

# ZIGBEE GATEWAY G1 HTTP REST API REFERENCE MANUAL



## Table of Contents

1. Overview .....	1
2. HTTP API .....	2
2.1. Query supported API versions .....	2
2.2. API version 1 .....	3
2.2.1. Data model .....	3
2.2.2. Device .....	3
2.2.3. Application .....	3
2.2.4. Service .....	4
2.2.5. Interface .....	4
2.2.6. Attribute .....	4
2.2.7. Scene .....	4
2.2.8. Group .....	4
2.2.9. Discover inventory/available devices .....	4
2.2.10. Set device state .....	7
2.2.11. Set device state via a batch update .....	8
2.2.12. Set device states via a groupcast .....	8
2.2.13. Invoke command .....	8
2.2.14. Invoke command via groupcast .....	9
2.2.15. Recall scene .....	9
2.2.16. Service types .....	10
2.2.17. Interface types .....	12
2.2.18. Attribute types .....	15
3. WebSocket API .....	20
3.1. Supported API versions .....	20
3.2. API version 1 .....	20
3.2.1. Protocol basics .....	20
3.2.2. Requests .....	21
3.2.3. Indications .....	25
4. Translated input commands .....	29
4.1. OnOffSwitch .....	29
4.2. LevelControlSwitch .....	29
4.3. ColorControlSwitch .....	30
4.4. UbisysAutomationSwitch .....	34
4.5. Scene .....	34
5. OAuth 2.0 flow .....	35
5.1. Flow description .....	35
5.1.1. Flow initiation .....	35
5.1.2. Redirection .....	35
5.1.3. Obtain tokens .....	35
5.1.4. Refresh access token .....	36

5.1.5. Bearer authentication .....	36
5.1.6. Flow example .....	36
6. Outlook .....	39
7. Revision History .....	40
8. Contact .....	41

## 1. Overview

The ubisys Zigbee Gateway G1 HTTP REST API provides access to the ubisys G1 and its connected zigbee devices via a easy to use Representational State Transfer (REST) Application Programming Interface (API) accessible via web standard Hypertext Transfer Protocol (HTTP). The API hides the details of the underlying zigbee technology and provides an abstracted view. It is itself based on the ubisys proprietary Smart Facility Service API. Compared to the Smart Facility Service API, the HTTP REST API is far easier to use and is also supported by a publicly routed API gateway proxy service available at the URL [c2f.ubisys.de](http://c2f.ubisys.de), which is very hand for cloud-to-device and cloud-to-cloud interactions. In addition, the G1 has the ability to serve HTTP requests originating from the local network, without cloud detour.

The API was initially implemented to support integration with Amazon's Alexa, but is intended for generic use as well. Currently, only a subset of functions is supported, with more functionality to be mapped at a later stage. If you feel that something is missing for your particular use case, please contact ubisys support.

## 2. HTTP API

The API is accessible either via direct HTTP access to the gateway, in which case accessibility is dependent on the network topology (e.g. firewall/NAT device in a typical home setup) or via the ubisys Cloud-to-Facility (C2F) API proxy, via which global access is possible.

Access via the C2F API proxy uses the root URL: <https://c2f.ubisys.de/api/>. Authentication must be implemented via OAuth 2.0 bearer tokens, c.f. [Chapter 5](#). Direct access is possible via the gateway's hostname/IP address, depending on the network configuration, e.g. <http://ubisys-g1-nnnn/api/>. Secured access via TLS/https can be provided upon request. Authentication must be implemented via bearer tokens, for which a user-configurable configuration interface can be provided upon request. Currently, only API version 1 ("v1") is defined. More versions may be added at a later stage if needed. NB: the API is implemented within the G1 firmware. That is, any new versions will be implemented in newer firmware releases. As such, different target gateways might support different API versions.

Payloads for various API calls are encoded in JavaScript Object Notation (JSON). It is important that any client implementation will ignore further fields, which may be introduced in future revisions of the same API version.

### 2.1. Query supported API versions

GET <https://c2f.ubisys.de/api/>

GET <http://ubisys-g1-nnnn/api/>



The Authentication HTTP header must be added and contain the "bearer" scheme as well as a valid bearer token, c.f. [Section 5.1.5](#).

Response:

A JSON information object is returned.



Prior to G1 firmware 2.0.6, a JSON array instead of an object is returned.

```
{
  "supported_api_versions": ["v1"]
}
```

API version 1 (v1) is currently the only implemented version.

## 2.2. API version 1

### 2.2.1. Data model

The data model consists of devices, applications, services, interfaces, attributes and scenes.

#### Device

A device usually resembles a single physical device. It is identified by a unique identifier and may contain multiple applications as well as per-device metadata. It contains further applications.

#### Application

An application usually represents a single logical unit within a device. It is identified by an identifier, which is unique within its device. It contains further per-application metadata and hosts further services.

#### Service

A service encapsulates a single logical function of an application, e.g. "Power" or "Light".

#### Interface

An interface encapsulates a control characteristic of a Service, e.g. On/Off state or Level.

#### Attribute

An attribute represents a single logical state of an interface, e.g. the on state or current level.

#### Command

Several interfaces expose commands, which provide functionality which cannot be exposed in a REST-style via attributes, e.g. **up**, **down** and **stop** commands.

#### Scene

A scene represents a zigbee scene, which allows setting a desired device target state on a group of devices and endpoints. This allows to e.g. set all lights in a certain room to a specific brightness with a single command or to set certain color scheme for lights in a room.

#### Group

A group represents a zigbee group, i.e. a collection of devices addressable via a single groupcast.

### 2.2.2. Device

Name	Type	Description
identifier	String	Unique identifier of this device. This identifier is not encoded as a separate field, but as the object key.
description	String	User-assigned description for this device.
model	String	Model string (vendor assigned)
vendor	String	Vendor string (vendor assigned)
applications	Map	Contains the applications for this device. Maps the application identifier to the application definition.

### 2.2.3. Application

Name	Type	Description
identifier	String	Identifier of this application. Unique within the hosting device. This identifier is not encoded as a separate field, but as the object key.
description	String	User-assigned description

Name	Type	Description
model	String	Relative URI path to this application
applications	Map	Contains the services for this application. Maps the service type to the service definition.

## 2.2.4. Service

Name	Type	Description
type	String	The type of this service. Unique within the hosting application. The service type is not encoded as a separate field, but at the object key in the application's services map.
href	String	Relative URI path to this service
interfaces	Map	Contains the interfaces for this service. Maps the interface type to the interface definition.

## 2.2.5. Interface

Name	Type	Description
type	String	The type of this interface. Unique within the hosting service. The interface type is not encoded as a separate field, but at the object key in the services' interfaces map.
href	String	Relative URI path to this interface
interfaces	Map	Contains the attributes for this interface. Maps the attribute type to the interface definition.

## 2.2.6. Attribute

Name	Type	Description
type	String	The type of this attribute. Unique within the hosting interface. The attribute type is not encoded as a separate field, but at the object key in the interface's attribute map.
href	String	Relative URI path to this attribute
value	Object	Current value of the attribute. Value may be null if the value is unknown. The actual type depends on the attribute type.

## 2.2.7. Scene

Name	Type	Description
description	String	User-assigned description for this scene
href	String	Relative URI path to this scene

## 2.2.8. Group

Name	Type	Description
description	String	User-assigned description for this group
href	String	Relative URI path to this group
members	Array of Strings	List of hrefs to group members

## 2.2.9. Discover inventory/available devices

GET <https://c2f.ubisys.de/api/v1/>

GET <http://ubisys-g1-nnnn/api/v1/>



The Authentication HTTP header must be added and contain the "bearer" scheme as well as a valid bearer token, c.f. [Section 5.1.5](#).

Returns a JSON representation of the known devices.

Example response:

```
{
  "devices": {
    "d13": {
      "applications": {
        "e1": {
          "description": "Hallway Light",
          "href": "/v1/devices/d13/e1",
          "services": {
            "Light": {
              "href": "/v1/devices/d13/e1/Light",
              "interfaces": {
                "Level": {
                  "attributes": {
                    "CurrentLevel": {
                      "href": "/v1/devices/d13/e1/Light/Level/CurrentLevel",
                      "value": 0.39
                    }
                  }
                }
              },
              "OnOff": {
                "attributes": {
                  "OnState": {
                    "href": "/v1/devices/d13/e1/Light/OnOff/OnState",
                    "value": false
                  }
                }
              }
            }
          }
        }
      },
      "description": "Dimmer Hallway",
      "model": "D1-R (5603)",
      "vendor": "ubisys"
    },
    "d2a": {
      "applications": {
        "e1": {
          "description": "Entrance Light",
          "href": "/v1/devices/d2a/e1",
          "services": {
            "Power": {
              "href": "/v1/devices/d2a/e1/Power",
              "interfaces": {
                "OnOff": {
                  "attributes": {
                    "OnState": {
                      "href": "/v1/devices/d2a/e1/Power/OnOff/OnState",
                      "value": true
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```



```

    }
  },
  "description": "Entrance S1-R",
  "model": "S1-R (5601)",
  "vendor": "ubisys"
},
"d36": {
  "applications": {
    "e1": {
      "description": "Window Blind",
      "hardware_address": "00:1f:ee:00:00:00:08:29/1",
      "href": "/v1/devices/d36/e1",
      "services": {
        "WindowCoveringControl": {
          "href": "/v1/devices/d36/e1/WindowCoveringControl",
          "interfaces": {
            "WindowCovering": {
              "attributes": {
                "LiftPercentage": {
                  "href":
"/v1/devices/d36/e1/WindowCoveringControl/WindowCovering/LiftPercentage",
                  "value": 89
                },
                "TiltPercentage": {
                  "href":
"/v1/devices/d36/e1/WindowCoveringControl/WindowCovering/TiltPercentage",
                  "value": 0
                }
              }
            },
            "commands": {
              "down": {
                "href":
"/v1/devices/d36/e1/WindowCoveringControl/WindowCovering/down"
              },
              "stop": {
                "href":
"/v1/devices/d36/e1/WindowCoveringControl/WindowCovering/stop"
              },
              "up": {
                "href":
"/v1/devices/d36/e1/WindowCoveringControl/WindowCovering/up"
              }
            },
            "href": "/v1/devices/d36/e1/WindowCoveringControl/WindowCovering",
            "type": "TiltBlindLiftAndTilt"
          }
        }
      }
    }
  }
},
  "e2": {
    "description": "Window Blind Inputs",
    "hardware_address": "00:1f:ee:00:00:00:08:29/2",
    "href": "/v1/devices/d36/e2",
    "services": {
      "Switch": {
        "href": "/v1/devices/d36/e2/Switch",
        "interfaces": {

```

```

        "UbisysAutomationSwitch": {
            "attributes": {},
            "href": "/v1/devices/d36/e2/Switch/UbisysAutomationSwitch"
        }
    },
    "description": "",
    "hardware_address": "00:1f:ee:00:00:12:07:28",
    "href": "/v1/devices/d36",
    "model": "J1 (5502)",
    "vendor": "ubisys"
}
},
"groups": {
    "g1": {
        "description": "Hallway lights",
        "href": "/v1/groups/g1"
    },
},
"scenes": {
    "s5": {
        "description": "Hallway Light On/dimmed",
        "href": "/v1/scenes/s5"
    }
}
}
}

```

## 2.2.10. Set device state

To update a device state, e.g. brightness (level), the corresponding attribute must be written to via a POST request.

POST [https://c2f.ubisys.de/api/v1/devices/device\\_id/application\\_id/service/interface/attribute](https://c2f.ubisys.de/api/v1/devices/device_id/application_id/service/interface/attribute)

POST [http://ubisys-g1-nnnn/api/v1/devices/device\\_id/application\\_id/service/interface/attribute](http://ubisys-g1-nnnn/api/v1/devices/device_id/application_id/service/interface/attribute)



The Authentication HTTP header must be added and contain the "bearer" scheme as well as a valid bearer token, c.f. [Section 5.1.5](#).

The Payload is the JSON-encoded update, e.g. { "value": <new-value> }.

Example:

POST <https://c2f.ubisys.de/api/v1/devices/d13/e1/Light/OnOff/OnState>

with the following payload:

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

### 2.2.11. Set device state via a batch update

In several scenarios, it may either be desired or required to update more than one attribute in a single transaction.

This is supported for Color Control to update Hue and Saturation in a single batch update. It is on the other hand required for Color Control to update X and Y, which cannot be written separately.

For a batch update, a POST must be done onto the interface with a JSON payload mapping the attribute to the corresponding value to be set.

Example:

POST <https://c2f.ubisys.de/api/v1/devices/d14/e1/Light/ColorControl>

with the following payload:

```
{ "X": 0.7, "Y": 0.2 }
```

### 2.2.12. Set device states via a groupcast

Attribute can be updated via a groupcast similarly as an update for a single attribute. This would allow updating a group of devices with a single command.

POST [https://c2f.ubisys.de/api/v1/groups/group\\_id/service/interface/attribute](https://c2f.ubisys.de/api/v1/groups/group_id/service/interface/attribute)

POST [http://ubisys-g1-nnnn/api/v1/groups/group\\_id/service/interface/attribute](http://ubisys-g1-nnnn/api/v1/groups/group_id/service/interface/attribute)

A batch update is done accordingly:

POST [https://c2f.ubisys.de/api/v1/groups/group\\_id/service/interface](https://c2f.ubisys.de/api/v1/groups/group_id/service/interface)

POST [http://ubisys-g1-nnnn/api/v1/groups/group\\_id/service/interface](http://ubisys-g1-nnnn/api/v1/groups/group_id/service/interface)

Updates via groupcasts are not supported to emulated attributes which depend on the device state, i.e. the WindowCovering LiftPercentageDelta and TiltPercentageDelta attributes.

Furthermore, several attributes in the ColorControl interface are emulated if the device does not support them, e.g. ColorTemperature as well as Hue and Saturation. This emulation does not work via a groupcast. Instead, the zigbee/ZCL command is sent as-is, independently of the capabilities of the group member devices.

### 2.2.13. Invoke command

Commands are invoked via a POST to the provided href, currently without any payload. In the future, commands may accept a payload.

POST [https://c2f.ubisys.de/api/v1/devices/device\\_id/application\\_id/service/interface/command](https://c2f.ubisys.de/api/v1/devices/device_id/application_id/service/interface/command)

POST [http://ubisys-g1-nnnn/api/v1/devices/device\\_id/application\\_id/service/interface/command](http://ubisys-g1-nnnn/api/v1/devices/device_id/application_id/service/interface/command)

### 2.2.14. Invoke command via groupcast

Commands via groupcast are invoked accordingly via a POST to the group instead of device/application:

POST [https://c2f.ubisys.de/api/v1/groups/group\\_id/service/interface/command](https://c2f.ubisys.de/api/v1/groups/group_id/service/interface/command)

POST [http://ubisys-g1-nnnn/api/v1/groups/group\\_id/service/interface/command](http://ubisys-g1-nnnn/api/v1/groups/group_id/service/interface/command)

### 2.2.15. Recall scene

POST [https://c2f.ubisys.de/api/v1/scenes/scene\\_id](https://c2f.ubisys.de/api/v1/scenes/scene_id)

POST [http://ubisys-g1-nnnn/api/v1/scenes/scene\\_id](http://ubisys-g1-nnnn/api/v1/scenes/scene_id)

A JSON payload might be provided to override the default transition time.

Example:

POST <http://ubisys-g1-nnnn/api/v1/scenes/s5>

```
{"transition_time": 5}
```

The transition time is to be given in seconds. The internal zigbee resolution is 0.1 seconds.

## 2.2.16. Service types

This section describes the currently defined service types. All services are identified by their service type identifier, e.g. “Light” or “Power”. More services will be added in the future.

### 2.2.16.1. Light

Represents a light or lamp, which can be switched on and off and potentially dimmed or colour-controlled.

Provided interfaces:

- OnOff
- Level (optional)
- ColorControl (optional)

### 2.2.16.2. Power

Represents a switchable power source, e.g. a switchable outlet or similar of unspecified nature.

Provided interfaces:

- OnOff

### 2.2.16.3. WindowCoveringControl

Represents a window covering device, e.g. controller for shutters, awnings, screens and blinds.

Provided interfaces:

- WindowCovering

### 2.2.16.4. Sensor

Represents a sensor or group of sensors with the capability to measure ambient parameters.

Provided interfaces:

- TemperatureSensor
- RelativeHumiditySensor
- OccupancySensor
- IlluminanceSensor

The actually provided interfaces depend on the sensor capabilities and may be one, some or all of the above.

### 2.2.16.5. Switch

Represent a switch and allows to receive events on commands generated by the switch.

Events are currently only receivable via the WebSocket protocol, not via HTTP.

### 2.2.16.6. IASZone

Represents an IAS (Intruder Alarm System) Zone. Different types are supported, represented by dedicated interface types:

- CIE
- MotionSensor
- ContactSwitch
- FireSensor
- WaterSensor
- CarbonMonoxideSensor
- GlassBreakSensor

The auxillary interface **ZoneStatus** is always presents and decodes status flags common to all IAS zone types.

### 2.2.17. Interface types

This section describes the currently defined interface types. All interfaces are identified by their interface type identifier, e.g. "OnOff" or "Level". More interfaces will be added in the future to accommodate further services and device types.

#### 2.2.17.1. OnOff

Provided attributes:

- OnState

#### 2.2.17.2. Level

Provided attributes:

- CurrentLevel

#### 2.2.17.3. ColorControl

Provided attributes:

- ColorTemperature
- Hue
- Saturation
- RGB
- X
- Y

#### 2.2.17.4. WindowCovering

Provided attributes:

- LiftPercentage
- TiltPercentage
- LiftPercentageDelta
- TiltPercentageDelta

These attributes are exposed based on the type of the WindowCovering device. Following table provides more information on supported types.

<b>WindowCovering Type</b>	<b>LiftPercentage/ LiftPercentageDelta</b>	<b>TiltPercentage/ TiltPercentageDelta</b>
Rollershade	Supported	-
Rollershade 2-Motor	Supported	-
Rollershade Exterior	Supported	-
Rollershade Exterior - 2-Motor	Supported	-
Drapery	Supported	-
Awning	Supported	-
Shutter	-	Supported
Tilt Blind - Tilt Only	-	Supported

WindowCovering Type	LiftPercentage/ LiftPercentageDelta	TiltPercentage/ TiltPercentageDelta
Tilt Blind - Lift and Tilt	Supported	Supported
Projector Screen	Supported	-

Provided commands:

+ up + down + stop

#### 2.2.17.5. TemperatureSensor

Provided attribute: ▪ Temperature

#### 2.2.17.6. RelativeHumiditySensor

Provided attribute: ▪ RelativeHumidity

#### 2.2.17.7. OccupancySensor

Provided attribute: ▪ Occupancy

#### 2.2.17.8. IlluminanceSensor

Provided attribute: ▪ Illuminance

#### 2.2.17.9. Switch

The following Switch (i.e. input) interfaces are implemented:

- OnOffSwitch
- LevelControlSwitch
- ColorControlSwitch
- UbisysAutomationSwitch

Represents an on/off, level controllable (dimnable) or color-controllable switch or the vendor-specific ubisys automation switch.

These interfaces provide no attributes, but are capable of generating the **input-event** indication for commands received from the switch.

The requirement is that a binding exists from the switch to the gateway to endpoint 16.

Please refer to [\[input-event-indication\]](#) for further details.

#### 2.2.17.10. CIE

Resembles an Intruder Alert System (IAS) Control and Indicating Equipment (CIE) zone.

Provided attributes: SystemAlarm, ZoneStatus

#### 2.2.17.11. MotionSensor

Resembles a standard motion sensor.



Provided attributes: Intrusion, Presence, ZoneStatus

#### **2.2.17.12. ContactSwitch**

Resembles a contact switch, e.g. a door contact. The meaning of the Portal1 and Portal2 attributes is vendor-defined and not standardized.

Provided attributes: Portal1, Portal2, ZoneStatus

#### **2.2.17.13. FireSensor**

Provided attributes: Fire, ZoneStatus

#### **2.2.17.14. WaterSensor**

Provided attributes: WaterOverflow, ZoneStatus

#### **2.2.17.15. CarbonMonoxideSensor**

Provided attributes: COIndication, ZoneStatus

#### **2.2.17.16. GlassBreakSensor**

Provided attributes: GlassBreakage, ZoneStatus

### 2.2.18. Attribute types

This section describes the currently defined attribute types. All attributes are identified by their attribute type identifier, e.g. "OnState". More attributes will be added in the future to accommodate further services and device types. Each attribute corresponds to data type for its current value, e.g. a boolean represents and OnState, with false corresponding to "Off" and true corresponding to "On".

#### 2.2.18.1. OnState

Value: bool

Supported operations: read/write

#### 2.2.18.2. CurrentLevel

Value: number (range 0..1)

Supported operations: read/write

#### 2.2.18.3. ColorTemperature

Indicates the current color temperature in K

Supported operations: read/write

If the target device does not support direct color temperature control, a color temperature is converted to a color. In this case, reading the color temperature is not supported. This feature is not available via a groupcast.

#### 2.2.18.4. Hue

Indicates the Hue in degrees [0..360]

Supported operations: read/write

This attribute will be emulated if the underlying device does not support the Hue/Saturation color space. This emulation is not available via a groupcast.

#### 2.2.18.5. Saturation

Indicates the saturation as a decimal number [0..1]

Supported operations: read/write

This attribute will be emulated if the underlying device does not support the Hue/Saturation color space. This emulation is not available via a groupcast.

#### 2.2.18.6. RGB

The color in RGB notation as a string, similar to what HTML uses. R, G and B are given as a two-digit hex number [0..ff] and concatenated, e.g. "ff0000" is plain red. The level of the color in RGB notation is not considered, i.e. "444444" and "ffffff" both result in the same white color. The level is to be controlled via the Level interface.

Supported operations: read/write

#### 2.2.18.7. X

X coordinate of the color in the CIE 1931 as a decimal number in the range of [0..1]

Supported operations: read; write only as a batch update by writing X and Y in conjunction. Please

refer to [Section 2.2.11](#) for details.

#### 2.2.18.8. Y

Y coordinate of the color in the CIE 1931 as a decimal number in the range of [0..1]

Supported operations: read; write only as a batch update by writing X and Y in conjunction. Please refer to [Section 2.2.11](#) for details.

#### 2.2.18.9. LiftPercentage

Value: number (range 0 ... 100)

Supported operations: read/write

#### 2.2.18.10. TiltPercentage

Value: number (range 0 ... 100)

Supported operations: read/write

#### 2.2.18.11. LiftPercentageDelta

Value: number (range -100 ... 100)

Supported operations: write

This attribute allows to provide a delta shift value in the range of -100 to +100, which will then result in updating the LiftPercentage accordingly. For example, if LiftPercentage is 30, LiftPercentageDelta 20 will result in LiftPercentage 50. Not supported as a groupcast.

#### 2.2.18.12. TiltPercentageDelta

Value: number (range -100 ... 100)

Supported operations: write

This attribute allows to provide a delta shift value in the range of -100 to +100, which will then result in updating the TiltPercentage accordingly. For example, if TiltPercentage is 60, TiltPercentageDelta -50 will result in TiltPercentage 10. Not supported as a groupcast.

#### 2.2.18.13. Temperature

Value: number

Unit: °C

Supported operations: read

The possible range is -273.15 °C up to 327.67 °C. If the sensor provides its measuring range, it will be made available via the **min** and **max** keys.

#### 2.2.18.14. RelativeHumidity

Value: number

Unit: %

Supported operations: read

The possible range is 0% to 100%. If the sensor provides its measuring range, it will be made available via the **min** and **max** keys.

### 2.2.18.15. Occupancy

Value: bool

Supported operations: read

A value of true indicates occupied, a value of false unoccupied.

### 2.2.18.16. Illuminance

Value: number

Unit: lx

Supported operations: read

The possible range is 1 lx to 3.576 Mlx. If the sensor provides its measuring range, it will be made available via the **min** and **max** keys.

### 2.2.18.17. IAS Zone state attributes

The following read-only attributes of type bool are defined:

- SystemAlarm
- Intrusion
- Presence
- Portal1
- Portal2
- Fire
- WaterOverflow
- COIndication
- GlassBreakage

Their meaning depends on the IAS zone type.

### 2.2.18.18. ZoneStatus

Value: array of strings, indicating the zone status

The following status flags are decoded and may be present in the array:

- Tampered
- LowBattery
- Trouble
- ACMainsFailure
- TestMode
- BatteryDefect

## 2.2.18.19. Examples

Create bearer token via ubisys gateway G1 web-interface

- Access the gateway's web interface in a browser using either the IP address of the gateway or the URL <http://ubisys-g1-serialnumber.local>, e.g. <http://ubisys-g1-2750.local>.
- Select "Settings" → "Integrations"
- In the "Authentication Tokens" section, click on the edit icon and then on the "ADD TOKEN" button
- Complete the fields as follows:

Description	Arbitrary
Scopes	facility-discover, facility-control, facility-state
Expiration date	Optional - can be left blank

- Click on the "ADD" button
- A new dialog opens displaying the bearer token information



For security reasons, this is the only instance where this token will be displayed in plain text. It is crucial that you take note of it and preserve it carefully for future use.

Application examples with the use of curl

Discover inventory/available devices using cURL (Chapter 2.2.9):

```
curl -H "Authorization: Bearer 1ac026f2b9bc75b277a22899796f0dc2" http://ubisys-g1-2750.local/api/v1/
```

Switch of a simple On/Off device:

```
Turn on -> curl -v -H "Authorization: Bearer 1ac026f2b9bc75b277a22899796f0dc2" -d '{"value":true}' http://ubisys-g1-2750.local/api/v1/devices/d12/e1/Power/OnOff/OnState
```

```
Turn off -> curl -v -H "Authorization: Bearer 1ac026f2b9bc75b277a22899796f0dc2" -d '{"value":false}' http://ubisys-g1-2750.local/api/v1/devices/d12/e1/Power/OnOff/OnState
```

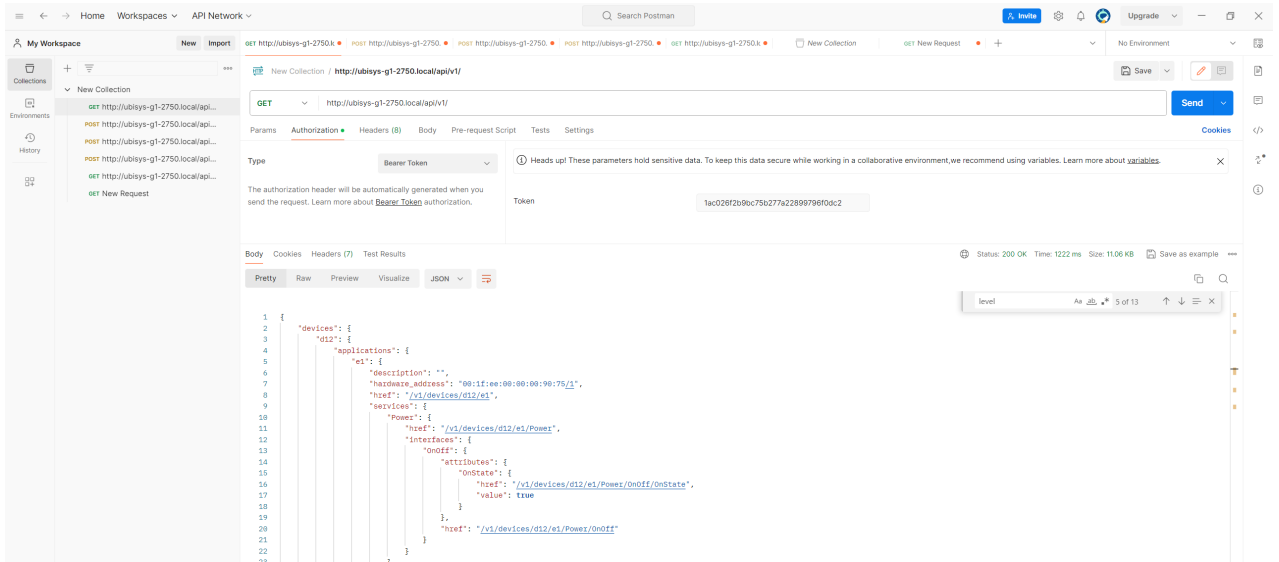
Change brightness of a dimable device:

```
Move to 35% -> curl -v -H "Authorization: Bearer 1ac026f2b9bc75b277a22899796f0dc2" -d '{"value":0.35}' http://ubisys-g1-2750.local/api/v1/devices/d16/e1/Light/Level/CurrentLevel
```

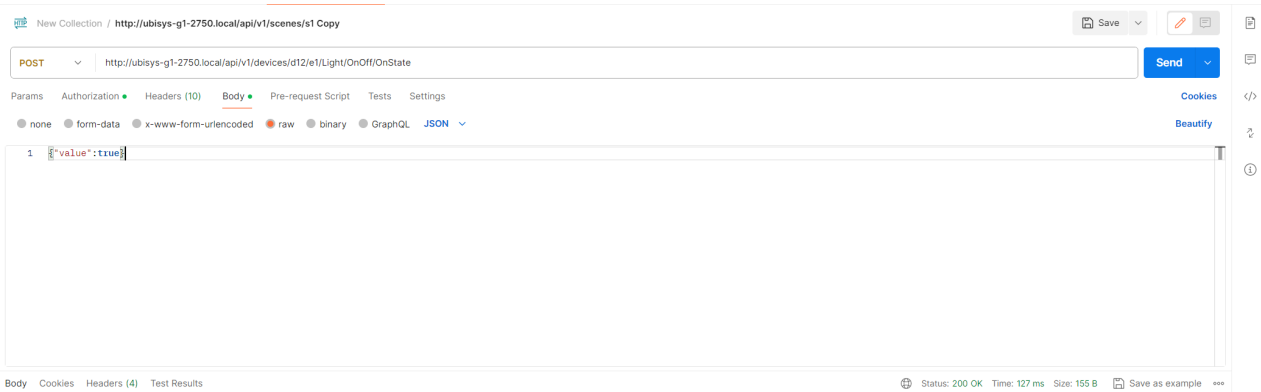
Instead of using curl it is possible to use an application like Postman:

We are doing the same queries as we did with curl before:

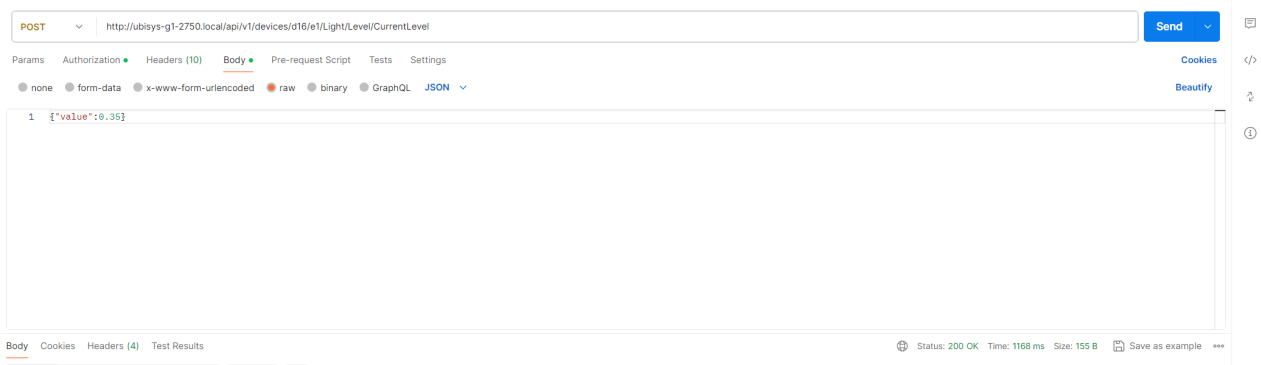
Device discovery



## Turn on-off device on



## Change brightness to 35%



## 3. WebSocket API

### 3.1. Supported API versions

There is currently no dedicated mechanism to query the supported WebSocket API versions. The currently supported version is **v1**, corresponding to the v1 HTTP API. Version **v1** was introduced in the ubisys G1, firmware version 4.1. Availability of the WebSocket API can be queried by trying to establish a WebSocket connection. If the WebSocket protocol is not available, the handshake will fail.

### 3.2. API version 1

The WebSocket API is reachable via the following URIs (or variations thereof, depending on the network configuration):

- `ws://ubisys-g1-nnnn/api/v1/`
- `wss://ubisys-g1-nnnn/api/v1/` (if https is configured on the gateway)
- `wss://c2f.ubisys.de/api/v1/`

No WebSocket subprotocol is currently used.

For authentication, the bearer token must be provided via the "Authorization" header during the WebSocket handshake.

The WebSocket API makes use of the same data model as the HTTP API with the difference that the **href** values don't contain the "/v1" prefix, as this is part of the protocol session.

#### 3.2.1. Protocol basics

The WebSocket protocol is based on the exchange of JSON objects. Each JSON message must be encapsulated in a WebSocket text message. Each message may consist of several fragments. Message size is limited to 10 kB (for requests).

The protocol consists of requests and responses, along with indications. Each request is answered with a response. An indication might be sent unsolicited after the client sent a previous subscribe request.

##### 3.2.1.1. Requests

A request object must contain the **request** field with the actual request to invoke. A request payload, if defined, must be contained within the **payload** field. An optional **correlation\_token** may be added, which maybe an arbitrary JSON value (e.g. number, string, array, object etc), which will be echoed back in the response.

##### 3.2.1.2. Response

A response object contains the **status** field with the status of the request. The **correlation\_token** of the request is echoed back, if any. An optional response payload is contained within the **payload** field, if defined.

### 3.2.1.3. Indication

An indication object contains the **indication** field, specifying the type of indication. The **payload** field contains the payload of the indication, if defined.

## 3.2.2. Requests

### 3.2.2.1. discover

Returns the current system inventory. The response payload is the same information as returned as by a HTTP GET to `/api/v1/`. Please refer to [Section 2.2.9](#) for further details.

Sample request:

```
{"request": "discover"}
```

Sample response (formatted for better readability):

```
{
  "status": "success",
  "payload": {
    "devices": {
      "d14": {
        "applications": {
          "e1": {
            "description": "Entrance light",
            "hardware_address": "00:1f:ee:00:00:00:95:09/1",
            "href": "/devices/d14/e1",
            "services": {
              "Light": {
                "href": "/devices/d14/e1/Light",
                "interfaces": {
                  "ColorControl": {
                    "attributes": {
                      "ColorTemperature": {
                        "href":
"/devices/d14/e1/Light/ColorControl/ColorTemperature"
                      },
                      "Hue": {
                        "href": "/devices/d14/e1/Light/ColorControl/Hue",
                        "value": 348
                      },
                      "RGB": {
                        "href": "/devices/d14/e1/Light/ColorControl/RGB",
                        "value": "ff002f"
                      },
                      "Saturation": {
                        "href": "/devices/d14/e1/Light/ColorControl/Saturation",
                        "value": 1.0
                      },
                      "X": {
                        "href": "/devices/d14/e1/Light/ColorControl/X",
                        "value": 0.7
                      },
                      "Y": {
```





### 3.2.2.2.1. Single-attribute update

A payload containing the **href** of the attribute to update along with the intended value must be sent.

Sample request:

```
{
  "request": "update",
  "payload": {
    "href": "/devices/d14/e1/Light/Level/CurrentLevel",
    "value": 0.1
  }
}
```

Sample success response:

```
{"status": "success"}
```

Sample error response:

```
{"payload": ["reason", "NWK:ROUTE_ERROR"], "status": "error"}
```

### 3.2.2.2.2. Batch update

A payload containing the **href** of the interface to update along with a field **values**, containing the mapping of attributes to intended values must be sent.

```
{
  "request": "update",
  "payload": {
    "href": "/devices/d14/e1/Light/ColorControl",
    "values": {
      "X": 0.2,
      "Y": 0.7
    }
  }
}
```

Sample success response:

```
{"status": "success"}
```

### 3.2.2.3. invoke

Invokes a command with the provided href.

Sample request:

```
{
  "request": "invoke",
  "payload": {
    "href": "/devices/d36/e1/WindowCoveringControl/WindowCovering/up"
  }
}
```

Sample success response:

```
{"status": "success"}
```

### 3.2.2.4. recall-scene

Recalls a scene.

Sample request:

```
{
  "request": "recall-scene",
  "payload": {
    "href": "/scenes/s5"
  }
}
```

Sample request with transition time override:

```
{
  "request": "recall-scene",
  "payload": {
    "href": "/scenes/s5",
    "transition_time": 0.5
  }
}
```

Sample response:

```
{"status": "success"}
```

### 3.2.2.5. subscribe-attributes

Subscribes for attribute updates.

Sample request:

```
{"request": "subscribe-attributes"}
```

Sample response:

```
{"status": "success"}
```

### 3.2.2.6. subscribe-inventory

Subscribes for inventory updates.

Sample request:

```
{"request": "subscribe-inventory"}
```

Sample response:

```
{"status": "success"}
```

### 3.2.2.7. subscribe-input-events

Subscribes for input events.

Sample request:

```
{ "request": "subscribe-input-events" }
```

Sample response:

```
{"status": "success"}
```

This will request to send **input-event** indications on this connection.

## 3.2.3. Indications

### 3.2.3.1. attribute-update

The indication is sent if it was previously subscribed to via the **subscribe-attribute** request on change of an attribute.

The payload contains the **href** of the change attribute alongside its new **value**.

Sample indication:

```
{
  "indication": "attribute-update",
  "payload": {
    "href": "/devices/d14/e1/Light/Level/CurrentLevel",
    "value": 0.311
  }
}
```

### 3.2.3.2. inventory-update

The indication is sent if it was previously subscribed for via the `subscribe-inventory` request whenever the system inventory, e.g. the set of devices or their associated names etc. was changed.

Sample indication:

```
{
  "indication": "inventory-update",
  "payload": {
    "devices": {
      "d14": {
        "applications": {
          "e1": {
            "description": "Entrance light",
            "hardware_address": "00:1f:ee:00:00:00:95:09/1",
            "href": "/devices/d14/e1",
            "services": {
              "Light": {
                "href": "/devices/d14/e1/Light",
                "interfaces": {
                  "ColorControl": {
                    "attributes": {
                      "ColorTemperature": {
                        "href":
"/devices/d14/e1/Light/ColorControl/ColorTemperature"
                      },
                      "Hue": {
                        "href":
"/devices/d14/e1/Light/ColorControl/Hue",
                        "value": 119
                      },
                      "RGB": {
                        "href":
"/devices/d14/e1/Light/ColorControl/RGB",
                        "value": "00ff00"
                      },
                      "Saturation": {
                        "href":
"/devices/d14/e1/Light/ColorControl/Saturation",
                        "value": 1.0
                      },
                      "X": {
                        "href":
"/devices/d14/e1/Light/ColorControl/X",
                        "value": 0.2
                      },
                      "Y": {
                        "href":
"/devices/d14/e1/Light/ColorControl/Y",
                        "value": 0.7
                      }
                    }
                  },
                  "href": "/devices/d14/e1/Light/ColorControl"
                }
              },
              "Level": {
                "attributes": {
                  "CurrentLevel": {
```

```

      "href":
"/devices/d14/e1/Light/Level/CurrentLevel",
      "value": 0.098
    },
    },
    "href": "/devices/d14/e1/Light/Level"
  },
  "OnOff": {
    "attributes": {
      "OnState": {
        "href":
"/devices/d14/e1/Light/OnOff/OnState",
        "value": true
      }
    }
  },
  "href": "/devices/d14/e1/Light/OnOff"
}
}
}
}
},
"description": "LD6 Entrance",
"hardware_address": "00:1f:ee:00:00:00:95:09",
"href": "/devices/d14",
"model": "LD6",
"vendor": "ubisys"
}
}
}
}
}

```

### 3.2.3.3. input-event

This indication is generated whenever there is a command received from a switch device and a `subscribe-input-events` command was previously executed. The payload contains the translated ZCL command, i.e. the command identifier and the parsed payload.

Sample indication:

```

{
  "indication": "input-event",
  "payload": {
    "command": {
      "id": "move-with-on-off",
      "payload": {
        "move-mode": 1,
        "rate": 50,
        "raw": "0132"
      }
    },
    "href": "/devices/d25/e1/Switch/LevelControlSwitch"
  }
}

```

Sample indication for a scene recall:

```
{
  "indication": "input-event",
  "payload": {
    "command": {
      "id": "recall-scene",
      "payload": {
        "group-id": 2,
        "raw": "020000",
        "scene-id": 0
      }
    },
    "href": "/scenes/s5"
  }
}
```

`href` contains the reference to the instance on which the command was received. `command` contains the parsed ZCL command, with `id` being the command identifier. The `raw` identifier in the `payload` sub-object contains the raw ZCL command payload, as received, as a hex string. All other fields depend on the actual command.

See [Chapter 4](#) for reference on translated commands and their payload.

## 4. Translated input commands

### 4.1. OnOffSwitch

- off
- on
- toggle
- off-with-effect  
Payload:
  - effect-identifier
  - effect-variant
- on-with-recall-global-scene
- on-with-timed-off  
Payload:
  - on-off-control
  - on-time
  - off-wait-time

### 4.2. LevelControlSwitch

- move-to-level  
Payload:
  - level
  - transition-time
  - options-mask
  - options-override
- move  
Payload:
  - move-mode
  - rate
  - options-mask
  - options-override
- step  
Payload:
  - step-mode
  - step-size
  - transition-time
  - options-mask
  - options-override



- stop
  - Payload:
    - options-mask
    - options-override
- move-to-level-with-on-off
  - Payload:
    - level
    - transition-time
    - options-mask
    - options-override
- move-with-on-off
  - Payload:
    - move-mode
    - rate
    - options-mask
    - options-override
- step-with-on-off
  - Payload:
    - step-mode
    - step-size
    - transition-time
    - options-mask
    - options-override

### 4.3. ColorControlSwitch

- move-to-hue
  - Payload:
    - hue
    - direction
    - transition-time
    - options-mask
    - options-override
- move-hue
  - Payload:
    - move-mode
    - rate
    - options-mask
    - options-override

- step-hue  
Payload:
  - step-mode
  - step-size
  - transition-time
  - options-mask
  - options-override
- move-to-saturation  
Payload:
  - saturation
  - transition-time
  - options-mask
  - options-override
- move-saturation  
Payload:
  - move-mode
  - rate
  - options-mask
  - options-override
- step-saturation  
Payload:
  - step-mode
  - step-size
  - transition-time
  - options-mask
  - options-override
- move-to-hue-and-saturation  
Payload:
  - hue
  - saturation
  - transition-time
  - options-mask
  - options-override
- move-to-color  
Payload:
  - x
  - y

- transition-time
- options-mask
- options-override
- move-color
  - Payload:
    - rate-x
    - rate-y
    - options-mask
    - options-override
- step-color
  - Payload:
    - step-x
    - step-y
    - transition-time
    - options-mask
    - options-override
- move-to-color-temperature
  - Payload:
    - color-temperature-mireds
    - transition-time
    - options-mask
    - options-override
- enhanced-move-to-hue
  - Payload:
    - enhanced-hue
    - direction
    - transition-time
    - options-mask
    - options-override
- enhanced-move-hue
  - Payload:
    - move-mode
    - rate
    - options-mask
    - options-override
- enhanced-step-hue
  - Payload:

- step-mode
- step-size
- transition-time
- options-mask
- options-override
- enhanced-move-to-hue-and-saturation  
Payload:
  - enhanced-hue
  - saturation
  - transition-time
  - options-mask
  - options-override
- color-loop-set  
Payload:
  - update-flags
  - action
  - direction
  - time
  - start-hue
  - options-mask
  - options-override
- stop-move-step  
Payload:
  - options-mask
  - options-override
- move-color-temperature  
Payload:
  - move-mode
  - rate
  - color-temperature-minimum-mireds
  - color-temperature-maximum-mireds
  - options-mask
  - options-override
- step-color-temperature  
Payload:
  - step-mode
  - step-size

- transition-time
- color-temperature-minimum-mireds
- color-temperature-maximum-mireds
- options-mask
- options-override

#### 4.4. UbisysAutomationSwitch

- switch-latched

Payload:

- position

- initial-press

Payload:

- position

- long-press

Payload:

- position

- short-release

Payload:

- position

- long-release

Payload:

- position

#### 4.5. Scene

- recall-scene

Payload:

- group-id
- scene-id
- transition-time (if present)

## 5. OAuth 2.0 flow

Authorization for the cloud-based API is implemented via OAuth 2.0 , using the “Authorization Code Grant” with client credentials embedded into the request body.

Before using it, you need to contact ubisys to obtain a client identifier and associated client secret and you need to provide a redirect URI endpoint, which receives a temporary authorization code.

References:

- RFC 6749: The OAuth 2.0 Authorization Framework <https://tools.ietf.org/html/rfc6749>
- OAuth 2 Simplified by Aaron Parecki <https://aaronparecki.com/oauth-2-simplified/>

### 5.1. Flow description

#### 5.1.1. Flow initiation

The flow is initiated by directing your end user to the authorization endpoint in his or her web browser: <https://c2f.ubisys.de/oauth/authorize> with the following URI parameters:

Name	Value	Description
vendor	ubisys	Vendor-styled theme to apply to the login page.
client_id		Client identifier, as assigned by ubisys
response_type	code	Fixed code identifying the OAuth 2.0 flow
state		Generated by the client application. Unique for each request. This parameter allows linking the initial request to the redirect response. Contents are application-defined, but must use an URL-safe encoding.
scope	facility-discover facility-control facility-state	Authorization scopes to be granted
redirect_uri		Redirect URI to receive the temporary authorization code. To be implemented and provided by the integrator. Must be communicated to ubisys upon registration of the client identifier.

This will present a login form to the end user, where the serial number of the target system is entered and a previously set password for authentication.

On successful authentication (and implicit authorization), the browser will be redirected to the redirect URI. On failure, the flow ends in this step.

#### 5.1.2. Redirection

On successful authorization, the browser will be redirected to the specified redirect URI with the following URI parameters appended:

Name	Value	Description
state		Value of the state parameter, as set when initiating the flow
code		Temporary authorization code

The application must verify the state parameter and not proceed if the state is invalid. The exact means of generating and verifying the state parameter are application-defined.

#### 5.1.3. Obtain tokens

Having received the temporary authorization code via the redirection, the application then needs to send a POST request to obtain the access and refresh tokens:

<https://c2f.ubisys.de/oauth/token>

with the following POST parameters transmitted in the payload, encoded as application/x-www-form-urlencoded.

Name	Value	Description
grant_type	authorization_code	Fixed string indicating the submitted grant type
client_id		Client identifier, as assigned by ubisys
client_secret		Client secret, as assigned by ubisys
code		Temporary authorization code received in the previous step
redirect_uri		Must be identical to the redirect_uri provided in the initial link (5.1.1 Flow initiation)

The response is a JSON object containing the following keys:

Name	Value	Description
access_token		The access token to be used to perform API requests
refresh_token		The refresh token, to be used to obtain a new access token
token_type	Bearer	Fixed token type "Bearer"
expires_in		Validity of the access token in seconds

#### 5.1.4. Refresh access token

To obtain a new access token, a POST request to the token endpoint <https://c2f.ubisys.de/oauth/token> must be submitted with the following POST parameters in the payload:

Name	Value	Description
grant_type	refresh_token	Fixed string indicating the submitted grant type
client_id		Client identifier, as assigned by ubisys
client_secret		Client secret, as assigned by ubisys
refresh_token		The refresh token initially obtained

The response is a JSON object containing the following keys:

Name	Value	Description
access_token		The new access token to be used to perform API requests
refresh_token		The refresh token (unchanged)
token_type	Bearer	Fixed token type "Bearer"
expires_in		Validity of the access token in seconds

#### 5.1.5. Bearer authentication

The access token obtained either via the initial request or via a token refresh must be used to authenticate each request.

It must be set via the HTTP Authentication header and the bearer scheme (c.f. RFC 6750), as per the following example:

```
Authorization: Bearer
Z3c9MTA.MTVmZjdiM2RjMGM5ZjU0M2I5YzJhZWYwYy.wOTtzPTZDQ0w3WVRJSHNn
```

#### 5.1.6. Flow example

The following example will use the application-generated state **8HarsG2DRpuci8n302GyQQ** and the authorization code **12345678abcdef**.

- Initial request The client application directs the user to the authorization endpoint, e.g.

[https://c2f.ubisys.de/oauth/authorize?vendor=ubisys&client\\_id=example\\_client&response\\_type=code&state=8HarsG2DRpuci8n302GyQQ&scope=facility-discover+facility-control+facility-state&redirect\\_uri=https%3A%2F%2Fauth.example.com%2Flink%2F1234%3F](https://c2f.ubisys.de/oauth/authorize?vendor=ubisys&client_id=example_client&response_type=code&state=8HarsG2DRpuci8n302GyQQ&scope=facility-discover+facility-control+facility-state&redirect_uri=https%3A%2F%2Fauth.example.com%2Flink%2F1234%3F)

- User authorizes the request The user will then enter his or her access credentials, which will be validated by the gateway.

On success, a redirect will be issued to:

<https://auth.example.com/link/1234/?state=8HarsG2DRpuci8n302GyQQ&code=12345678abcdef>

- The client application verifies the state

If verification passes, the received temporary authorization code will be used to request the access and refresh token.

- Request tokens

A POST request will be issued to:

<https://c2f.ubisys.de/oauth/token>

with the following payload:

```
grant_type=authorization_code&
client_id=example_client&
client_secret=example_secret&
code=12345678abcdef&
redirect_uri=https%3A%2F%2Fauth.example.com%2Flink%2F1234%3F
```

The response will be JSON:

```
{
  "access_token": "Z3c9MTA.YjiMjUwZjcxMTk.cz1LLUM4UUdjdWNE",
  "refresh_token": "Z3c9MTA.YjNjNTUyZThiN2.cz1FeUt1cXV4OVd1",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

The access token can now be used to authorize requests for a duration of one hour (3600 seconds).

- Token refresh

Once the access token expired, it has to be refreshed by using the previously obtained refresh token.

A POST request will be issued to:

<https://c2f.ubisys.de/oauth/token>

with the following payload:



```
grant_type=authorization_code&
client_id=example_client&
client_secret=example_secret&
refresh_token=Z3c9MTA.YjNjNTUyZThiN2.RXlLdXF1eDlXdQ==
```

The response will be JSON:

```
{
  "access_token": "Z3c9MTA.n6KIn4z33oyppjQ.cz15ZmRmMDNhYzBE",
  "refresh_token": "Z3c9MTA.YjNjNTUyZThiN2.cz1FeUt1cXV4OVd1",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

The new access token can then be used to authorize further requests for another hour.

## 6. Outlook

- Exposure of rooms and mapped devices

## 7. Revision History

<b>Revision</b>	<b>Date</b>	<b>Remarks</b>
1.0	27/05/2019	Initial public version
1.1	26/01/2023	Added window covering documentation
1.2	21/04/2023	Added color control documentation
1.3	26/04/2023	Added documentation of the WebSocket protocol
1.4	17/05/2023	Added documentation on sensors and input events
1.5	31/08/2023	Added documentation on scenes
1.6	12/02/2024	Added documentation on groupcasts, IAS zones and inbound scene-recall commands
1.7	04/04/2024	Added documentation on group members and window covering commands; added examples

## 8. Contact

ubisys technologies GmbH  
Neumannstr. 10  
40235 Düsseldorf  
Germany

T: +49. 211. 54 21 55 - 19

F: +49. 211. 54 21 55 - 99

[www.ubisys.de](http://www.ubisys.de)

[info@ubisys.de](mailto:info@ubisys.de)