                "alarmDelay": 30,
                "notifications": null,
                "disarmedDefault": true,
                "disarmedDevices": [],
                "alarmsOffDevices": []
            },
            {
                "_id": "4",
                "name": "Vacation",
                "description": "",
                "switchToDelay": 30,
                "alarmDelay": 30,
                "notifications": null,
                "disarmedDefault": true,
                "disarmedDevices": [],
                "alarmsOffDevices": []
            }
        ],
        "devices": [],
        "alarms": []
    },
    "error": null,
    "id": "_ID_",
    "sender": {
        "conn_id": "8e6f7eee-aea4-480e-9a02-b6ac3d7a9804",
        "type": "ui"
    }
}
```


The possible fields on the reply are the following:

Field	Type	Description
current	string	Id of the current mode
switchTo	string	Id of the next mode (after switch to) or empty
switchToDelay	integer	Delay (sec) before switch to the mod
modes	JsonArray	Array of the houseModes entries
modes._id	string	Id of the mode
modes.name	string	Name of the mode
modes.description	string	Brief description of the mode

modes.notifications	JsonArray	Array of user IDs or null if need notify all user IDs
modes.disarmedDefault	bool	Use default (not editable) disarmed list or custom
modes.disarmedDevices	JsonArray	Array of disarmed device id (current)
modes.alarmsOffDevices	JsonArray	Array of alarmsOff device id (current)
devices	JsonArray	Array of device id with security sensors
alarms	JsonArray	Array of device id which make alarms after trips

Get current house mode

The actual house mode can be queried with the call:

```
{
   "method": "hub.modes.current.get",
   "id": "_ID_",
   "params": {}
}
```

The reply will be like:

```
Answer received: {
   "method": "hub.modes.current.get",
   "result": {
       "modeId": "1"
   },
   "error": null,
   "id": "_ID_",
   "sender": {
       "conn_id": "8e6f7eee-aea4-480e-9a02-b6ac3d7a9804",
       "type": "ui"
   }
}
```

Change house mode

To change the house mode the following call must be done with the id of the mode to set:

```
{
   "method": "hub.modes.switch",
   "id": "_ID_",
```

```
    "params": {
        "modeId": "4"
    }
}
```

The Hub will reply with the delay defined to change to the requested mode and will confirm the mode after the given time:

```
Answer received: {
    "method": "hub.modes.switch",
    "result": {
        "switchToDelay": 30
    },
    "error": null,
    "id": "_ID_",
    "sender": {
        "conn_id": "8e6f7eee-aea4-480e-9a02-b6ac3d7a9804",
        "type": "ui"
    }
}
```

Create room

Room creation allows to group devices according user needs, they can be created with this call:

```
{
    "method": "hub.room.create",
    "id": "_ID_",
    "params": {
        "name": "Test room"
    }
}
```

Get list of rooms

The existing rooms can be retrieved with the call:

```
{
    "method": "hub.room.list",
    "id": "_ID_",
    "params": {}
}
```

The reply will be the list of rooms with their IDs for reference:


```
Answer received: {
   "method": "hub.room.list",
   "result": [
       {
           "_id": "537BD0A2",
           "name": "Bedroom"
       },
       {
           "_id": "D50737BC",
           "name": "Living room"
       },
       {
           "_id": "3BC25F49",
           "name": "Test room"
       }
   ],
   "error": null,
   "id": "_ID_",
   "sender": {
       "conn_id": "8e6f7eee-aea4-480e-9a02-b6ac3d7a9804",
       "type": "ui"
   }
}
```


Delete room

The following call with remove the given room, all devices assigned to it will be marked as “no room” but will keep working as usual:


```
{
   "method": "hub.room.delete",
   "id": "_ID_",
   "params": {
       "_id": "D50737BC"
   }
}
```

Assign device to a room

To add a device (not valid for items) is important to know the id values of device and room, and then perform the following call:

```
{
   "method": "hub.device.room.set",
   "id": "_ID_",
   "params": {
       "_id": "ZFD0894A6",
       "roomId": "3BC25F49"
   }
}
```

The reply will be like:

```
Answer received: {
   "method": "hub.device.room.set",
   "result": {},
   "error": null,
   "id": "_ID_",
   "sender": {
       "conn_id": "8e6f7eee-aea4-480e-9a02-b6ac3d7a9804",
       "type": "ui"
   }
}
```

Remove network information

In case of require remove the network information, the following call can be performed:

```
{
   "method": "hub.network.reset",
   "id": "_ID_",
   "params": {}
}
```

This will clean all the network information and will close the WebSocket communication, the initial setup must be performed to use the unit locally again.

Use case example

In the following example is explained the interaction with the Hub in order to get the information from the meters of connected plug. Here the steps to follow:

- Query the list of devices
- Query the list of items
- Check metering items

Query the list of devices

First thing to do is get the ***_id*** of the plug connected to the Hub to properly identify its items where metering services are defined, to do so the get devices list call must be performed. The plug is easy to identify within the list of devices since it has specific values for ***deviceTypeId, gatewayId*** and ***name*** parameters:


```
{
   "_id": "U214912D2",
   "deviceTypeId": "plug",
   "parentDeviceId": "",
   "category": "switch",
   "subcategory": "in_wall",
   "gatewayId": "plug",
   "name": "Plug Switch",
   "type": "switch.inwall",
   "batteryPowered": false,
   "reachable": true,
   "persistent": true,
   "serviceNotification": true,
   "roomId": "",
   "security": ""
}
```


Query the list of items

Once the id of the device is known (in the example case is "**U214912D2**"), all items needs to be queried, this is done using the get items list call

Check metering items

The current version of the API retrieves all items present in the Hub, for this reason is important to know the id of the device implementing the services:


```
{
   "_id": "electric_meter_watt9BE94673",
   "deviceId": "U214912D2",
   "hasGetter": true,
   "hasSetter": true,
   "name": "electric_meter_watt",
   "show": true,
   "valueType": "float",
   "value": 0
},
{
   "_id": "electric_meter_kwhCD5ADB9F",
   "deviceId": "U214912D2",
   "hasGetter": true,
   "hasSetter": true,
   "name": "electric_meter_kwh",
   "show": true,
   "valueType": "float",
   "value": 0
},
{
   "_id": "electric_meter_voltB3E61956",
   "deviceId": "U214912D2",
   "hasGetter": true,
   "hasSetter": true,
   "name": "electric_meter_volt",
   "show": true,
   "valueType": "float",
   "value": 0
},
{
   "_id": "electric_meter_amper92C39CFC",
   "deviceId": "U214912D2",
   "hasGetter": true,
   "hasSetter": true,
   "name": "electric_meter_amper",
   "show": true,
   "valueType": "float",
   "value": 0
}
```

Worths to mention that this is the way of how to get the values of items, in this case metering values, at any desired time. However the Hub will report any change of items without querying them as a broadcast message:


```
{
   "id": "ui_broadcast",
   "msg_subclass": "hub.item.updated",
   "result": {
       "_id": "electric_meter_watt9BE94673",
       "deviceId": "U214912D2",
       "deviceName": "Plug Switch",
       "deviceCategory": "switch",
       "deviceSubcategory": "in_wall",
       "serviceNotification": false,
       "roomName": "Test room",
       "userNotification": false,
       "notifications": null,
       "name": "electric_meter_watt",
       "value": 0
   }
}
```


Scenes General information

Scenes provide the possibility to make the relations between devices and make some actions with them. Generally they are named as conditions and actions so this 2 blocks are: when and then.

When blocks

When block currently supports one or several events (conditions) and these blocks are connected by OR logical operators by default.

Field	Type	Required	Description
blockOptions	JsonObject	+	Options of the block

blockOptions.method	JsonObject	+	Json representation of the function for triggering
blockOptions.method.args	JsonObject	+	Json object with the names of the fields that must be extracted from the `fields` list
blockOptions.method.name	string	+	Descibes the event type. Possible values: see below
blockType	string	+	Name of the block type. Should be set as "when"
fields	JsonArray	+	Array of the triggers. There is used the same format as it is in the then block but item address, values etc
fields[].type	Enum	+	Represents the Item Value Type
fields[].value	Any Json Value	+	actual value corresponding the field name

isItemState

This events arises when the value of item is equal to the value is set in this when block. Optionally it checks device armed status by logical AND operator with isItemState condition.

Field	Type	Required	Description

blockOptions.method.args.item	string	+	Argument declaration of item ID. The value should be in `field` block with name `item`.
blockOptions.method.args.value	string	+	Argument declaration of item value. The value should be in `field` block with name `value`.
blockOptions.method.args.armed	string	-	Argument declaration of armed state of device is corresponding to current item. The value should be in `field` block with name `armed`. This adds optional condition checks device armed status and is connected with isItemState condition by logical AND operator.

Example:

```
"when" : [{
   "blockOptions":{
      "method":{
         "args":{
            "item":"item",
            "value":"value",
            "armed":"armed"
         },
         "name":"isItemState"
      }
   },
   "blockType":"when",
   "fields":[
      {
          "name":"item",
          "type":"item",
          "value" : "35656_5656_56"
      },
      {
          "name":"value",
```

```
            "type":"int",
            "value":0
        },
        {
            "name":"armed",
            "type":"bool",
            "value":true
        }
    ]
}]
```

compareNumbers

This event arises when the value of item is corresponded to condition is set in this block. For example, if the comparator is ==, value is equal to 50 and item value is 50 then event arises. If condition is >50 then event arises only once when threshold was exceeded. For example, if item value is 49 and after that item value becomes 51 then event arises. When item value becomes 52 the event doesn't arise. The event will arise again when threshold was exceeded again. Similar situation is for other comparators (>=, <, <=).

Field	Type	Required	Description
blockOptions.method.args.item	string	+	Argument declaration of item ID. The value should be in `field` block with name`item`.
blockOptions.method.args.value	string	+	Argument declaration of item value. The value should be in `field` block with name`value`.
blockOptions.method.args.comparator	string	+	Argument declaration of comparator state. The value should be in `field` block with name`comparator`. Possible comparators are `==`, `!=`, `>`, `>=`, `<`, `<=`.

Example:


```
"when" : [
    {
        "blockType":"when",
        "blockOptions":{
            "method":{
                "name":"compareNumbers",
                "args":{
                    "item":"item",
                    "comparator":"comparator",
                    "value":"value"
                }
            }
        },
        "fields":[
            {
                "name":"item",
                "type":"item",
                "value":"5de64f6a70c7be0541cc0853"
            },
            {
                "name":"comparator",
                "type":"string",
                "value": ">"
            },
            {
                "name":"value",
                "type":"int",
                "value": 51
            }
        ]
    },
    {
        "blockType":"when",
        "blockOptions":{
            "method":{
                "name":"compareNumbers",
                "args":{
                    "item":"item",
                    "comparator":"comparator",
                    "value":"value"
                }
            }
        },
        "fields":[
            {
```

```
                "name":"item",
                "type":"item",
                "value":"5de64f6a70c7be0541cc0854"
            },
            {
                "name":"comparator",
                "type":"string",
                "value": "<="
            },
            {
                "name":"value",
                "type":"float",
                "value": 51.55
            }
        ]
    }
]
```

isInterval

Periodically fires the list of actions

Example:

```
"when" : [{
   "blockOptions":{
      "method":{
         "args":{
            "interval":"interval"
         },
         "name":"isInterval"
      }
   },
   "blockType":"when",
   "fields":[
      {
         "name":"interval",
         "type":"interval",
         "value":"212s"
      }
   ]
}]
```

isSunState

Fires the actions corresponding sunset/sunrise event.it's possible to set the special days of the week or days of the month. Also timeoffset could be used. For that field before/after must be set.

Sunstate	Description
sunrise	The possible values: `intime`, `before` or `after`
sunset	The possible values: `intime`, `before` or `after`

Example:

```
"when" : [{
   "blockOptions":{
      "method":{
         "args":{
            "sunstate":"sunrise",
            "time":"time"
         },
         "name":"isSunState"
      }
   },
   "blockType":"when",
   "fields":[
      {
         "name":"sunrise",
         "type":"string",
         "value":"before"
      },
      {
         "name":"time",
         "type":"hms_interval",
         "value":"10:30"
      }
   ]
}]
```

Logic operators

and

The AND logic operator is when block. This condition is true in case when all conditions in blocks array are true also. The AND operation could contain a different when blocks except some restictions are described below. The AND operator could contain a nested logic operators.

Field	Type	Required	Description
blockOptions.method.args.blocks	string	+	The argument declaration of `blocks` field. The name is "blocks". The type is "blocks". The `blocks` field could contain several when blocks. If all contained blocks are `true` then this block is also `true` otherwise it is `false`.

Examples:

```
{
   "blockType":"when",
   "blockOptions":{
      "method":{
         "name":"and",
         "args":{
            "blocks":"blocks"
         }
      }
   },
   "fields":[
      {
         "name":"blocks",
         "type":"blocks",
         "value": [
           {
               __WHEN_BLOCK__
           },
           {
```

```
                __WHEN_BLOCK__
            },
            {
                "blockType":"when",
                "blockOptions":{
                   "method":{
                      "name":"and",
                      "args":{
                         "blocks":"blocks"
                      }
                   }
                },
                "fields":[
                   {
                      "name":"blocks",
                      "type":"blocks",
                      "value": [
                        {
                            __WHEN_BLOCK__
                        },
                        {
                            __WHEN_BLOCK__
                        }
                      ]
                   }
                ]
            }
         ]
      }
   ]
}
```

not

The NOT logic operator is when block. This condition is true if contained condition is false otherwise if contained condition is true then it is false. The NOT operation could contain any when block. The NOT operator could contain a nested logic operator.

Field	Type	Required	Description

blockOptions.method.args.block	string	+	The argument declaration of `block` field. The name is "block". The type is "block". The `block` field could contain when only one block. This condition is `true` if contained condition is `false` otherwise if contained condition is `true` then it is `false`.

Examples:

```
{
   "blockType":"when",
   "blockOptions":{
      "method":{
         "name":"not",
         "args":{
            "block":"block"
         }
      }
   },
   "fields":[
      {
         "name":"block",
         "type":"block",
         "value": {
             "blockType":"when",
             "blockOptions":{
                 "method":{
                     "name":"not",
                     "args":{
                         "block":"block"
                     }
                 }
             },
             "fields":[
                 {
                     "name":"block",
                     "type":"block",
                     "value": {
                         __WHEN_BLOCK__
                     }
                 }
             ]
         }
```

```
        }
    ]
}
```

or

The OR logic operator is when block. This condition is true if any contained condition is true otherwise if all contained condition is false then it is also false. The OR operation could contain several when blocks. The OR operator could contain a nested logic operators.

Field	Type	Required	Description
blockOptions.method.args.blocks	string	+	The argument declaration of `blocks` field. The name is "blocks". The type is "blocks". The `blocks` field could contain several when blocks. If any contained block is `true` then this block is also `true` otherwise it is `false`.

Examples:

```
{
   "blockType":"when",
   "blockOptions":{
      "method":{
         "name":"or",
         "args":{
            "blocks":"blocks"
         }
      }
   },
   "fields":[
      {
         "name":"blocks",
         "type":"blocks",
         "value": [
           {
               __WHEN_BLOCK__
           },
           {
               __WHEN_BLOCK__
           },
```

```
            {
                "blockType":"when",
                "blockOptions":{
                   "method":{
                      "name":"or",
                      "args":{
                         "blocks":"blocks"
                      }
                   }
                },
                "fields":[
                   {
                      "name":"blocks",
                      "type":"blocks",
                      "value": [
                        {
                            __WHEN_BLOCK__
                        },
                        {
                            __WHEN_BLOCK__
                        }
                      ]
                   }
                ]
             }
          ]
       }
    ]
}
```

Then blocks

then block currently supports one or several actions and execution of these actions is provided in order is set in array of then block.

Field	Type	Required	Description
blockOptions	JsonObject	+	Action block options
blockOptions.method	JsonObject	+	Action description

blockOptions.method.args	JsonObject	+	Action description arguments that should be read from `field` attribute
blockOptions.method.name	string	+	Name of action
fields	JsonArray	+	Array of fields that must be extracted by the `blockOptions.method.args` names
fields[].name	string	+	Name of the field block corresponding of the declaration in `blockOptions.method.args`
fields[].type	string	-	Represents the Item Value Type
fields[].value	string	+	Value should be corresponded to the `fields[].type`
delay	JsonObject	-	Delay to action after event arises. If this field is absent or is empty then delay is absent.
delay.seconds	int	-	Seconds number of delay
delay.minutes	int	-	Minutes number of delay
delay.hours	int	-	Hours number of delay

delay.days	int	-	Days number of delay

setItemValue

Set the value for the specific item.

Field	Type	Required	Description
blockOptions.method.args.item	string	+	Argument declaration of of Item ID. The name is `item`. The type is `item`.
blockOptions.method.args.value	string	+	Argument declaration of Value should be set to item when event arises. The name is `value`. The type is `value`.

Examples:

```
"then" : [{
   "blockOptions":{
      "method":{
         "args":{
            "item":"item",
            "value":"value"
         },
         "name":"setItemValue"
      }
   },
   "blockType":"then",
   "delay" : {
       "seconds": 12,
       "minutes": 30,
       "hours": 1,
       "days": 0
   },
   "fields":[
      {
         "name":"item",
```

```
            "type":"item",
            "value" : "897607_32771_1"
        },
        {
            "name":"value",
            "type":"int",
            "value": 10
        }
    ]
}]
```

Examples of possible values:

```
{
    "name":"value",
    "type":"bool",
    "value": false
}
{
    "name":"value",
    "type":"token",
    "value": "idle_off"
}
{
    "name":"value",
    "type":"power",
    "value": 30,
    "scale": "watt"
}
{
    "name":"value",
    "type": "float",
    "value": 5.0
}
{
    "name":"value",
    "type": "string",
    "value": "example"
}
```

Scenes commands

hub.scenes.create

Create a new scene.

Field	Type	Required	Description
enabled	boolean	+	enable or disable scene
group_id	string	-	group identifier, Scenes could be unite to the group for enabling/disabling
name	string	+	scene name. Maximum name length is 25 characters.
parent_id	string	+	room identifier for that it was created
then	JsonArray	+	Array of the `actions` blocks
when	JsonArray	+	Array of the `conditions` blocks
user_notifications	JsonArray	-	Array of the user IDs for notification broadcasts making. This is array of strings.
house_modes	StringArray	-	Array of house mode IDs. If this array is added then it makes new condition along with `when` block and this condition is

	connected with `when` block by logical AND operator. So if one of `conditions` arises and one of these house modes is set then `actions` will be executed.

broadcasts:

Broadcasts	Description
hub.scene.added	Broadcast when the scene is successfully created.
hub.scene.run.progress	Notification about the scene status. It's fired when scene is started, finished or failed.

errors:

Code	Message	Data
-32600	Bad request, name is empty	`rpc.params.empty.name`
-32600	Bad request, name does not exist	`rpc.params.notfound.name`
-32500	Scene is ill formed. Can't parse when block	`ezlo.scenes.block.when.wrong`
-32500	Scene is ill formed. Can't parse then block	`ezlo.scenes.block.then.wrong`
-32500	Scene is failed. There is no such method	`ezlo.scenes.method.unknown`

-32500	Scene contain conditions for not intersect numbers values inside of AND condition	`scenes.when.not_intersect_numbers`
-32500	Scene contain conditions for same functionality inside of AND condition	`scenes.when.same_button_in_and`
-32500	Scene contain conditions for same functionality inside of AND condition	`scenes.when.same_item_in_and`
-32500	Scene cannot contain more than one "time" condition in the same AND operator	`scenes.when.more_than_one_time`

return result fields:

Empty result or an error.

Here is it an example of usage:

call:

```
{
    "id": "_ID_",
    "jsonrpc": "2.0",
    "method": "hub.scenes.create",
    "params": {
        "enabled": true,
        "group_id": null,
        "is_group": false,
        "name": "testRule",
        "parent_id": "5c6ec961cc01eb07f86f9dd9",
        "user_notifications" : [
            "324234234",
            "456456453",
            "678678678"
        ],
        "house_modes" : [
            "1",
            "2",
            "4"
        ],
```

```
        "then" : [
            {
                "blockOptions":{
                    "method":{
                        "args":{
                            "item":"item",
                            "value":"value"
                        },
                        "name":"setItemValue"
                    }
                },
                "blockType":"then",
                "fields":[
                    {
                        "name":"item",
                        "type":"item",
                        "value" : "897607_32771_1"
                    },
                    {
                        "name":"value",
                        "type":"int",
                        "value": 10
                    }
                ]
            }
        ],
        "when": [
            {
                "blockOptions": {
                    "method": {
                        "args": {
                            "item": "item",
                            "value": "value"
                        },
                        "name": "isItemState"
                    }
                },
                "blockType": "when",
                "fields": [
                    {
                        "name": "item",
                        "type": "item",
                        "value": "5c7fea6b7f00000ab55f2e55"
                    },
                    {
                        "name": "value",
                        "type": "bool",
```

```
                        "value": true
                    }
                ]
            }
        ]
    }
}
```

reply:

```
{
    "error": null,
    "id": "_ID_",
    "result": {}
}
```

This is another example of the creation `interval` scene:

```
{
    "id": "_ID_",
    "jsonrpc": "2.0",
    "method": "hub.scenes.create",
    "params": {
        "enabled": true,
        "group_id": null,
        "is_group": false,
        "name": "testRule",
        "parent_id": "5c6ec961cc01eb07f86f9dd9",
        "house_modes" : [
            "1",
            "2",
            "4"
        ],
        "then" : [
            {
                "blockOptions":{
                    "method":{
                        "args":{
                            "item":"item",
                            "value":"value"
                        },
                        "name":"setItemValue"
                    }
                },
                "blockType":"then",
                "fields":[
```

```
                    {
                        "name":"item",
                        "type":"item",
                        "value" : "897607_32771_1"
                    },
                    {
                        "name":"value",
                        "type":"int",
                        "value": 10
                    }
                ]
            }
        ],
        "when": [
            {
                "blockOptions": {
                    "method": {
                        "args": {
                            "interval": "interval"
                        },
                        "name": "isInterval"
                    }
                },
                "blockType": "when",
                "fields": [
                    {
                        "name": "interval",
                        "type": "interval",
                        "value": "10s"
                    }
                ]
            }
        ]
    }
}


{
    "error": null,
    "id": "_ID_",
    "result": {}
}
```

hub.scenes.list

Get scene json object.

call:

```
{
    "id": "_ID_",
    "jsonrpc": "2.0",
    "method": "hub.scenes.list",
    "params": {}
}
```

reply:

```
{
    "error": null,
    "id": "_ID_",
    "result": {
         "scenes":
        [
        {
            "_id": "5c7ff48b7f00002a07a408e3",
            "enabled": true,
            "group_id": null,
            "is_group": false,
            "name": "testRule",
            "parent_id": "5c6ec961cc01eb07f86f9dd9",
            "house_modes" : [
                "1",
                "2",
                "4"
            ],
            "then" : [
                {
                    "blockOptions":{
                        "method":{
                            "args":{
                                "item":"item",
                                "value":"value"
                            },
                            "name":"setItemValue"
                        }
                    },
                    "blockType":"then",
                    "fields":[
                        {
                            "name":"item",
                            "type":"item",
                            "value" : "897607_32771_1"
```

```
                        },
                        {
                            "name":"value",
                            "type":"int",
                            "value": 10
                        }
                    ]
                }
            ],
            "when": [
                {
                    "blockOptions": {
                        "method": {
                            "args": {
                                "item": "item",
                                "value": "value"
                            },
                            "name": "isItemState"
                        }
                    },
                    "blockType": "when",
                    "fields": [
                        {
                            "name": "item",
                            "type": "item",
                            "value": "5c7fea6b7f00000ab55f2e55",
                        },
                        {
                            "name": "value",
                            "type": "bool",
                            "value": true
                        }
                    ]
                }
            ]
        }
    ]
  }
}
```

hub.scenes.edit

Update the scene json by it's id.

Field	Type	Required	Description
_id	string	+	rule identifier
eo	JsonObject	+	Json object of the rule description. Please see hub.scenes.create

broadcasts:

Broadcasts	Description
hub.scene.changed	Updating the information about the scene.

return result fields:

Empty result or an error

errors:

Code	Message	Data
-32600	Bad request, name is empty	`rpc.params.empty.name`
-32600	Bad request, name does not exist	`rpc.params.notfound.name`
-32500	Scene is ill formed. Can't parse when block	`ezlo.scenes.block.when.wrong`

-32500	Scene is ill formed. Can't parse then block	`ezlo.scenes.block.then.wrong`
-32500	Scene is failed. There is no such method	`ezlo.scenes.method.unknown`
-32500	Scene contain conditions for not intersect numbers values inside of AND condition	`scenes.when.not_intersect_numbers`
-32500	Scene contain conditions for same functionality inside of AND condition	`scenes.when.same_button_in_and`
-32500	Scene contain conditions for same functionality inside of AND condition	`scenes.when.same_item_in_and`
-32500	Scene cannot contain more than one "time" condition in the same AND operator	`scenes.when.more_than_one_time`

Here is an example of usage:

call:

```
{
    "id": "_ID_",
    "jsonrpc": "2.0",
    "method": "hub.scenes.edit",
    "params" :
     {
         "_id": "5c5318aa518af44041018347",
         "eo": {
             "_id": "5c5318aa518af44041018347",
             "enabled": true,
             "group_id": null,
             "is_group": false,
             "name": "NewR",
             "parent_id": "5c050abd518af4117b2e2496",
```

```
            "house_modes" : [
                "1",
                "2",
                "4"
            ],
            "then": [
                {
                    "blockOptions":{
                        "method":{
                            "args":{
                                "item":"item",
                                "value":"value"
                            },
                            "name":"setItemValue"
                        }
                    },
                    "blockType":"then",
                    "fields":[
                        {
                            "name":"item",
                            "type":"item",
                            "value" : "897607_32771_1"
                        },
                        {
                            "name":"value",
                            "type":"int",
                            "value": 10
                        }
                    ]
                },
                {
                    "blockOptions": {
                        "method": {
                            "args": {
                                "item": "item"
                            },
                            "name": "decreaseDimmer"
                        }
                    },
                    "blockType": "then",
                    "fields": [
                        {
                            "name": "item",
                            "type": "item",
                            "value": "897607_32771_1"
                        }
                    ]
```

```
                }
            ],
            "when": [
                {
                    "blockOptions": {
                        "method": {
                            "args": {
                                "item": "item",
                                "value": "value"
                            },
                            "name": "isItemState"
                        }
                    },
                    "blockType": "when",
                    "fields": [
                        {
                            "name": "item",
                            "type": "item",
                            "value": "897607_32770_1"
                        },
                        {
                            "name": "value",
                            "type": "bool",
                            "value": true
                        }
                    ]
                }
            ]
        },
        "permission": {
            "devices": "s",
            "ezlo": "s",
            "rules": "s",
            "ui": "s",
            "users": "s"
        },
        "sender": "_USER_",
        "serial": "_HUB_ID_"
    }
}
```

hub.scenes.delete

Delete the scene by it's id

Field	Type	Required	Description
_id	string	+	rule identifier

broadcasts:

Broadcasts	Description
hub.scene.deleted	Notification about the scene deleting.

return result fields:

Empty result or an error.

errors:

Code	Message	Data
-32600	Bad request, name is empty	`rpc.params.empty.name`
-32600	Bad request, name does not exist	`rpc.params.notfound.name`
-32500	The scene with this id does not exist	`ezlo.scenes.not.exist`

Here is it an example of usage:

```
{
    "id": "_ID_",
    "jsonrpc": "2.0",
    "method": "hub.scenes.delete",
    "params": {
        "_id": "5c7ff48b7f00002a07a408e3"
    }
}
```

reply:

```
{
    "error": null,
    "id": "_ID_",
    "result": {}
}
```

Hub.scenes.blocks.list

Getting possible conditional/action blocks related to the current device set on the hub for creating the scenes.

Field	Type	Required	Description
blockType	string	+	enumed literal value. Possible values : {"when", "then"}
devices	stringArray	+	The array of device IDs are used for filtering of items by device ID

return result fields: they are depend on the input param

Field	Type	Required	Description
when	JsonArray	+	Array of the possible `WHEN` blocks related to the current devices/items which are included. By them full set of rules is filtered

or

Field	Type	Required	Description

then	JsonArray	+	Array of the possible `THEN` blocks related to the current devices/items which are included. By them full set of rules is filtered

Here is it an example of usage:

```
{
    "id": "_ID_",
    "jsonrpc": "2.0",
    "method": "hub.scenes.blocks.list",
    "params": {
        "blockType": "when",
        "devices": [ "5dd2a8eebfb5be6d20008c55" ]
    }
}
```

reply:

```
{
    "error": null,
    "id": "_ID_",
    "result": {
        "when": [
            {
                "label": {
                    "lang_tag": "ui0_token",
                    "text": "English string"
                },
                "blockOptions":{
                    "method":{
                        "args":{
                            "item":"item",
                            "value":"value",
                            "armed": "armed"
                        },
                        "name":"isItemValue"
                    }
                },
                "blockType":"when",
                "fields":[
                    {
                        "name":"item",
```

```
                        "type":"item",
                        "value" : "5dd2a8efc1b5be6d20008c56"
                    },
                    {
                        "name":"value",
                        "type":"bool",
                        "options":[
                            {
                                "value":true,
                                "label": {
                                    "lang_tag": "ui1_token",
                                    "text": "Enable"
                                }
                            },
                            {
                                "value":false,
                                "label": {
                                    "lang_tag": "ui2_token",
                                    "text": "Disable"
                                }
                            }
                        ],
                        "value": true
                    },
                    {
                        "name":"armed",
                        "type":"bool",
                        "value":true
                    }
                ]
            },
            {
                "label":{
                    "lang_tag":"ui0_token",
                    "text":"English string"
                },
                "blockOptions": {
                    "method": {
                        "args": {
                            "item": "item",
                            "comparator":"comparator",
                            "value":"value"
                        },
                        "name": "compareNumbers"
                    }
                },
                "blockType": "when",
```

```
                "fields": [
                    {
                        "name": "item",
                        "type": "item",
                        "value": "897607_32771_2"
                    },
                    {
                        "name":"comparator",
                        "type":"string",
                        "options":[
                            {
                                "value":"==",
                                "label":{
                                    "lang_tag":"ui3_token",
                                    "text":"Equal"
                                }
                            },
                            {
                                "value":"!=",
                                "label":{
                                    "lang_tag":"ui4_token",
                                    "text":"Not equal"
                                }
                            },
                            {
                                "value":">",
                                "label":{
                                    "lang_tag":"ui5_token",
                                    "text":"Greater"
                                }
                            },
                            {
                                "value":"<",
                                "label":{
                                    "lang_tag":"ui6_token",
                                    "text":"Less"
                                }
                            },
                            {
                                "value":">=",
                                "label":{
                                    "lang_tag":"ui7_token",
                                    "text":"Greater and equal"
                                }
                            },
                            {
                                "value":"<=",
```

```
                                "label":{
                                    "lang_tag":"ui8_token",
                                    "text": "Less and equal"
                                }
                            }
                        ],
                        "value":"=="
                    },
                    {
                        "name":"value",
                        "type":"int",
                        "value": 10
                    }
                ]
            },
            {
                "label": {
                    "lang_tag": "ui0_token",
                    "text": "English string"
                },
                "blockOptions": {
                    "method": {
                        "args": {
                            "item": "item",
                            "value":"value"
                        },
                        "name": "isDictonaryValueState"
                    }
                },
                "blockType": "when",
                "fields": [
                    {
                        "name": "item",
                        "type": "item",
                        "value": "897607_32771_3"
                    },
                    {
                        "name":"value",
                        "type":"token",
                        "options":[
                            {
                                "value":"low_battery",
                                "label": {
                                    "lang_tag": "ui9_token",
                                    "text": "Low battery"
                                }
                            },
```

```
                                {
                                    "value":"not_detected",
                                    "label": {
                                        "lang_tag": "ui10_token",
                                        "text": "Not detected"
                                    }
                                }
                            ],
                            "value": "low_battery"
                        }
                    ]
                }
            ]
        }
}
```

Service methods

hub.reset

Hub supports two levels of reset: *Soft reset* and *Reset to factory defaults*

call

```
{
    "id": "_ID_",
    "method": "hub.reset",
    "params": {
        "softReset": false,
        "resetToFactoryDefaults": true
    }
}
}
```

Field	Type	Description
params.softReset	bool	*Soft reset*
params.resetToFactoryDefaults	bool	*Reset to factory defaults*

Only one field either softReset or resetToFactoryDefaults must be set to true. Only one field may be specified. If both fields are specified and both are set to true *Reset to factory defaults* will be executed.

reply

```
{
    "error": null,
    "id": "_ID_",
    "result": {}
}
```

hub.info.get

Common information about controller

parameters:

No parameters required.

return result fields:

Field	Type	Required	Description
model	string	+	Production name
architecture	string	+	SOC architecture
firmware	string	+	Firmware version
kernel	string	+	Kernel version
hardware	string	+	Hardware version

serial	string	+	Serial number
location	JsonObj	+	Build info
location.latitude	float32	+	Latitude value
location.longitude	float32	+	Longitude value
location.timezone	string	+	Time zone name.
location.state	string	+	Can contain one of possible values: `default`, `customAll`, `customTimezone`, `customCoordinates`
build	JsonObj	+	Build info
build.time	string	+	Time and date when the build was made (ISO 8601: YYYY-MM-DDThh:mm:ss±hhmm)
build.builder	string	+	Where the firmware was builded (user@host)
build.branch	string	+	Branch name

build.commit	string	+	Commit hash
uptime	string	+	Hub uptime
localtime	string	+	Current time on hub (ISO 8601: YYYY-MM-DDThh:mm:ss±hhmm)

call:

```
{
    "method": "hub.info.get",
    "id": "_ID_",
    "params": {}
}
```

reply:

```
{
    "error": null,
    "id": "_ID_",
    "result": {
        "model": "ATOM32",
        "architecture": "esp32",
        "firmware": "0.9.2",
        "kernel": "v3.3-dev-239-g18118a6d5",
        "hardware": "rev1",
        "serial": "0000001",
        "location": {
            "latitude": 50.5074,
            "longitude": 0.1278,
            "timezone": "Europe/London",
            "state": "custom"
        },
        "build": {
            "time": "2019-12-23T14:25:10+0200",
            "builder": "jenkins@builder1",
            "branch": "live",
            "commit": "cc2b9921c8572147d762674eacb9b02974ece302"
        },
        "uptime": "5d 22h 18m 11s",
        "localtime": "2004-05-23T14:25:10+0200"
```

```
    }
}
```

hub.software.info.get

Information about installed firwmare, addons, plugins

request:

no params

response:

Field	Type	Required	Description
firmware	string	+	Firmware version
addons	array	-	List of installed addons
addons[].id	string	+	Addon id (name)
addons[].version	string	+	Addon version
plugins	array	-	List of installed addons
plugins[].id	string	+	Plugin id (name)
plugins[].version	string	+	Plugin version

Example

request:

```
{
    "method": "hub.software.info.get",
```

```
    "id": "_ID_",
    "params": {}
}
```

response:

```
{
    "error": null,
    "id": "_ID_",
    "result": {
        "firmware": "0.9.1",
        "plugins": [
            {
                "id": "zwave",
                "version": "1.0.234"
            }
        ],
        "addons": [
            {
                "id": "zwave",
                "version": "1.0.8"
            }
        ]
    }
}
```

hub.firmware.luup.switch

Run process of switching from current firmware to native Vera firmware

call

```
{
    "id": "_ID_",
    "method": "hub.firmware.luup.switch",
    "params": {}
}
```

reply

```
{
    "error": null,
    "id": "_ID_",
    "result": {}

}
```

Broadcasts

hub.gateway.added

Broadcast sent when a gateway got registered on the hub (after plugin which provides this gateway has been installed).


```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.gateway.added",
    "result": {
      "_id": "588b7eb528b12d03be86f36f",
      "label": "ZWave",
      "name": "zwave",
      "pluginId": "zwave",
      "reason": "Start bus",
      "ready": true,
      "operations":{
          "deviceAdding":"ZWAVE:/start_include.template",
          "deviceRemoving":"ZWAVE:/start_exclude.template"
          "deviceSettings":[
              {
                  "label":"Parameters",
                  "template":"ZWAVE:/device/parameters_page.template"
              },
              {
                  "label":"Advanced",
                  "template":"ZWAVE:/device/advanced_page.template"
              }
          ],
      },
      "settings": [
          {
              "label":"General",
              "template":"ZWAVE:/settings/general_page.template"
          },
          {
              "label":"Advanced",
              "template":"ZWAVE:/settings/advanced_page.template"
          },
      ],
      "setItemValueCommand": "HUB:zwave/scripts/set_item_value"
    }
  }
```

Field	Type	Required	Description
_id	string	+	an id of the gateway
label	string	+	a public name of the gateway
name	string	+	a name specified within a plugin's config which provides this gateway
pluginId	string	+	an id (name, not a db's id) of a plugin this gateway is a part of
ready	bool	+	whether gateway is ready for work
operations	object	+	a gateway external API
settings	object	+	a section with custom setting templates for the gateway

---

hub.gateway.updated

Broadcast sent when some changes happened to a gateway (broadcast contains only changes).


```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.gateway.updated",
    "result": {
      "_id": "588b7eb528b12d03be86f36f",
```

```
        "__GATEWAY_CHANGEABLE_FIELD": value
      }
    }
```

Field	Type	Required	Description
_id	string	+	an id of the gateway
__GATEWAY_CHANGEABLE_FIELD	any	+	

Gateway fields which may be updated (__GATEWAY_CHANGEABLE_FIELD):

Field	Type	Description
ready	bool	whether gateway is ready for work

---

hub.gateway.removed

Broadcast sent when a gateway got unregistered on the hub (after plugin which provides this gateway has been uninstalled).

```
  {
      "id": "ui_broadcast",
      "msg_subclass": "hub.gateway.removed",
      "result": {
        "_id": "588b7eb528b12d03be86f36f"
      }
    }
```

Field	Type	Required	Description
_id	string	+	an id of the gateway

hub.device.added

Broadcast sent when a device got registered for some gateway.

```
  {
      "id": "ui_broadcast",
      "msg_subclass": "hub.device.added",
      "result": {
        "_id": "588b7eb528b12d03be86f36f",
        "parentDeviceId": "588t7eb528b12d03be86f36f",
        "deviceTypeId": "16_4_1_351_8706_256",
        "gatewayId" : "588b76a44e8c6e50a2826d9f",
        "serviceNotification":false,
        "category": "dimmable_light",
        "subcategory": "dimmable_bulb",
        "name" : "_DEVICE_NAME_",
        "type": "switch",
        "batteryPowered": false,
        "reachable": true,
        "armed": false,
        "roomId" : "_ROOM_ID_",
        "security" : "low",
        "info": "" { "key": "value", "keyN": "valueN" },
        "ready": true,
        "status": "idle"
      }
    }
```

Field	Type	Required	Description
_id	string	+	an id of the device
parentDeviceId	string	+	an id of the parent device. Empty in case of main device
category	string	+	a device category

subcategory	string	+	a device subcategory
deviceTypeId	string	+	a device type id, generated from manufacturer info
gatewayId	string	+	an id of a gateway this device belongs to
name	string	+	a device name
type	string	+	a device type
batteryPowered	bool	+	is device battery powered
reachable	bool	+	is device reachable
armed	bool	+	is device armed by house mode
roomId	string	+	an id of a room this device is assigned to
persistent	bool	-	is device persistent. Persistent device can't be removed by force removing. False by default.
info	object	-	some additional information for this device

serviceNotification	bool	+	Special mark for forwarding all changes with this device and items to different Cloud services
security	string	+	Security level how the device is connected. Possible options: `no`, `low`, `middle`, `high`
ready	bool	+	Ready status of device. `true` value means device is ready to any changes. `false` value means device is busy.
status	string	+	Possible options: `idle`, `broken`. `idle` - device is in normal mode, `broken` - device has invalid data.

---

hub.device.updated

Broadcast sent when some changes happened to a device (broadcast contains only changeS).

```
  {
     "id": "ui_broadcast",
     "msg_subclass": "hub.device.updated",
     "result": {
       "_id": "588b7eb528b12d03be86f36f",
       "deviceArmed": true,
       "serviceNotification":false,
       "__DEVICE_CHANGEABLE_FIELD": value
     }
   }
```

Field	Type	Required	Description

_id	string	+	An id of the device
deviceArmed	bool	+	See device.armed
serviceNotification	bool	+	Special mark for forwarding all changes with this device and items to different Cloud services.
__DEVICE_CHANGEABLE_FIELD	any	+	

Device fields which may be updated (__DEVICE_CHANGEABLE_FIELD):

Field	Type	Description
name	string	a device name
reachable	bool	whether device is reachable
roomId	string	an id of a room this device is assigned to
ready	bool	Ready status of device. `true` value means device is ready to any changes. `false` value means device is busy.

hub.device.removed

Broadcast sent when a device got unregistered from some gateway.

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.device.removed",
    "result": {
      "_id": "588b7eb528b12d03be86f36f",
      "serviceNotification":true
    }
}
```

Field	Type	Required	Description
_id	string	+	an id of the device
serviceNotification	bool	+	Special mark for forwarding all changes with this device and items to different Cloud services

---

hub.item.added

Broadcast sent when an item got registered for some device (can be sent only as a part of a device.added sequence).

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.item.added",
    "result": {
      "_id": "<item_id>",
      "deviceId": "<device_id>",
      "enum": [],
      "deviceName": "_DEVICE_NAME_",
      "deviceArmed": false,
      "hasGetter": true,
      "hasSetter": false,
```

```
        "name": "alarm_water",
        "show": true,
        "scale": "",
        "valueType": "int",
        "valueFormatted": ""
        "value": 0,
        "valueMin": 0,
        "valueMax": 0,
        "elementsMaxNumber": 2,
        "userCodeRestriction": "\d{4}"
        }
    }
```

Field	Type	Required	Description
_id	string	+	an id of the item
deviceId	string	+	an id of a device this item belongs to
enum	array	-	finite array of possible token values
deviceName	string	-	add for house mode events
deviceArmed	bool	-	add for house mode events
hasGetter	bool	+	whether the item provides an ability to get a value
hasSetter	bool	+	whether the item provides an ability to set a value

name	string	+	a name(type) of the item
show	bool	+	whether to show the item (on the UI) or not
scale	string	-	a name of measurement units
valueType	string	+	a type of an item's value
valueFormatted	string	+	an item formatted value
value	object	+	an item value
valueMin	object	-	lower limit of item's value field
valueMax	object	-	upper limit of item's value field
elementsMaxNumber	int	-	max allowed elements of a dictionary value
userCodeRestriction	string	-	restriction for a userCode code field http://www.lua.org/manual/5.3/manual.html#6.4.1

---

hub.item.updated

Broadcast sent when some changes happened to an item (broadcast contains only changes).

```
{
```

```
    "id": "ui_broadcast",
    "msg_subclass": "hub.item.updated",
    "result": {
      "_id": "588b7eb528b12d03be86f36f",
      "deviceId": "<DEVICE_ID>",
      "deviceName": "<DEVICE_NAME>",
      "deviceCategory" : "<DEVICE_CATEGORY>",
      "deviceSubcategory" : "<DEVICE_SUBCATEGORY>",
      "roomName" : "<DEVICE_ROOM_NAME>",
      "userNotification": false,
      "serviceNotification":false,
      "notifications": [ "12314324", "978343" ],
      "deviceArmed": false,
      "name": "alarm_water",
      "elementsMaxNumber": 2,
      "userCodeRestriction": "\d{4}",
      "<ITEM_CHANGEABLE_FIELD>": "<FIELD_VALUE>"
    }
  }
```

Field	Type	Required	Description
_id	string	+	an id of the item
deviceId	string	+	related device._id
deviceName	string	+	related device.name
deviceCategory	string	+	a device category
deviceSubcategory	string	+	a device subcategory
roomName	string	+	a room name

userNotification	bool	+	Special flag for Cloud for converting this broadcast to User Notification
notifications	JsonArray	+	List of user ids for sending broadcast notification to users(null - all users / - no one)
name	string	+	see item.name(type)
deviceArmed	bool	-	related device.armed state
serviceNotification	bool	+	Special mark from related device
elementsMaxNumber	int	-	max allowed elements of a dictionary value
userCodeRestriction	string	-	restriction for a userCode code field http://www.lua.org/manual/5.3/manual.html#6.4.1
__ITEM_CHANGEABLE_FIELD	any	+	

Item fields which may be updated (__ITEM_CHANGEABLE_FIELD):

Field	Type	Description
show	bool	whether to show the item (on the UI) or not

valueFormatted	string	an item value formatted
value	object	an item value
valueMin	object	lower limit of item's value field
valueMax	object	upper limit of item's value field

---

hub.item.removed

Broadcast sent when an item got unregistered from some device (can be sent only as a part of a device.removed sequence).

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.item.removed",
    "result": {
      "_id": "588b7eb528b12d03be86f36f"
    }
}
```

Field	Type	Required	Description
_id	string	+	an id of the item

---

hub.favorite.added

Broadcast with info about added devices, items, rules to favorite. It's sent after hub.favorite.set has been triggered.

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.favorite.added",
```

```
    "initiator": {
        "api_name": "some api name here",
        "connection_type": "UI",
        "peer_unique_id": "23/3"
    },
    "result": {
        "devices": [
            "_DEVICE_ID_"
        ],
        "items": [
            "_ITEM_ID_"
        ],
        "rules": [
            "_RULE_ID_"
        ]
    }
}
```

Field	Description
result.devices *(optional)*	Added devices array
result.items *(optional)*	Added items array
result.rules *(optional)*	Added rules array

---

hub.favorite.removed

Broadcast with info about removed devices, items, rules from favorite. It's sent after hub.favorite.set has been triggered.

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.favorite.removed",
    "initiator": {
        "api_name": "some api name here",
        "connection_type": "UI",
        "peer_unique_id": "23/3"
```

```
    },
    "result": {
        "devices": [
            "_DEVICE_ID_"
        ],
        "items": [
            "_ITEM_ID_"
        ],
        "rules": [
            "_RULE_ID_"
        ]
    }
}
```

Field	Description
result.devices *(optional)*	Removed devices array
result.items *(optional)*	Removed items array
result.rules *(optional)*	Removed rules array

---

hub.modes.switched

Sends information about house mode switch process

result fields:

Field	Type	Required	Description
form	string	+	Id of the from mode

to	string	+	Id of the to mode
status	string	+	"done", "begin" or "cancel"
switchToDelay	integer	+	Delay (sec) before switch to the mod

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.modes.switched",
    "result": {
        "from": "<modeId>",
        "to": "<modeId>",
        "status": "done",
        "switchToDelay": 0
    }
}
```

---

hub.modes.notifications.notify_all

Sends information about house mode notification "send to all"

result fields:

Field	Type	Required	Description
modeId	string	+	Id of the mode
all	bool	+	enable/diable send notifications to all

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.modes.notifications.notify_all",
    "result": {
        "modeId": "<modeId>",
        "all": true
```

```
    }
}
```

---

hub.modes.notifications.added

Sends information about house mode notification list changes (after add)

result fields:

Field	Type	Required	Description
modeId	string	+	Id of the mode
notification	string	+	Id new user ID

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.modes.notifications.added",
    "result": {
        "modeId": "<modeId>",
        "notification": "<userId>"
    }
}
```

---

hub.modes.notifications.removed

Sends information about house mode notification list changes (after remove)

result fields:

Field	Type	Required	Description
modeId	string	+	Id of the mode

notification	string	+	Id removed user ID

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.modes.notifications.removed",
    "result": {
        "modeId": "<modeId>",
        "notification": "<userId>"
    }
}
```

---

hub.modes.disarmed_devices.added

Sends information about house mode disarmed devices list changes (after add)

result fields:

Field	Type	Required	Description
modeId	string	+	Id of the mode
disarmedDevice	string	+	Id new device ID

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.modes.disarmed_devices.added",
    "result": {
        "modeId": "<modeId>",
        "disarmedDevice": "<deviceId>"
    }
}
```

---

hub.modes.disarmed_devices.removed

Sends information about house mode disarmed devices list changes (after remove)

result fields:

Field	Type	Required	Description
modeId	string	+	Id of the mode
disarmedDevice	string	+	Id removed device ID

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.modes.disarmed_devices.removed",
    "result": {
        "modeId": "<modeId>",
        "disarmedDevice": "<deviceId>"
    }
}
```

---

hub.modes.alarms_off.added

Sends information about house mode alarms_off list changes (after add)

result fields:

Field	Type	Required	Description
modeId	string	+	Id of the mode
alarmsOffDevice	string	+	Id new device ID

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.modes.alarms_off.added",
    "result": {
        "modeId": "<modeId>",
        "alarmsOffDevice": "<deviceId>"
```

```
    }
}
```

---

hub.modes.alarms_off.removed

Sends information about house mode alarms_off list changes (after remove)

result fields:

Field	Type	Required	Description
modeId	string	+	Id of the mode
alarmsOffDevice	string	+	Id removed device ID

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.modes.alarms_off.removed",
    "result": {
        "modeId": "<modeId>",
        "alarmsOffDevice": "<deviceId>"
    }
}
```

---

hub.modes.changed

Sends information about changed properties of particular house mode

result fields:

Field	Type	Required	Description
modeId	string	+	Mode ID

disarmedDefault	bool	-	Disarmed default state

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.modes.changed",
    "result": {
        "modeId": "<modeId>",
        "disarmedDefault": true
    }
}
```

---

hub.network.wifi.scan.progress

Sends information about wifi scan process

initiators:

- hub.network.wifi.scan.start
- hub.network.wifi.scan.stop

result fields:

Field	Type	Required	Description
interfaceId	string	+	Id of the network interface (type: wifi)
status	enum	+	"started", "process", "finished" or "failed"
error	object	-	For status "failed" contains error description
networks	JsonArray	-	For status "process" contains Wifi networks list
networks.ssid	string	+	SSID of AP

networks.bssid	string	-	MAC address of AP
networks.security	string	+	Security (open, wep, wpa-psk, wpa2-psk, ...)
networks.rssi	integer	-	Signal strength of AP

errors:

Code	Message	Data	Reason (optional)
-32500	Wifi scan command failed	network.wifi.scan.failed	Scan error message

example:

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.network.wifi.scan.progress",
    "result": {
        "interfaceId": "wlan0",
        "status": "process",
        "networks": [
            {
                "ssid": "TestWIFI",
                "bssid" : "",
                "security": "open",
                "rssi": 0
            },
            {
                "ssid": "eZLO_House",
                "bssid" : "",
                "security": "wpa2-psk",
                "rssi": 0
            }
        ]
    }
}
```

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.network.wifi.scan.progress",
    "result": {
        "error": {
            "code": -32500,
            "data": "network.wifi.scan.failed",
            "message": "Wifi scan failed",
            "reason": "command failed: No such device (-19)"
        },
        "interfaceId": "ra0000000",
        "status": "failed"
    }
}
```


---

hub.network.changed

Sends changes to network interfaces.

result:

Fields and their meaning are same as in hub.network.get result. If some information disappeared corresponding field will have null value

Examples

Ethernet cable unplugged


```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.network.changed",
    "result": {
        "interfaces": [
            {
                "_id": "eth0",
                "internetAvailable": false,
                "ipv4": {
                    "dns": null,
                    "gateway": null,
                    "ip": null,
                    "mask": null
                },
                "status": "down"
            }
```

```
        ]
    }
}
```

Ethernet cable plugged


```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.network.changed",
    "result": {
        "interfaces": [
            {
                "_id": "eth0",
                "internetAvailable": true,
                "ipv4": {
                    "dns": [
                        "192.168.0.1"
                    ],
                    "gateway": "192.168.0.1",
                    "ip": "192.168.0.228",
                    "mask": "255.255.255.0"
                },
                "status": "up"
            }
        ]
    }
}
```


Ethernet connection losses internet


```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.network.changed",
    "result": {
        "interfaces": [
            {
                "_id": "eth0",
                "internetAvailable": false
            }
        ]
    }
}
```

Successful connect to wifi network


```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.network.changed",
    "result": {
        "interfaces": [
            {
                "_id": "wlan0",
                "hwaddr": "a9:b8:c7:d6:e5:f4",
                "internetAvailable": true,
                "ipv4": {
                    "dns": [
                        "192.168.10.10",
                        "1.1.1.1",
                        "8.8.8.8"
                    ],
                    "gateway": "192.168.10.1",
                    "ip": "192.168.11.142",
                    "mask": "255.255.254.0"
                },
                "status": "up",
                "wifi": {
                    "network": {
                        "bssid": "ab:cd:ef:01:23:45",
                        "encryption": "psk2",
                        "key": "super_wifi_network",
                        "mode": "sta",
                        "ssid": "super_wifi_password"
                    },
                    "region": "00"
                }
            }
        ]
    }
}
```


Failed attempt to connect to wifi network


```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.network.changed",
    "result": {
        "interfaces": [
            {
                "_id": "wlan0",
```

```
                "error": {
                    "code": -32500,
                    "data": "network.connection.failed",
                    "message": "Could not connect to the network"
                },
                "hwaddr": "a9:b8:c7:d6:e5:f4",
                "wifi": {
                    "network": {
                        "bssid": "ab:cd:ef:01:23:45",
                        "encryption": "psk2",
                        "key": "super_wifi_network",
                        "mode": "sta",
                        "ssid": "awful_wifi_password"
                    },
                    "region": "00"
                }
            }
        ]
    }
}
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.network.changed",
    "result": {
        "interfaces": [
            {
                "_id": "wlan0",
                "error": null,
                "hwaddr": null,
                "wifi": {
                    "network": null,
                    "region": null
                }
            }
        ]
    }
}
```


---

hub.extensions.plugin.ui_broadcast

Broadcast with custom data from Lua scripts.

initiators:

- core.send_ui_broadcast

result:

custom format

Examples

Zwave gateway inclusion operation started


```
{
    "id": "ui_broadcast",
    "initiator": {
        "api_name": "",
        "connection_type": "HUB",
        "peer_unique_id": "17/103088"
    },
    "msg_subclass": "hub.extensions.plugin.ui_broadcast",
    "result": {
        "event": "include_started",
        "plugin": "zwave"
    }
}
```


---

hub.extensions.plugin.run.progress

Sends information about progress and result of running plugin script

initiators:

- hub.extensions.plugin.run

result:

Field	Type	Required	Description
completed	int	+	Progress of operation
error	object	-	Error information if happened
operationId	string	+	ID of message

status	enum	+	`finished`

errors:

Code	Message	Data	Reason (optional)
`-32603`	`Script error`	`ezlo.lua.script.error`	Status of lua interpretator
`-32603`	`Script open error`	`ezlo.lua.script.open`	-

Examples

Successful execution of plugin script

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.extensions.plugin.run.progress",
    "result": {
        "operationId": "_ID_",
        "completed": 100,
        "error": null,
        "status": "finished"
    }
}
```

Failed execution of plugin script

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.extensions.plugin.run.progress",
    "result": {
        "operationId": "_ID_",
        "completed": 0,
        "error": {
            "code": -32603,
            "data": "ezlo.lua.script.error",
            "message": "Script error",
            "reason": "basic_string::_M_construct null not valid"
        },
```

```
        "status": "failed"
    }
}
```

---

hub.room.created

Broadcast with id of created room on method hub.room.create

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.room.created",
    "initiator": {
        "api_name": "some api name here",
        "connection_type": "UI",
        "peer_unique_id": "23/3"
    },
    "result": {
        "_id": "_OBJECT_ID_",
        "name": "Guest room"
    }
}
```

Field	Description
_id	Room's id
name	Room's name

hub.room.deleted

Broadcast with room id of deleted room on method hub.room.delete

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.room.deleted",
    "initiator": {
        "api_name": "some api name here",
        "connection_type": "UI",
        "peer_unique_id": "23/3"
    },
    "result": {
```

```
        "_id": "_OBJECT_ID_"
    }
}
```

Field	Description
roomsId	Room's ids

---

hub.room.edited

Broadcast with id of edited room on methods hub.room.name.set

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.room.edited",
    "initiator": {
        "api_name": "some api name here",
        "connection_type": "UI",
        "peer_unique_id": "23/3"
    },
    "result": {
        "_id": "_OBJECT_ID_",
        "name" : "Guest room"
    }
}
```

Field	Description
_id	id of created room
name	Room's name

---

hub.scene.added

Broadcast about either the scene is successfully created or creation failed. Related to hub.scenes.create function.

Successful example:


```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.scene.added",
    "result": {
        "_id": "5c7ff48b7f00002a07a408e3",
        "enabled": true,
        "group_id": null,
        "is_group": false,
        "name": "testRule",
        "parent_id": "5c6ec961cc01eb07f86f9dd9",
        "then": [
            {
                "blockOptions": {
                    "method": {
                        "args": {
                            "item": "item",
                            "value": "value"
                        },
                        "name": "setItemValue"
                    }
                },
                "blockType": "then",
                "fields": [
                    {
                        "name": "item",
                        "type": "item",
                        "value": "5c7fea737f00000ab55f2e5d"
                    },
                    {
                        "name": "value",
                        "type": "bool",
                        "value": false
                    }
                ]
            }
        ],
        "when": [
            {
                "blockOptions":{
```

```
                    "method":{
                        "args":{
                            "sunstate":"sunrise",
                            "time":"time"
                        },
                        "name":"isSunState"
                    }
                },
                "blockType":"when",
                "fields":[
                    {
                        "name":"sunrise",
                        "type":"string",
                        "value":"before"
                    },
                    {
                        "name":"time",
                        "type":"hms_interval",
                        "value":"10:30"
                    }
                ]
            }
        ]
    }
}
```

Parameters:

Field	Type	Required	Description
_id	string	+	Scene identifier
enabled	bool	+	Scene is enabled or disable
group_id	string	+	Group identifier
is_group	bool	+	Does this rules related to some group or not

name	string	+	Scene name
parent_id	string	+	Identifier of the room which this scene is linked to
when	Objects Array	+	List of `when` blocks
then	Objects Array	+	List of `then` blocks

hub.scene.deleted

Broadcast about the scene deleting. Related to hub.scenes.delete function.

Example:

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.scene.deleted",
    "result": {
        "_id": "5c7ff48b7f00002a07a408e3"
    }
}
```

Parameters:

Field	Type	Required	Description
_id	string	+	Scene identifier

hub.scene.changed

Broadcast about updating of the scene. Related to hub.scenes.edit, hub.scenes.enabled.set, hub.scenes.notification.add, hub.scenes.notification.remove and hub.scenes.room.set functions.

Example:

```
{
    "id": "ui_broadcast",
```

```
    "msg_subclass": "hub.scene.changed",
    "changed_by": "hub.scenes.edit",
    "result": {
        "_id": "5c7ff48b7f00002a07a408e3",
        "enabled": true,
        "group_id": null,
        "is_group": false,
        "name": "testRule",
        "parent_id": "5c6ec961cc01eb07f86f9dd9",
        "then": [
            {
                "blockOptions": {
                    "method": {
                        "args": {
                            "item": "item",
                            "value": "value"
                        },
                        "name": "setItemValue"
                    }
                },
                "blockType": "then",
                "fields": [
                    {
                        "name": "item",
                        "type": "item",
                        "value": "5c7fea737f00000ab55f2e5d"
                    },
                    {
                        "name": "value",
                        "type": "bool",
                        "value": false
                    }
                ]
            }
        ],
        "when": [
            {
               "blockOptions":{
                  "method":{
                     "args":{
                        "sunstate":"sunrise",
                        "time":"time"
                     },
                     "name":"isSunState"
                  }
               },
               "blockType":"when",
```

```
                "fields":[
                    {
                        "name":"sunrise",
                        "type":"string",
                        "value":"before"
                    },
                    {
                        "name":"time",
                        "type":"hms_interval",
                        "value":"10:30"
                    }
                ]
            }
        ]
    }
}
```

Parameters:

Field	Type	Required	Description
_id	string	+	Scene identifier
enabled	bool	+	Scene is enabled or disable
group_id	string	+	Group identifier
is_group	bool	+	Does this rules related to some group or not
name	string	+	Scene name
parent_id	string	+	Identifier of the room which this scene is linked to

when	Objects Array	+	List of `when` blocks
then	Objects Array	+	List of `then` blocks
changed_by	string	+	Method made changes in scene

---

hub.scene.run.progress

Notification about the scene status. It's fired when scene is started, finished or failed.

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.scene.run.progress",
    "result": {
        "scene_id": "5c7ff48b7f00002a07a408e3",
        "scene_name": "Scene Name",
        "status": "started",
        "userNotification": false,
        "notifications": [
            "23342342342",
            "87976434688"
        ],
        "room_id": "34524dsfsd",
        "room_name": "Living Room"
    }
}
```

Field	Type	Required	Description
_id	string	+	Scene identifier
name	string	+	Scene name

status	string	+	Status of scenes execution progress. Possible statuses are `started`, `finished` and `failed`
userNotification	bool	+	This flag is false if status is `started`. Otherwise it's true. It is needed for cloud
notifications	JsonArray	+	List of user ids for sending broadcast notification to users(null - all users / - no one)
roomId	string	-	Identifier of room to what scene is related
roomName	string	-	Name of room to what scene is related

---

hub.user.notification

The notification of user about specific types of alerts.

Example:

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.user.notification",
    "result": {
        "type": "pin_code_is_used",
        "params": {
              "user": "User Name",
              "deviceName": "Device Name",
              "roomName": "Room Name"
        }
    }
}
```

Parameters:

Field	Type	Required	Description
type	string	+	Broadcast type
params	string	+	Broadcast parameters

Type pin_code_is_used:

Field	Type	Required	Description
params.user	string	+	Related with pin code user name
params.deviceName	string	+	Related device name
params.roomName	string	+	Related room name

---

hub.item.dictionary.updated

Broadcast sent when some changes happened to an dictionary item. This broadcast is sent instead of hub.item.update broadcast.

Initiators:

- hub.item.dictionary.value.add
- hub.item.dictionary.value.set
- hub.item.dictionary.value.remove

Example:

```
{
    "id": "ui_broadcast",
    "msg_subclass": "hub.item.dictionary.updated",
    "result": {
        "_id": "588b7eb528b12d03be86f36f",
        "deviceId": "5df0b9e4dfdabe58a5a60020",
```

```
        "deviceName": "Touchscreen Deadbolt",
        "deviceCategory" : "door_lock",
        "deviceSubcategory" : "",
        "roomName" : "",
        "userNotification": false,
        "serviceNotification":false,
        "notifications": [ "12314324", "978343" ],
        "deviceArmed": false,
        "name": "user_codes",
        "operation":"added",
        "element": {
            "1": {
                "value": {
                    "code": "Alex",
                    "name": "1234"
                }
            }
        }
    }
}
```

Parameters:

Field	Type	Required	Description
_id	string	+	an id of the item
deviceId	string	+	related device._id
deviceName	string	+	related device.name
deviceCategory	string	+	a device category
deviceSubcategory	string	+	a device subcategory

roomName	string	+	a room name
userNotification	bool	+	Special flag for Cloud for converting this broadcast to User Notification
notifications	JsonArray	+	List of user ids for sending broadcast notification to users(null - all users / - no one)
name	string	+	see item.name(type)
deviceArmed	bool	-	related device.armed state
serviceNotification	bool	+	Special mark from related device
elementsMaxNumber	int	+	max allowed elements of a dictionary value
userCodeRestriction	string	-	restriction for a userCode code field Lua Manual
operation	string	+	The operation field defines the initiator of this broadcast: hub.item.dictionary.value.add - `added` hub.item.dictionary.value.set - `updated`

			hub.item.dictionary.value.remove - `removed`
element	JsonObject	+	The element what was updated. It contains element number and element value.
element.N.value	any	+	The element value.

Device categories

Category	Vera ID	Vera Name	Subcategory	Vera Sub ID	Vera Sub Name
`interface`	1	Interface			
`dimmable_light`	2	Dimmable Light	`dimmable_bulb`	1	Bulb
			`dimmable_plugged`	2	Plugged
			`dimmable_in_wall`	3	In Wall

			`dimmable_colored`	4	RGB
`switch`	3	Switch	`interior_plugin`	1	Interior
			`exterior_plugin`	2	Exterior
			`in_wall`	3	In Wall
			`refrigerator`	4	Refrigerator
				5	Garage Door
				6	Doorbell
			`valve`	7	Valve
			`relay`	8	Relay
`security_sensor`	4	Security Sensor	`door`	1	Door Sensor
			`leak`	2	Leak Sensor
			`motion`	3	Motion Sensor

			smoke	4	Smoke Sensor
			co	5	CO Sensor
			glass	6	Glass Break Sensor
			co2	7	CO2 Sensor
			gas	8	Gas Sensor
			heat	9	Heat Sensor
hvac	5	HVAC	hvac	1	HVAC
			heater	2	Heater
			custom_hvac	3	Custom HVAC
camera	6	Camera	indoor_cam	1	Indoor
			outdoor_cam	2	Outdoor
			doorbell_cam	3	Doorbell

door_lock	7	Door Lock			
window_cov	8	Window Covering	window_cov	1	Window Covering
			zrtsi	2	ZRTSI
remote_control	9	Remote Control			
ir_tx	10	IR Transmitter	irt	1	IR Transmitter
			usbuirt	2	USB UIRT
generic_io	11	Generic I/O	generic_io	1	Generic I/O
			repeater	2	Repeater
generic_sensor	12	Generic Sensor			
serial_port	13	Serial Port			

`scene_controller`	14	Scene Controller
`av`	15	A/V
`humidity`	16	Humidity Sensor
`temperature`	17	Temperature Sensor
`light_sensor`	18	Light Sensor
`zwave_int`	19	Z-Wave Interface
`insteon_int`	20	Insteon Interface
`power_meter`	21	Power Meter
`alarm_panel`	22	Alarm Panel
`alarm_partition`	23	Alarm Partition

`siren`	24	Siren
`weather`	25	Weather
`philips_controller`	26	Philips Controller
`appliance`	27	Appliance
`uv_sensor`	28	UV Sensor
`mouse_trap`	29	Mouse Trap
`doorbell`	30	Doorbell
`keypad`	31	Keypad
`garage_door`	32	Garage Door
`flow_meter`	33	Flow meter
`voltage_sensor`	34	Voltage Sensor

state_sensor	35	State Sensor	light	1	Light
			rain	2	Rain
			moisture	3	Moisture
			freeze	4	Freeze
			power	5	Power
level_sensor	36	Level Sensor	co2	1	CO2
			co	2	CO
			current	3	Current
			velocity	4	Velocity
			capacity	5	Capacity
			water	6	Water

	`particulate_matter`	7	Particulate Matter
	`frequency`	8	Frequency
	`health`	9	Health
	`modulation`	10	Modulation
	`smoke`	11	Smoke
	`soil`	12	Soil
	`moisture`	13	Moisture
	`air_pollution`	14	Air Pollution
	`electricity`	15	Electricity
	`sound`	16	Sound
	`navigation`	17	Navigation
	`seismicity`	18	Seismicity

			`time`	19	Time
`clock`	37	Clock			

Items

Name	Value Type	Enum	Methods	Default value	Description
acceleration_x_axis	`acceleration`		Getter	*0*	Device position in space was canged around X axis
acceleration_y_axis	`acceleration`		Getter	*0*	Device position in space was canged around Y axis
acceleration_z_axis	`acceleration`		Getter	*0*	Device position in space was canged around Z axis

appliance_status	`token`	Appliance Status	Getter	*unknown*
lock_operation	`token`	Lock Operation	Getter	*unknown*
user_code_operation	`token`	User Code Operation	Getter	*no_operation*
dw_handle_state	`token`	Door/Window Handle State	Getter	*unknown*
dw_state	`token`	Door/Window State	Getter	*unknown*
keypad_state	`token`	Keypad State	Getter	*unknown*

barrier_initialization	`token`	Barrier initialization states	Getter	*unknown*
barrier_unattended_operation	`token`	Barrier unattended operation enabled/disabled events	Getter	*unknown*
barrier_vacation_mode	`token`	Barrier vacation mode	Getter	*unknown*
barrier_safety_beam_obstacle	`token`	Barrier safety beam obstacle events	Getter	*unknown*

barrier_problem_sensors	`dictionary` of `token`	Barrier Problem Sensors	Getter	Empty dictionary	Map sensorID to status token.
barrier_short_circuit	`token`	Short Circuit States	Getter	*unknown*	
barrier_fail_events	`token`	Barrier fail events	Getter	*no_barrier_fails*	
button_state	`button_state`	Button actions	Getter	*released*	Controller button state.
test_state	`token`	Test State	Getter	*unknown*	Get result of testing device
co_alarm	`token`	Alarm Events(CO)	Getter	*unknown*	
maintenance_state	`token`	Maintenanc	Getter	*unknown*	

		e State			
sounding_mode	`token`	Sounding Mode	Getter(optional) / Setter(optional)	*unknown*	Getter and/or setter are present depend on device functionality
sound_list	`dictionary` of `sound_info`		Getter	Empty dictionary	List of the supported tones
sound_volume	`int`		Getter / Setter	*100*	Tone volume configuration
sound_select	`int`		Getter / Setter	*1*	Selected tone configuration
sound_playback	`token`	Sound Playback	Getter / Setter	*false*	Play or stop playing the selected tone
periodic_inspection_state	`token`	Periodic Inspec	Getter	*unknown*	

		tion State		
co2_alarm	`token`	Alarm Events(CO2)	Getter	*unknown*
gas_alarm	`token`	Alarm Sensor Events(Gas)	Getter	*unknown*
heat_alarm	`token`	Alarm Events(Heat)	Getter	*unknown*
siren_alarm	`token`	Alarm Events(Siren)	Getter	*unknown*
light_alarm	`token`	Alarm Event	Getter	*unknown*

		s(Light)		
light_color_transition	`token`	Light changes	Getter	*unknown*
temperature_changes	`token`	Temperature changes	Getter	*unknown*
intrusion_alarm	`token`	Intrusion State	Getter	*unknown*
tampering_cover_alarm	`token`	Tampering Cover State	Getter	*unknown*
glass_breakage_alarm	`token`	Glass Breakage State	Getter	*unknown*

tampering_move_alarm	`token`	Tampering Move State	Getter	*unknown*
tampering_impact_alarm	`token`	Tampering Impact State	Getter	*unknown*
tampering_invalid_code_alarm	`token`	Tampering Invalid Code State	Getter	*unknown*
smoke_alarm	`token`	Alarm Sensor Events(Smoke)	Getter	*unknown*
dust_in_device	`token`	Dust In Device	Getter	*unknown*

water_leak_alarm	`token`	Water Leak Alarm	Getter	*unknown*
water_filter_replacement_alarm	`token`	Water Filter Replacement Alarm	Getter	*unknown*
water_flow_alarm	`token`	Water Flow Alarm	Getter	*unknown*
water_pressure_alarm	`token`	Water Pressure Alarm	Getter	*unknown*
water_temperature_alarm	`token`	Water Temperature Alarm	Getter	*unknown*

water_level_alarm	`token`	Water Level Alarm	Getter	*unknown*
water_pump_state	`token`	Pump State	Getter	*unknown*
water_valve_state	`token`	Water Walve State	Getter / Setter(optional)	*unknown*
master_water_valve_state	`token`	Water Walve State	Getter / Setter(optional)	*unknown*
water_valve_short_circuit	`token`	Short Circuit State	Getter	*unknown*
master_water_valve_short_circuit	`token`	Short Circuit State	Getter	*unknown*

water_valve_current_alarm	`token`	Valve Current Alarm State	Getter	*unknown*
master_water_valve_current_alarm	`token`	Valve Current Alarm State	Getter	*unknown*
rain_alarm	`token`	Rain State	Getter	*unknown*
moisture_alarm	`token`	Moisture State	Getter	*unknown*
freeze_alarm	`token`	Freeze State	Getter	*unknown*
power_state	`token`	Power State	Getter	*unknown*

ac_state	`token`	AC State	Getter	*unknown*
power_surge_state	`token`	Power Surge State	Getter	*no_surge*
voltage_drop_drift_state	`token`	Voltage Drop/ Drift State	Getter	*unknown*
over_current_state	`token`	Over Current State	Getter	*unknown*
over_voltage_state	`token`	Over Voltage State	Getter	*unknown*
over_load_state	`token`	Over Load State	Getter	*unknown*

load_error_state	`token`	Load Error State	Getter	*unknown*	
battery_maintenance_state	`token`	Battery Maintenance State	Getter	*unknown*	Actual state of device battery. State may describe desired action, e.g. "replace battery now"
battery_charging_state	`token`	Battery Charging State	Getter	*unknown*	
battery_backup	`int`		Getter	*100*	
angle_position	`angle`		Getter	*0*	
atmospheric_pressure	`pressure`		Getter	*0*	
barometric_pressure	`pressure`		Getter	*0*	

barrier	`token`	Barrier Operator Events	Setter	First available token from enum	Request barier state change, only "barrier_opened" and "barrier_closed" events are supported
barrier_state	`token`	Barrier Operator Events	Getter	*operating_unknown*	Get current barrier state
hw_state	`token`	Hardware State	Getter	*unknown*	
sw_state	`token`	Software State	Getter	*unknown*	
emergency_shutoff	`token`	Emergency State	Getter	*unknown*	

digital_input_state	`token`	Digital Input State	Getter	*unknown*
clock_state	`token`	Clock State	Getter	*unknown*
remaining_time	`time`		Getter	*0*
basic	`bool`		Getter / Setter	*false*
battery	`int`		Getter	*100*
co2_level	`substance_amount`		Getter	*0*
co_level	`substance_amount`		Getter	*0*
dew_point	`temperature`		Getter	*0*
dimmer	`int`		Getter / Setter	*0*
dimmer_down	`int`		Setter	*0*

dimmer_stop	`int`		Setter	*0*
dimmer_up	`int`		Setter	*0*
direction	`direction`		Getter	*0*
distance	`length`		Getter	*0*
door_lock	`token`	Door Lock Modes	Getter / Setter	*unknown*
electric_meter_amper	`float`		Getter	*0*
electric_meter_kvah	`float`		Getter	*0*
electric_meter_kvar	`float`		Getter	*0*
electric_meter_kvarh	`float`		Getter	*0*
electric_meter_kwh	`float`		Getter	*0*
electric_meter_power_factor	`float`		Getter	*0*

electric_meter_pulse	`int`	Getter	*0*
electric_meter_volt	`float`	Getter	*0*
electric_meter_watt	`float`	Getter	*0*
electric_resist	`float`	Getter	*0*
humidity	`humidity`	Getter	*0*
loudness	`loudness`	Getter	*0*
lux	`illuminance`	Getter	*0*
meter_reset	`int`	Setter	*0*
moisture	`moisture`	Getter	*0*
Deprecated: **motion**	`bool`	Getter	*false*
power	`power`	Getter	*0*
pressure	`pressure`	Getter	*0*

rgbcolor	`rgb`		Getter / Setter	*0* for all available colors	
rotation	`frequency`		Getter	*0*	
security_threat	`bool`		Getter	*0*	Manual item. Should be added for all 'security' devices. It has `true` value in dangerous states of security device.
seismic_intensity	`seismic_intensity`		Getter	*0*	
seismic_magnitude	`seismic_magnitude`		Getter / Setter	*0*	
rain_rate	`precipitation`		Getter	*0*	
shutter_command	`token`	Shutter	Setter		Item to control shutter's border and

		Commands			favorite position learning.
shutter_state	`token`	Shutter States	Getter	*unknown*	Item to control shutter's learning states.
soil_temperature	`temperature`		Getter	*0*	
solar_radiation	`irradiance`		Getter	*0*	
switch	`bool`		Getter / Setter	*false*	
temp	`temperature`		Getter	*0*	Temperature
thermostat_fan_mode	`token`	Thermostat Fan Modes	Getter / Setter	First available token from enum	
thermostat_fan_state	`token`	Thermostat	Getter / Setter	First available token	

		Fan States		from enum	
thermostat_mode	`token`	Thermostat Modes	Getter / Setter	First available token from enum	
thermostat_operating_state	`token`	Thermostat Operating States	Getter	*idle*	
thermostat_setpoint	`temperature`		Getter / Setter	*0* or closest to *0*	If thermostat mode is "auto", "aux" or "away", value of this item is an average of "thermostat_setpoint_heating" and "thermostat_setpoint_cooling" item values

thermostat_setpoint_heating	`temperature`	Getter / Setter	*0* or closest to *0*	Valid only for "auto", "aux" and "away" thermostat modes.
thermostat_setpoint_cooling	`temperature`	Getter / Setter	*0* or closest to *0*	Valid only for "auto", "aux" and "away" thermostat modes.
tide_level	`length`	Getter	*0*	
ultraviolet	`ultraviolet`	Getter	*0*	
user_codes	`dictionary` of `userCode`	Getter / Add dictionary value / Set dictionary value / Remove diction	Empty dictionary	

		ary value		
user_lock_operation	`user_lock_operation`	Getter	*userId=-1, action="unknown"*	Display which user code was used to lock or unlock device
user_codes_scan_progress	`int`	Getter	*100*	Shows progress of user codes requesting (in percents), when it is required to get all user codes values.
velocity	`velocity`	Getter	*0*	
voltage	`electric_potential`	Getter	*0*	
current	`electric_current`	Getter	*0*	
weight	`mass`	Getter	*0*	

air_flow	`volume_flow`	Getter	*0*
tank_capacity	`volume`	Getter	*0*
water_temperature	`temperature`	Getter	*0*
electrical_resistivity	`electrical_resistance`	Getter	*0*
electrical_conductivity	`electrical_conductivity`	Getter	*0*
frequency	`frequency`	Getter	*0*
time_period	`time`	Getter	*0*
target_temperature	`temperature`	Getter	*0* or closest to *0*
blood_pressure	`blood_pressure`	Getter	*0*
water_flow	`volume_flow`	Getter	*0*
water_pressure	`pressure`	Getter	*0*

boiler_water_temperature	`temperature`	Getter	*0* or closest to *0*
domestic_hot_water_temperature	`temperature`	Getter	*0* or closest to *0*
outside_temperature	`temperature`	Getter	*0* or closest to *0*
exhaust_temperature	`temperature`	Getter	*0* or closest to *0*
heat_rate_lf_hf_ratio	`float`	Getter	*0*
water_oxidation_reduction_potential	`electric_potential`	Getter	*0*
water_chlorine_level	`substance_amount`	Getter	*0*
exhaust_temperature	`temperature`	Getter	*0* or closest to *0*

outside_temperature	`temperature`		Getter	*0* or closest to *0*	
relative_modulation_level	`temperature`		Getter	*0* or closest to *0*	
water_chlorine_level	`substance_amount`		Getter	*0*	
water_acidity	`acidity`		Getter	*0*	
particulate_matter_10	`substance_amount`		Getter	*0*	
particulate_matter_2_dot_5	`substance_amount`		Getter	*0*	
program_status	`token`	Program Status	Getter	*unknown*	
program_failures	`dictionary` of `token`	Program Failed Status	Getter	*all supported programs with 'ok' value*	Program to status (failed or not)

position	`token`	Position	Getter	*unknown*	
sleep_apnea	`token`	Sleep Apnea Status	Getter	*unknown*	
sleep_stage	`token`	Sleep Stage	Getter	*unknown*	
voc_level_status	`token`	VOC Level Status	Getter	*unknown*	Volatile Organic Compound level status
rf_signal_strength	`rf_signal_strength`		Getter	*0*	
basal_metabolic_rate	`energy`		Getter	*0*	
body_mass_index	`float`		Getter	*0*	
body_mass	`mass`		Getter	*0*	
total_body_water	`mass`		Getter	*0*	

fat_mass	`mass`	Getter	*0*
muscle_mass	`mass`	Getter	*0*
relative_modulation_level	`general_purpose`	Getter	*0*
respiratory_rate	`frequency`	Getter	*0*
smoke_density	`density`	Getter	*0*
heart_rate	`frequency`	Getter	*0*
soil_salinity	`substance_amount`	Getter	*0*
soil_reactivity	`substance_amount`	Getter	*0*
soil_humidity	`humidity`	Getter	*0*
volatile_organic_compound_level	`substance_amount`	Getter	*0*
methane_density	`substance_amount`	Getter	*0*

radon_concentration	`radon_concentration`	Getter	*0*
formaldehyde_level	`substance_amount`	Getter	*0*
weekly_user_code_intervals	`dictionary` of `weekly_interval_array`	Getter / Setter	Empty dictionary
daily_user_code_intervals	`dictionary` of `daily_interval_array`	Getter / Setter	Empty dictionary

Item value types

Basic types

---

int

Integer

example:

```
    {
        ...
        "valueType": "int",
        "value": 526
        ...
    }
```

---

bool

Boolean

example:

```
    {
          ...
          "valueType": "bool",
          "value": false
          ...
     }
```

---

float

Floating point number

example:

```
    {
          ...
          "valueType": "float",
          "value": 5.0
          ...
     }
```

---

string

String

example:

```
    {
          ...
          "valueType": "string",
          "value": "example",
          ...
     }
```

Custom types

---

dictionary

Specific type for mapping multiple values with same type within item
'value' - map of unique string keys to elements of the supported types
'enum' - array of supported token values
'valueType' - "dictionary"
'elementType' - valid value type of elements in the dictionary. If element type is container (dictionary or array), split subtypes by '.'

example:

```
    {
        ...
        "valueType": "dictionary",
        "elementType": "array.daily_interval",
        "value": {
            "1": [
                {
                    "startDateTime": "2019-11-30T21:00:00",
                    "stopDateTime": "2019-11-30T22:00:00",
                },
                {
                    "startDateTime": "2020-11-30T21:00:00",
                    "stopDateTime": "2020-11-30T22:00:00",
                }
            ],
            "3": [
                {
                    "startDateTime": "2019-10-30T21:00:00",
                    "stopDateTime": "2019-10-30T22:00:00",
                }
            ],
        },
        ...
    }
```

```
    {
        ...
        "valueType": "dictionary",
        "elementType": "temperature"
        "value": {
            "key_1": {
                "value": 36,
                "scale": "celsius"
            },
```

```
            "key_2": {
                "value": 6,
                "scale": "fahrenheit"
            },
        },
        ...
    }
```

array

The array gives possibility to create array of types described in this page.
'value' - array of one of supported types.
'valueType' - "array"
'elementType' - valid value type of elements in the array. If element type is container (dictionary or array), split subtypes by '.'
[deprecated] 'valueType' - #subtype_array. Add suffix _array to name of existing type.

examples:

```
{
    "valueType": "array",
    "elementType": "int",
    "value": [ 11, 232, 5 ]
}
```

```
{
    "valueType": "array",
    "elementType": "daily_interval",
    "value": [
        {
            "startDateTime": "2019-10-30T21:30:00",
            "stopDateTime": "2019-11-30T22:30:00"
        },
        {
            "startDateTime": "2020-10-30T19:30:00",
            "stopDateTime": "2020-11-30T20:30:00"
        }
    ]
}
```

```
{
    "valueType": "array",
    "elementType": "array.string",
    "value": [
```

```
        [ "a", "b" ]
    ]
}
```

[deprecated] examples:

```
{
    "valueType": "int_array",
    "value": [ 11, 232, 5 ]
}
```

```
{
    "valueType": "daily_interval_array",
    "value": [
        {
            "startDateTime": "2019-10-30T21:30:00",
            "stopDateTime": "2019-11-30T22:30:00"
        },
        {
            "startDateTime": "2020-10-30T19:30:00",
            "stopDateTime": "2020-11-30T20:30:00"
        }
    ]
}
```

---

rgb

RGB color value

example:

```
    {
        ...
        "valueType": "rgb",
        "value": {
            "wwhite":10,
            "cwhite":10,
            "red":10.
            "green":10,
            "blue":10,
            "amber":10,
            "cyan":10,
            "purple":10,
```

```
            "indexed":10
        }
        ...
    }
```

---

userCode

Special User Code format

example:

```
    {
        ...
        "valueType": "userCode",
        "value": {
            "code": "some code",
            "name": "code name"
        }
        ...
    }
```

---

weekly_interval

Special weekly interval format Value contains time interval and set of week days. If startTime > stopTime, stopTime means time in next day.

example:

```
    {
        ...
        "valueType": "weekly_interval",
        "value": {
            "weekDays": [ "monday", "thursday" ],
            "startTime": "11:50:54",
            "stopTime": "17:59:59",
        }
        ...
    }
```

---

daily_interval

Special daily interval format

example:

```
    {
        ...
        "valueType": "daily_interval",
        "value": {
            "startDateTime": "2020-01-03T11:50:54",
            "stopDateTime": "2020-04-30T17:59:59",
        }
        ...
    }
```

---

token

Value of enumeration from 'enum' field

example:

```
    {
        ...
        "valueType": "token",
        "value": "ezlo.device.value.dry_ok",
        "enum": [
            "ezlo.device.value.dry_ok",
            "ezlo.device.value.leak",
            "ezlo.device.value.unknown_event"
        ],
        ...
    }
```

---

button_state

Value for scenes controller buttons

example:

```
    {
        ...
```

```
        "valueType": "button_state",
        "value": {button_number = 1, button_state = "press_1_time"},
        "enum": [
            "press_1_time",
            "held_down",
            "released"
        ],
        ...
    }
```

---

user_lock_operation

User lock operation value
'action' - display what user has done
'userId' - user id of user code (PIN code). Value '-1' is invalid user id

Actions
`unknown`
`lock`
`unlock`

example:

```
    {
        ...
        "valueType": "user_lock_operation",
        "value": {
            "action": "unknown",
            "userId": -1
        },
        ...
    }
```

---

sound_info

Tone info
'name' - name of the tone
'duration' - duration of the tone in seconds

example:

```
    {
            ...
            "valueType": "sound_info",
            "value": {
                "name": "1AMBUL~1",
                "duration": 20
            },
            ...
        }
```

Scalable types

Illuminance

Illuminance

Scales

`percent`

`lux`

example:

```
    {
          ...
          "valueType": "illuminance",
          "value": 6.0,
          "scale": "lux",
          ...
     }
```

Pressure

Pressure

Scales

`kilo_pascal`

`pound_per_square_inch`

`inches_of_mercury`

example:

```
    {
          ...
          "valueType": "pressure",
          "value": 4623.0,
          "scale": "kilo_pascal",
          ...
     }
```

Substance_amount

Substance amount

Scales
`percent`
`gram_per_cubic_meter`
`micro_gram_per_cubic_meter`
`mole_per_cubic_meter`
`parts_per_million`
`milli_gram_per_liter`

example:

```
    {
          ...
          "valueType": "substance_amount",
          "value": 5.0,
          "scale": "percent",
          ...
     }
```

Power

Power

Scales

watt

btu_per_hour

example:

```
   {
        ...
        "valueType": "power",
        "value": 24.0,
        "scale": "watt",
        ...
    }
```

Velocity

Velocity

Scales

meter_per_second

mile_per_hour

example:

```
   {
        ...
        "valueType": "velocity",
        "value": 3,
        "scale": "meter_per_second",
        ...
    }
```

Acceleration

Acceleration

Scales

`meter_per_square_second`

example:

```
    {
        ...
        "valueType": "acceleration",
        "value": 14.6,
        "scale": "meter_per_square_second",
        ...
    }
```

Direction

Direction

Scales

`no_direction`

`east`

`south`

`west`

`north`

example:

```
   {
        ...
        "valueType": "direction",
        "value": 1,
        "scale": "south",
        ...
    }
```

General_purpose

General purpose value for absolute and relative magnitudes

Scales

`percent`

`float`

example:

```
   {
        ...
        "valueType": "general_purpose",
        "value": 56,
        "scale": "percent",
        ...
    }
```

Acidity

Acidity

Scales

`potential_of_hydrogen`

example:

```
    {
        ...
        "valueType": "acidity",
        "value": 36,
        "scale": "potential_of_hydrogen",
        ...
    }
```

Electric_potential

Electric potential

Scales

`milli_volt`

`volt`

example:

```
    {
        ...
        "type": "electric_potential",
        "value": 36,
        "scale": "milli_volt",
        ...
    }
```

Electric_current

Electric current

Scales

`ampere`

`milli_ampere`

example:

```
   {
         ...
         "valueType": "electric_current",
         "value": 36,
         "scale": "milli_ampere",
         ...
    }
```

Force

Force

Scales

`newton`

example:

```
   {
         ...
         "valueType": "force",
         "value": 36,
         "scale": "newton",
         ...
    }
```

Irradiance

Irradiance

Scales

`watt_per_square_meter`

example:

```
    {
        ...
        "valueType": "irradiance",
        "value": 36,
        "scale": "watt_per_square_meter",
        ...
    }
```

Precipitation

Precipitation

Scales

`milli_meter_per_hour`

`inches_per_hour`

example:

```
    {
        ...
        "valueType": "precipitation",
        "value": 36,
```

```
        "scale": "inches_per_hour",
        ...
    }
```

Length

Length

Scales

`meter`

`feet`

`centi_meter`

example:

```
    {
        ...
        "valueType": "length",
        "value": 36,
        "scale": "meter",
        ...
    }
```

Mass

Mass

Scales

kilo_gram

pounds

example:

```
    {
        ...
        "type": "mass",
        "value": 36,
        "scale": "kilo_gram",
        ...
    }
```

Volume_flow

Volume flow

Scales

cubic_meter_per_hour

cubic_feet_per_minute

liter_per_hour

example:

```
    {
        ...
        "valueType": "volume_flow",
        "value": 36,
        "scale": "cubic_feet_per_minute",
        ...
    }
```

Volume

Volume

Scales

`liter`

`cubic_meter`

`gallons`

example:

```
    {
          ...
          "valueType": "volume",
          "value": 36,
          "scale": "cubic_meter",
          ...
     }
```

Angle

Angle

Scales

`percent`

`north_pole_degress`

example:

```
   {
        ...
        "type": "angle",
        "value": 36,
        "scale": "north_pole_degress",
        ...
    }
```

Frequency

Frequency

Scales

`revolutions_per_minute`

`hertz`

`kilo_hertz`

`breaths_per_minute`

`beats_per_minute`

example:

```
   {
        ...
        "valueType": "frequency",
        "value": 36,
```

```
        "scale": "revolutions_per_minute",
        ...
    }
```

Seismic_intensity

Seismic intensity

Scales

`mercalli`

`european_macroseismic`

`liedu`

`shindo`

example:

```
    {
        ...
        "valueType": "seismic_intensity",
        "value": 36,
        "scale": "mercalli",
        ...
    }
```

Seismic_magnitude

Seismic magnitude

Scales

`local`

`moment`

`surface_wave`

`body_wave`

example:

```
   {
        ...
        "valueType": "seismic_magnitude",
        "value": 36,
        "scale": "surface_wave",
        ...
    }
```

Ultraviolet

Ultraviolet

Scales

`uv_index`

example:

```
   {
        ...
        "valueType": "ultraviolet",
        "value": 36,
        "scale": "uv_index",
        ...
```

```
    }
```

Electrical_resistance

Electrical resistance

Scales

`ohm_meter`

example:

```
    {
        ...
        "valueType": "electrical_resistance",
        "value": 36,
        "scale": "ohm_meter",
        ...
    }
```

Electrical_conductivity

Electrical conductivity

Scales

`siemens_per_meter`

example:

```
    {
        ...
        "valueType": "electrical_conductivity",
        "value": 36,
        "scale": "siemens_per_meter",
        ...
    }
```

Loudness

Loudness

Scales

`decibel`

`a_weighted_decibels`

example:

```
   {
         ...
         "valueType": "loudness",
         "value": 36,
         "scale": "a_weighted_decibels",
         ...
    }
```

Moisture

Moisture

Scales

`percent`

`volume_water_content`

`impedance`

`water_activity`

example:

```
    {
        ...
        "valueType": "moisture",
        "value": 36,
        "scale": "water_activity",
        ...
    }
```

Time

Time

Scales

`seconds`

example:

```
    {
        ...
        "type": "time",
        "value": 36,
        "scale": "seconds",
        ...
    }
```

Radon_concentration

Radon concentration

Scales

`becquerel_per_cubic_meter`

`picocuries_per_liter`

example:

```
    {
        ...
        "valueType": "radon_concentration",
        "value": 36,
        "scale": "becquerel_per_cubic_meter",
        ...
    }
```

Blood_pressure

Blood pressure

Scales

`systolic`

`diastolic`

example:

```
    {
        ...
        "valueType": "blood_pressure",
        "value": 36,
        "scale": "systolic",
        ...
    }
```

Energy

Energy

Scales

`joule`

example:

```
    {
          ...
          "valueType": "energy",
          "value": 36,
          "scale": "joule",
          ...
     }
```

RF_signal_strength

RF signal strength

Scales

`percent`

`decibel_milli_watt`

example:

```
    {
          ...
          "valueType": "rf_signal_strength",
          "value": 36,
          "scale": "decibel_milli_watt",
          ...
     }
```

Temperature

Temperature

Scales

`celsius`

`fahrenheit`

example:

```
    {
        ...
        "valueType": "temperature",
        "value": 36,
        "scale": "celsius",
        ...
    }
```

Humidity

Humidity

Scales

`percent`

`gram_per_cubic_meter`

example:

```
    {
        ...
        "valueType": "humidity",
        "value": 36,
        "scale": "percent",
        ...
    }
```

Scene Value Types

The description of types are used in scene fields. Basically scene uses item types but there are specific types created for scene only. These specific types described here to prevent any intersections with Item Value Types. In future Scene Value Types could be used for Items also.

item

Id of device.

example:

```
{
    "name": "name",
    "type": "device",
    "value": "sdafd8f7sdf756t76d"
}
```

item

Id of device item.

example:

```
{
    "name": "name",
    "type": "item",
    "value": "34234_234_23"
}
```

24_hours_time

Time in format "hh:mm".

possible values:

Time Unit	Possible Values
hours	0...23
minutes	0...60

example:

```
{
    "name": "name",
    "type": "24_hours_time",
```

```
    "value": "13:22"
}
```

Interval

The time interval in format number plus suffix. The suffix defines time units. The possible time units are days, hours, minutes and seconds.

suffix description:

Suffix	Time units
d	days
h	hours
m	minutes
s	seconds

example:

```
{
    "name": "name",
    "type": "interval",
    "value": "10s"
}
```

hms_interval

Time interval in one of following formats: "hh:mm:ss", "mm:ss", "ss".

examples:

```
{
    "name": "name",
    "type": "hms_interval",
    "value": "24:10:30"
```

```
}
```

```
{
    "name": "name",
    "type": "hms_interval",
    "value": "10:30"
}
```

```
{
    "name": "name",
    "type": "hms_interval",
    "value": "30"
}
```


Item Enumerations

Lock Operation

№	Token	Description
1	`manual_lock_operation`	
2	`manual_unlock_operation`	
3	`rf_lock_operation`	
4	`rf_unlock_operation`	

5	`keypad_lock_operation`
6	`keypad_unlock_operation`
7	`manual_not_fully_locked_operation`
8	`rf_not_fully_locked_operation`
9	`auto_lock_not_fully_locked_operation`
10	`unlock_by_rf_with_invalid_user_code`
11	`lock_by_rf_with_invalid_user_code`
12	`lock_jammed`
13	`unknown`

User Code Operation

№	Token	Description
1	`no_operation`	

2	`all_user_codes_deleted`
3	`single_user_code_deleted`
4	`new_user_code_added`
5	`new_user_code_not_added_due_to_duplicated_code`
6	`new_program_code_entered`
7	`manually_enter_user_code_exceeds_code_limit`

Door/Window Handle State

№	Token	Description
1	`dw_handle_is_opened`	
2	`dw_handle_is_closed`	
3	`unknown`	

Door/Window State

№	Token	Description
1	`dw_is_opened`	
2	`dw_is_closed`	
3	`unknown`	

Keypad State

№	Token	Description
1	`keypad_idle`	
2	`keypad_busy`	
3	`keypad_temporary_disabled`	
3	`unknown`	

Barrier Vacation Mode

№	Token	Description

1	`mode_enabled`
2	`mode_disabled`
3	`unknown`

Barrier Unattended Operation

№	Token	Description
1	`unattended_operation_enabled`	
2	`unattended_operation_disabled_per_ul_requirements`	
3	`unknown`	

Barrier Initialization

№	Token	Description
1	`not_started`	
2	`performing_process`	

3	`process_completed`
4	`unknown`

Barrier Safety Beam Obstacle

№	Token	Description
1	`obstruction`	
2	`no_obstruction`	
3	`unknown`	

Short Circuit

№	Token	Description
1	`no_short_circuit`	
2	`short_circuit_detected`	
3	`unknown`	

Barrier Fail Events

№	Token	Description
1	`no_barrier_fails`	
2	`barrier_operation_force_has_been_exceeded`	
3	`barrier_motor_exceeded_operational_time_limit`	
4	`barrier_operation_exceeded_physical_mechanical_limits`	
5	`barrier_failed_to_perform_requested_operation_device_malfunction`	
6	`barrier_unable_to_perform_requested_operation_due_to_ul_requirements`	

Barrier Problem Sensors

№	Token	Description
1	`low_battery`	Sensor's battery has low charge
2	`not_detected`	Sensor is unaccessible

Alarm Events(CO)

№	Token	Description
1	`no_co`	
2	`co_detected`	
3	`unknown`	

Maintenance State

№	Token	Description
1	`ok`	
2	`replacement_required`	
3	`device_end_of_life`	
4	`replace_main_filter`	
5	`unknown`	

Test State

№	Token	Description
1	`no_test`	
2	`test_ok`	
3	`test_failed`	
3	`test_in_porgress`	
4	`unknown`	

Sounding Mode

№	Token	Description
1	`silent`	
2	`audible`	
3	`unknown`	

Sound Playback

№	Token	Description
1	`stop`	
2	`play`	
3	`unknown`	

Periodic Inspection State

№	Token	Description
1	`inspection_not_required`	
2	`inspection_required`	
3	`unknown`	

Alarm Events(CO2)

№	Token	Description

1	`no_co2`
2	`co2_detected`
3	`unknown`

Alarm Events(Heat)

№	Token	Description
1	`heat_ok`	
2	`overheat_detected`	
3	`under_heat_detected`	
4	`unknown`	

Alarm Events(Siren)

№	Token	Description
1	`siren_inactive`	

2	`siren_active`
3	`unknown`

Alarm Events(Light)

№	Token	Description
1	`no_light`	
2	`light_detected`	
3	`unknown`	

Light changes

№	Token	Description
1	`no_light_color_transition`	
2	`light_color_transition_detected`	
3	`unknown`	

Temperature changes

№	Token	Description
1	`no_changes`	
2	`rapid_temperature_rise`	
3	`rapid_temperature_fail`	
4	`unknown`	

Water Leak Alarm

№	Token	Description
1	`no_water_leak`	
2	`water_leak_detected`	
3	`unknown`	

Water Filter Replacement Alarm

№	Token	Description
1	`water_filter_ok`	
2	`replace_water_filter`	
3	`unknown`	

Water Flow Alarm

№	Token	Description
1	`water_flow_ok`	
2	`water_flow_below_low_threshold`	
3	`water_flow_above_high_threshold`	
4	`water_flow_max`	
5	`unknown`	

Water Pressure Alarm

№	Token	Description
1	`water_pressure_ok`	
2	`water_pressure_below_low_threshold`	
3	`water_pressure_above_high_threshold`	
4	`water_pressure_max`	
5	`unknown`	

Water Temperature Alarm

№	Token	Description
1	`water_temperature_ok`	
2	`water_temperature_below_low_threshold`	
3	`water_temperature_above_high_threshold`	
4	`unknown`	

Water Level Alarm

№	Token	Description
1	`water_level_ok`	
2	`water_level_below_low_threshold`	
3	`water_level_above_high_threshold`	
4	`unknown`	

Pump State

№	Token	Description
1	`pump_idle`	
2	`pump_active`	
3	`pump_failure`	
4	`unknown`	

Alarm Sensor Events(Smoke)

№	Token	Description
1	`no_smoke`	
2	`smoke_detected`	
3	`unknown`	

Dust In Device(Smoke)

№	Token	Description
1	`no_dust`	
2	`dust_detected`	
3	`unknown`	

Alarm Sensor Events(Gas)

№	Token	Description

1	`no_gas`
2	`combustible_gas_detected`
3	`toxic_gas_detected`
4	`unknown`

Intrusion State

№	Token	Description
1	`no_intrusion`	
2	`intrusion`	
3	`unknown`	

Tampering Cover State

№	Token	Description
1	`no_tampering_cover`	

2	`tampering_cover`
3	`unknown`

Glass Breakage State

№	Token	Description
1	`no_glass_breakage`	
2	`glass_breakage`	
3	`unknown`	

Tampering Move State

№	Token	Description
1	`no_moving`	
2	`product_moved`	
3	`unknown`	

Tampering Impact State

№	Token	Description
1	`no_impact`	
2	`impact_detected`	
3	`unknown`	

Tampering Invalid Code State

№	Token	Description
1	`no_invalid_code`	
2	`invalid_code`	
3	`unknown`	

Barrier Operator Events

№	Token	Description

1	`barrier_closed`
2	`operating_closing`
3	`operating_unknown`
4	`operating_opening`
5	`barrier_opened`

Hardware State

№	Token	Description
1	`hardware_ok`	
2	`hardware_failure`	
3	`unknown`	

Software State

№	Token	Description

1	`software_ok`
2	`software_failure`
3	`unknown`

Emergency State

№	Token	Description
1	`state_ok`	
2	`emergency_shutoff`	
3	`unknown`	

Digital Input State

№	Token	Description
1	`digital_input_high_state`	
2	`digital_input_low_state`	

3	`digital_input_open`
4	`unknown`

Clock State

№	Token	Description
1	`no_state`	
2	`wake_up_alert`	
3	`time_ended`	
4	`unknown`	

Tilt Binary Sensor

№	Token	Description
1	`no_tilt`	
2	`tilt`	

3	`unknown`	

Door Lock Modes

№	Token	Description
1	`unsecured`	
2	`unsecured_with_timeout`	
3	`unsecured_for_inside`	
4	`unsecured_for_inside_with_timeout`	
5	`unsecured_for_outside`	
6	`unsecured_for_outside_with_timeout`	
7	`unknown`	
8	`secured`	

Thermostat Modes

№	Token	Description
1	`off`	
2	`heat`	
3	`cool`	
4	`auto`	
5	`aux`	
6	`resume`	
7	`fan_only`	
8	`furnace`	
9	`dry_air`	
10	`moist_air`	
11	`auto_change_over`	

`saving_heat`	
`saving_cool`	
`away_heat`	
`away_cool`	
`full_power`	
`special`	
`eco`	
`emergency_heating`	Use the electric heat strip constantly
`precooling`	Cooling a building in the early (cooler) part of the day, so that the thermal mass of the building decreases cooling needs in the later (hotter) part of the day

2 1	`sleep`	Mode for period, when user in bed

Thermostat Fan Modes

№	Token	Description
1	`fanmode_on_auto_low`	
2	`fanmode_on_low`	
3	`fanmode_on_auto_high`	
4	`fanmode_on_high`	
5	`fanmode_on_auto_medium`	
6	`fanmode_on_medium`	
7	`fanmode_on_circulation`	
8	`fanmode_on_humidity_circulation`	

9	`fanmode_on_lr_circulation`
10	`fanmode_on_ud_circulation`
11	`fanmode_on_quiet_circulation`
12	`fanmode_on_auto`
13	`fanmode_on`
14	`fanmode_off_auto_low`
15	`fanmode_off_low`
16	`fanmode_off_auto_high`
17	`fanmode_off_high`
18	`fanmode_off_auto_medium`
19	`fanmode_off_medium`
20	`fanmode_off_circulation`
21	`fanmode_off_humidity_circulation`

22	`fanmode_off_lr_circulation`
23	`fanmode_off_ud_circulation`
24	`fanmode_off_quiet_circulation`
25	`fanmode_off`

Thermostat Fan States

№	Token	Description
1	`idle_off`	
2	`running_low`	
3	`running_high`	
3	`running_medium`	
4	`circulation_mode`	
5	`humidity_circulation_mode`	

6	`right_left_circulation_mode`
7	`up_down_circulation_mode`
8	`quiet_circulation_mode`

Thermostat Setpoint Modes

№	Token	Description
1	`heating`	
2	`cooling`	
3	`furnace`	
4	`dry_air`	
5	`moist_air`	
6	`auto_change_over`	
7	`energy_save_heating`	

8	`energy_save_cooling`
9	`away_heating`
10	`away_cooling`
11	`full_power`
12	`unknown`

Thermostat Operating States

№	Token	Description
1	`idle`	
2	`heating`	
3	`cooling`	
4	`fan_only`	
5	`pending_heat`	

6	`pending_cool`
7	`vent_economizer`
8	`aux_heating`
9	`2nd_stage_heating`
10	`2nd_stage_cooling`
11	`2nd_stage_aux_heat`
12	`3rd_stage_aux_heat`
13	`2nd_stage_fan`
14	`3rd_stage_fan`

Shutter Commands

№	Token	Description
1	`off`	

2	`store`
3	`learn_upper`
4	`learn_lower`
5	`learn_favorite`
6	`change_roll_direction`

Shutter States

№	Token	Description
1	`idle`	
2	`learn_upper`	
3	`learn_lower`	
4	`learn_favorite`	
5	`change_roll_direction`	

6	`unknown`

Color Control Capability

№	Token	Description
1	`warm_white`	
2	`cold_white`	
3	`red`	
4	`green`	
5	`blue`	
6	`amber`	
7	`cyan`	
8	`purple`	
9	`indexed_color`	

Button Actions

№	Token	Description
1	`idle`	Default state
1	`press_1_time`	Shortly pressed 1 time
2	`released`	Button released
3	`held_down`	Button long press
4	`press_2_times`	Shortly pressed 2 times
5	`press_3_times`	Shortly pressed 3 times
6	`press_4_times`	Shortly pressed 4 times
7	`press_5_times`	Shortly pressed 5 times

Valve State

№	Token	Description
1	`valve_is_opened`	

2	`valve_is_closed`
3	`unknown`

Valve Current Alarm State

№	Token	Description
1	`current_ok`	
2	`current_below_low_threshold`	
3	`current_above_high_threshold`	
4	`current_max`	
5	`unknown`	

Rain State

№	Token	Description
1	`no_rain`	

2	`rain_detected`
3	`unknown`

Moisture State

№	Token	Description
1	`no_moisture`	
2	`moisture_detected`	
3	`unknown`	

Freeze State

№	Token	Description
1	`no_freeze`	
2	`freeze_detected`	
3	`unknown`	

Power State

№	Token	Description
1	`no_power_applied`	
2	`power_applied`	
3	`unknown`	

AC State

№	Token	Description
1	`disconnected`	
2	`connected`	
3	`unknown`	

Power Surge State

№	Token	Description

1	`no_surge`
2	`surge_detected`

Voltage Drop/Drift State

№	Token	Description
1	`no_drop_drift`	
2	`drop_drift_detected`	
3	`unknown`	

Over Current State

№	Token	Description
1	`no_over_current`	
2	`over_current_detected`	
3	`unknown`	

Over Voltage State

№	Token	Description
1	`no_over_voltage`	
2	`over_voltage_detected`	
3	`unknown`	

Over Load State

№	Token	Description
1	`no_over_load`	
2	`over_load_detected`	
3	`unknown`	

Load Error State

№	Token	Description

1	`no_load_error`
2	`load_error`
3	`unknown`

Battery Maintenance State

№	Token	Description
1	`battery_maintenance_ok`	
2	`replace_battery_soon`	
3	`replace_battery_now`	
4	`battery_fluid_is_low`	
5	`unknown`	

Battery Charging State

№	Token	Description

1	`no_charging`
2	`in_progress`
3	`unknown`

Appliance Status

№	Token	Description
1	`idle`	
2	`supplying_water`	
3	`boiling`	
4	`washing`	
5	`rinsing`	
6	`draining`	
7	`spinning`	

8	`drying`
9	`in_progress`
10	`unknown`

Program Status

№	Token	Description
1	`idle`	
2	`started`	
3	`in_progress`	
4	`completed`	
5	`unknown`	

Program Name

№	Token	Description

1	`target_temperature_setting`
2	`water_supply`
3	`boiling`
4	`washing`
5	`rinsing`
6	`draining`
7	`spinning`
8	`drying`
9	`fan`
10	`compressor`

Program Failed Status

№	Token	Description

1	`ok`
2	`failed`

Position

№	Token	Description
1	`leaving_bed`	
2	`sitting_on_bed`	
3	`lying_on_bed`	
4	`sitting_on_bed_edge`	
5	`unknown`	

Sleep Apnea Status

№	Token	Description
1	`not_detected`	

2	`low_breathe`
3	`no_breathe`
4	`unknown`

Sleep Stage

№	Token	Description
1	`not_sleeping`	
2	`dreaming`	
3	`light_sleep`	
4	`medium_sleep`	
5	`deep_sleep`	
6	`unknown`	

VOC Level Status

№	Token	Description
1	`clean`	
2	`slightly_poluted`	
3	`moderately_polluted`	
4	`highly_polluted`	
5	`unknown`	

Week days

№	Token	Description
1	`monday`	
2	`tuesday`	
3	`wednesday`	
4	`thursday`	

5	`friday`
6	`saturday`
7	`sunday`