RDC Charger

User manual

Robotina d.o.o.
OIC-Hrpelje 38 Hrpelje
SI-6240 Kozina
Slovenia

Table of Contents

S-RMS API

The API is a REST API with a bearer token authorization. The API specification can be found here.

Adding an access code

An access code can be added with a POST to /api/v1/room/{access_code_controller_id}/code

The response is a HTTP 202 Accepted.

The access_code_controller_id path parameter is the ID of the controller to add the code to (eg.: access_code_controller_01, access_code_controller_02).

The POST data is a JSON containing the necessary data for a valid access code.

```
{
  "code": 123, // The access code in decimal representation
  "label": "Guest 1", // The code label or description (can be empty)
  "code_type": 1, // The code type
  "valid_from": "2022-10-24T07:33:21.674Z", // Start time of code validity
in the UTC timezone (can be null for no start limit)
  "valid_to": "2022-10-30T07:33:21.674Z", // End time of code validity in
the UTC timezone (can be null for no end limit)
  "access": [ // List of access points to grant the access to
    {
      "access_point": 1, // Access point ID
      "remaining_passes": 3 // Number of passes for the access point (can be
null for unlimited passes)
    },
    {
      "access_point": 2, // Access point ID
      "remaining_passes": null // Number of passes for the access point (can
be null for unlimited passes)
    }
  ]
}
```

A valid code must contain at least one access point with the remaining number of passes greater than 0 and must not be expired.

Removing an access code

An access code can be removed with a DELETE to /api/v1/room/{access_code_controller_id}/code/{code}

The response is a HTTP 204.

The access_code_controller_id path parameter is the ID of the controller to remove the code from (eg.: access_code_controller_01, access_code_controller_02). The code path parameter is the decimal representation of the code to remove.

Updating an access code

An access code can be updated with a PUT to /api/v1/room/{access_code_controller_id}/code/{code}

The response is a HTTP 202 Accepted.

The access_code_controller_id path parameter is the ID of the controller to update the code on (eg.: access_code_controller_01, access_code_controller_02). The code path parameter is the decimal representation of the code to update.

The PUT data is a JSON containing the necessary data for a valid access code.

```
{
  "code": 123, // The access code in decimal representation
  "label": "Guest 1", // The code label or description (can be empty)
  "code_type": 1, // The code type
  "valid_from": "2022-10-24T07:33:21.674Z", // Start time of code validity
in the UTC timezone (can be null for no start limit)
  "valid_to": "2022-10-30T07:33:21.674Z", // End time of code validity in
the UTC timezone (can be null for no end limit)
  "access": [ // List of access points to grant the access to
    {
      "access_point": 1, // Access point ID
      "remaining_passes": 3 // Number of passes for the access point (can be
null for unlimited passes)
    },
    {
      "access_point": 2, // Access point ID
      "remaining_passes": null // Number of passes for the access point (can
be null for unlimited passes)
    }
  ]
}
```

A valid code must contain at least one access point with the remaining number of passes greater than 0 and must not be expired.

If no access points are specified then the access points of the access code are **not modified**.

Updating only one access point of a code

Removing an access point

An access point of an access code can be removed with a DELETE to /api/v1/room/{access_code_controller_id}/code/{code}/access_point/{ap_id}

The response is a HTTP 204.

The access_code_controller_id path parameter is the ID of the controller to update the code on (eg.: access_code_controller_01, access_code_controller_02). The code path parameter is the decimal representation of the code to update. The ap_id path parameter is the ID of the access point to remove.

Adding or updating an access point

An access point of an access code can be added/updated with a PUT to /api/v1/room/{access_code_controller_id}/code/{code}/access_point/{ap_id}

The response is a HTTP 202.

The access_code_controller_id path parameter is the ID of the controller to update the code on (eg.: access_code_controller_01, access_code_controller_02). The code path parameter is the decimal representation of the code to update. The ap_id path parameter is the ID of the access point to remove.

The PUT data is a JSON containing the necessary data for a valid access point.

```
{
  "access_point": 1, // Access point ID
  "remaining_passes": 3 // Number of passes for the access point (can be
null for unlimited passes)
}
```

Reading an access code

An access code can be read with a GET to /api/v1/room/{access_code_controller_id}/code/{code}

The access_code_controller_id path parameter is the ID of the controller to update the code on (eg.: access_code_controller_01, access_code_controller_02). The code path parameter is the decimal representation of the code to update.

The response is a JSON containing code data.

```
{
  "code": 123, // The access code in decimal representation
  "label": "Guest 1", // The code label or description (can be empty)
  "code_type": 1, // The code type
  "valid_from": "2022-10-24T07:33:21.674Z", // Start time of code validity
in the UTC timezone (can be null for no start limit)
  "valid_to": "2022-10-30T07:33:21.674Z", // End time of code validity in
the UTC timezone (can be null for no end limit)
  "access": [ // List of access points to grant the access to
```

```
    {
      "access_point": 1, // Access point ID
      "remaining_passes": 3 // Number of passes for the access point (can be
null for unlimited passes)
    },
    {
      "access_point": 2, // Access point ID
      "remaining_passes": null // Number of passes for the access point (can
be null for unlimited passes)
    }
  ]
}
```

Reading code data of a specified access point

Access point data of an access code can be read with a GET to /api/v1/room/{access_code_controller_id}/code/{code}/access_point/{ap_id}

The access_code_controller_id path parameter is the ID of the controller to read the code from (eg.: access_code_controller_01, access_code_controller_02). The code path parameter is the decimal representation of the code to read. The ap_id path parameter is the ID of the access point to read.

The response is a JSON with the access point data.

```
{
  "access_point": 1, // Access point ID
  "remaining_passes": 3 // Number of passes for the access point (can be
null for unlimited passes)
}
```

Reading/Setting room devices data

Each room is split into logical *things* that represent controllable devices in the room.

The different types of things are:

- light
- dimmable light
- scene
- blinds
- DND/MUR
- SOS
- fire alert
- door
- HVAC
- HVAC setpoints
- bathroom ventilation
- room presence

- room power supply
- room
- wake up

The parent of all these things is the room controller.

Reading room device data

Geting the room device state is done with a GET to /api/v1/things/rt/{thing_id}

The thing_id path parameter is the ID of the device to read the state from (eg.: room_01_light_1, room_02_hvac).

The response is a JSON with the device state.

Light state

```
{
  "id": "room_01_light_1",
  "type": "com.robotina.s_rms.iot_things.light-1_0",
  "vars": {
    "light": "0"
  }
}
```

Dimmable light state

```
{
  "id": "room_01_dimmable_light_1",
  "type": "com.robotina.s_rms.iot_things.dimmable_light-1_0",
  "vars": {
    "dimmer": "0",
    "dimmer_level": "0"
  }
}
```

Scene state

```
{
  "id": "room_01_scene_1",
  "type": "com.robotina.s_rms.iot_things.scene-1_0",
  "vars": {
    "scene_status": "1"
  }
}
```

Blinds state

```
{
  "id": "room_01_scene_1",
  "type": "com.robotina.s_rms.iot_things.scene-1_0",
  "vars": {
    "scene_status": "1"
  }
}
```

DND/MUR state

```
{
  "id": "room_01_dnd_mur",
  "type": "com.robotina.s_rms.iot_things.dnd_mur-1_0",
  "vars": {
    "mur_status": "1",
    "dnd_status": "0"
  }
}
```

SOS state

```
{
  "id": "room_01_sos",
  "type": "com.robotina.s_rms.iot_things.sos_alert-1_0",
  "vars": {
    "sos_status": "0"
  }
}
```

Fire alert state

```
{
  "id": "room_01_fire_alert",
  "type": "com.robotina.s_rms.iot_things.fire_alert-1_0",
  "vars": {
    "fire_alert_status": "0"
  }
}
```

Door state

```
{
  "id": "room_01_door",
```

```
  "type": "com.robotina.s_rms.iot_things.door-1_0",
  "vars": {
    "door_status": "0",
    "door_open_code_type": "0",
    "door_bell": "0",
    "door_bell_status": "0",
    "door_bell_preset": "1000",
    "access_lock": "0"
  }
}
```

HVAC state

```
{
  "id": "room_01_hvac",
  "type": "com.robotina.s_rms.iot_things.hvac-1_0",
  "vars": {
    "hvac_room_mode": "2",
    "hvac_room_status": "6",
    "hvac_room_measured": "23.1",
    "hvac_room_fan": "0",
    "hvac_room_setpoint": "23.8",
    "hvac_room_fan_mode": "2"
  }
}
```

HVAC setpoints state

```
{
  "id": "room_01_hvac_setpoints",
  "type": "com.robotina.s_rms.iot_things.hvac_setpoints-1_0",
  "vars": {
    "hvac_room_setpoint_cool_idle": "30.0",
    "hvac_room_setpoint_heat_idle": "10.0",
    "hvac_room_fan_idle": "1",
    "hvac_room_setpoint_cool_book": "25.0",
    "hvac_room_setpoint_heat_book": "18.0",
    "hvac_room_fan_book": "2",
    "hvac_room_setpoint_guest": "23.8",
    "hvac_room_fan_guest": "2",
    "hvac_room_setpoint_cool_max": "30.0",
    "hvac_room_setpoint_cool_min": "18.0",
    "hvac_room_setpoint_heat_max": "25.0",
    "hvac_room_setpoint_heat_min": "10.0",
    "hvac_room_offset_cool_night": "2.0",
    "hvac_room_offset_heat_night": "-2.0",
    "hvac_room_fan_night": "1",
    "def_hvac_room_setpoint_cool_idle": "30.0",
```

```
    "def_hvac_room_setpoint_heat_idle": "10.0",
    "def_hvac_room_fan_idle": "1",
    "def_hvac_room_setpoint_cool_book": "25.0",
    "def_hvac_room_setpoint_heat_book": "18.0",
    "def_hvac_room_fan_book": "2",
    "def_hvac_room_setpoint_guest": "22.4",
    "def_hvac_room_fan_guest": "1",
    "def_hvac_room_setpoint_cool_max": "30.0",
    "def_hvac_room_setpoint_cool_min": "18.0",
    "def_hvac_room_setpoint_heat_max": "25.0",
    "def_hvac_room_setpoint_heat_min": "10.0",
    "def_hvac_room_offset_cool_night": "2.0",
    "def_hvac_room_offset_heat_night": "-2.0",
    "def_hvac_room_fan_night": "1"
  }
}
```

Bathroom ventilation state

```
{
  "id": "room_01_ventilation",
  "type": "com.robotina.s_rms.iot_things.ventilation-1_0",
  "vars": {
    "ventilation_status": "0",
    "ventilation_timeout": "30"
  }
}
```

Room presence state

```
{
  "id": "room_01_presence",
  "type": "com.robotina.s_rms.iot_things.room_presence-1_0",
  "vars": {
    "presence_status": "1",
    "presence_state": "1",
    "presence_type": "0",
    "presence_pir_sensor": "0",
    "presence_door_sensor": "0",
    "presence_aux_sensor": "0",
    "presence_long_preset": "900",
    "presence_short_preset": "5"
  }
}
```

Room power supply state

```
{
  "id": "room_01_power_supply",
  "type": "com.robotina.s_rms.iot_things.room_power_supply-1_0",
  "vars": {
    "power_booked": "1",
    "power_presence": "1"
  }
}
```

Room state

```
{
  "id": "room_01",
  "type": "com.robotina.s_rms.iot_things.room-1_0",
  "vars": {
    "room_number": "102",
    "room_status": "0",
    "booked_state": "1",
    "booked_on_code": "1"
  }
}
```

Wake up state

```
{
  "id": "room_01_wake_up",
  "type": "com.robotina.s_rms.iot_things.wake_up-1_0",
  "vars": {
    "wakeup_status": "0",
    "wakeup_enabled": "0",
    "wakeup_time_hour": "8",
    "wakeup_time_min": "0",
    "wakeup_scene_enabled": "0"
  }
}
```

Setting room device state

For writable device variables the state can be changed with a POST to /api/v1/things/rt/{thing_id}

The thing_id path parameter is the ID of the device to read the state to (eg.: room_01_light_1, room_02_hvac).

The POST data is a JSON containing **only** the variables to change.

```
{
  "vars": {
```

```
    "variable_to_change_1": "new_value_1",
    "variable_to_change_2": "new_value_2"
  }
}
```

The variable structure is the same as the response for reading device states.

The response is the updated device state (same as in a read request).