Implementation manual Octalarm Link

REST API for Network Controlled Alarms

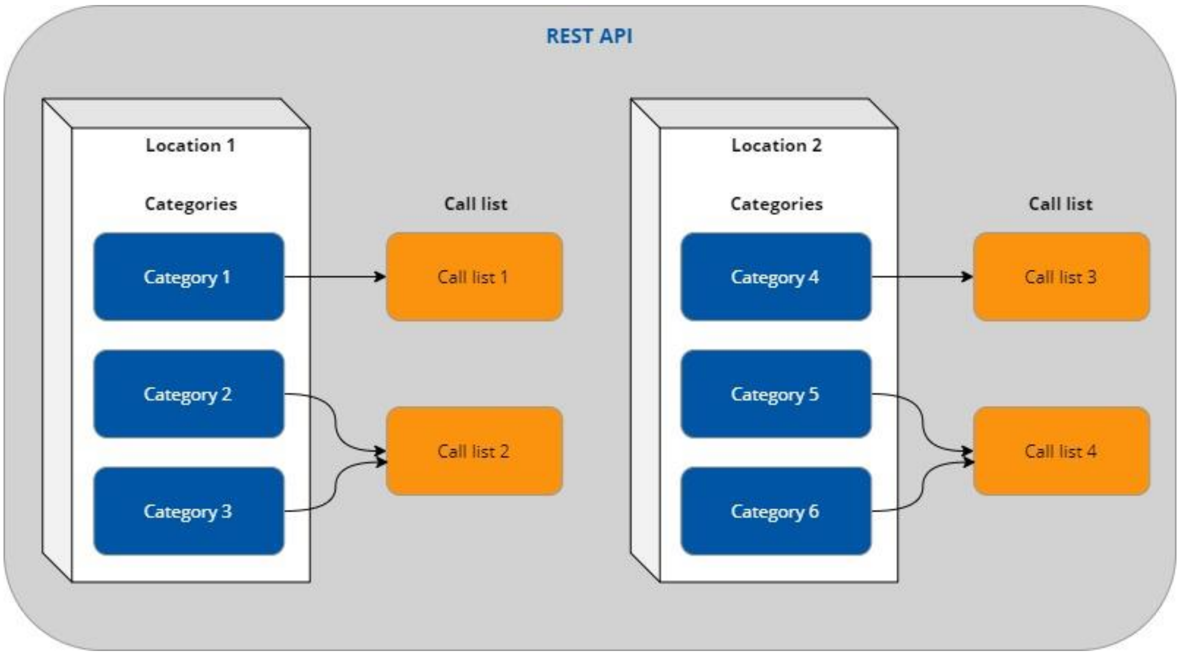
Version 1.3

Index

1 Introduction

The REST API can be used by external devices to communicate with the Octalarm-Touch Pro. These external devices can use the REST API to start alarm reports with a specific alarm text, and to monitor the connection with the Octalarm-Touch Pro.

Instead of creating all these external alarms separately in the Octalarm-Touch Pro, the idea is to use a number of categories that were created in the Octalarm-Touch Pro. These categories (with their specific alarm handling) can be used by the external device to start alarm reports with a specific alarm text.



For every device that wants to communicate with the Octalarm-Touch Pro, a REST API must be created. The communication with the REST API can be monitored by using a watchdog mechanism. When the communication fails, a watchdog alarm will become active.

Because the alarm handling can differ per location or group, it's necessary to create one or more locations on the REST API.

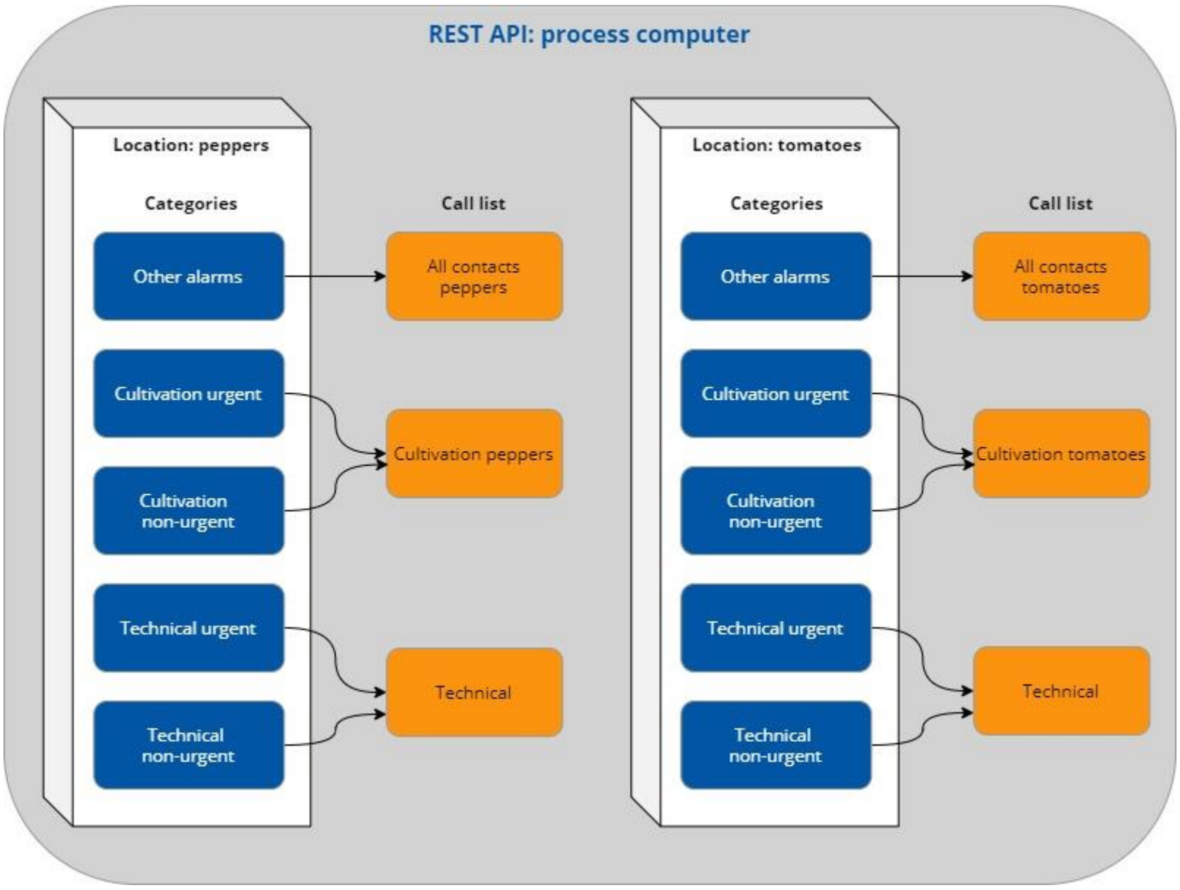
And finally the categories per location must be created. So then these categories can be used to distribute the alarms across the different categories.

2 Example

A Process Computer is used in a greenhouse for controlling the climate, irrigation and energy. This climate computer communicates via the REST API with the Octalarm-Touch Pro.

Because tomatoes and peppers are grown in this greenhouse, different locations have to be created. In this way the alarms can be handled differently per location.

And finally, per location, the alarms are divided between cultivation alarms (urgent or non-urgent) and technical alarms (urgent or non-urgent).



This document will explain, step by step, how to implement the REST API, the locations and the categories. And finally it will explain how to start alarm reports via the REST API.

3 HTTP methods

Just like every REST API, you use

- HTTP GET requests to retrieve data;
- POST and PUT requests to add or overwrite data;
- and HTTP DELETE to remove things.

All requests and responses use JSON objects and arrays as data format.

Method	Description
GET	Used for retrieving all elements of the entity, or one specific element by ID.
POST	Used for creating a new entity.
PUT	Used for updating an element with new data.
DELETE	Used for deleting an entity.

The REST API is using OpenApi (Swagger) for API documentation. The API documentation is available by entering the IP-address of the Octalarm-Touch Pro, followed by "/rest_api/1/", in a web browser. For example: http://192.168.10.60/rest_api/1/

The "/1/" in the URL is the version number of the REST API. As long as the REST API is compatible with older software versions, this version number will remain unchanged.

You will find an overview of all the API's, including documentation. It's also possible to execute or test the API's. At every API you will also find the cURL command. cURL is a command-line tool for getting or sending data and can be used to communicate with the REST API.

4 Authentication

The REST API uses a web token (type Bearer) to authenticate the user, and is used by the external device to make API calls. The token will be sent in the authorization header.

See https://jwt.io/ for more information about web tokens.

5 Responses

The following responses should be implemented as a minimum:

200 (OK)

This response indicates that the request has succeeded.

400 (Bad request)

General error status, used when there is no specific error code that fits better. Also used when a limit is reached, for example a maximum number of active alarms.

401 (Unauthorized)

The request has not been applied because it lacks valid authentication credentials for the target resource.

403 (Forbidden)

This response indicates that the client's request is formed correctly, but the REST API refuses to authorize it.

404 (Not Found)

This response indicates that the requested resource was not found.

405 (Method Not Allowed)

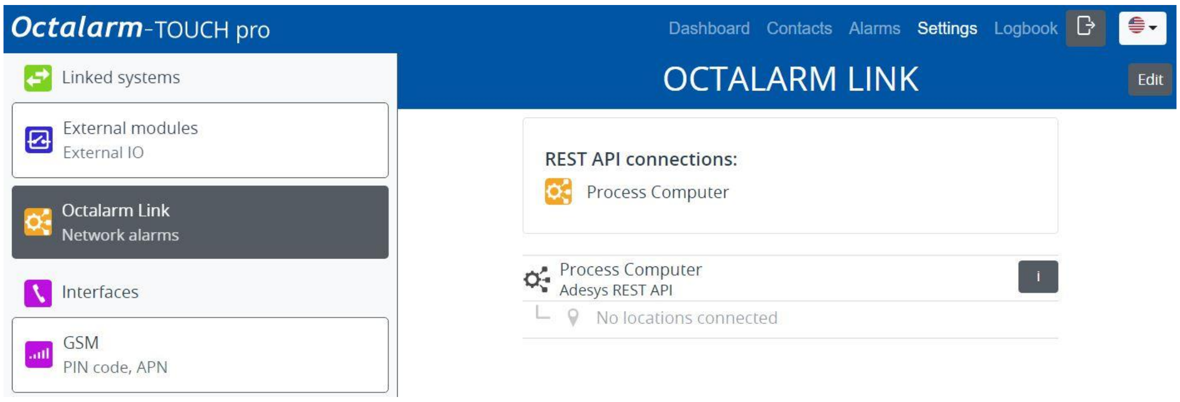
This response indicates that the client tried to use an HTTP method that the resource does not allow.

500 (Internal Server Error)

There was an error on the server and the request could not be completed. This is the generic REST API error response.

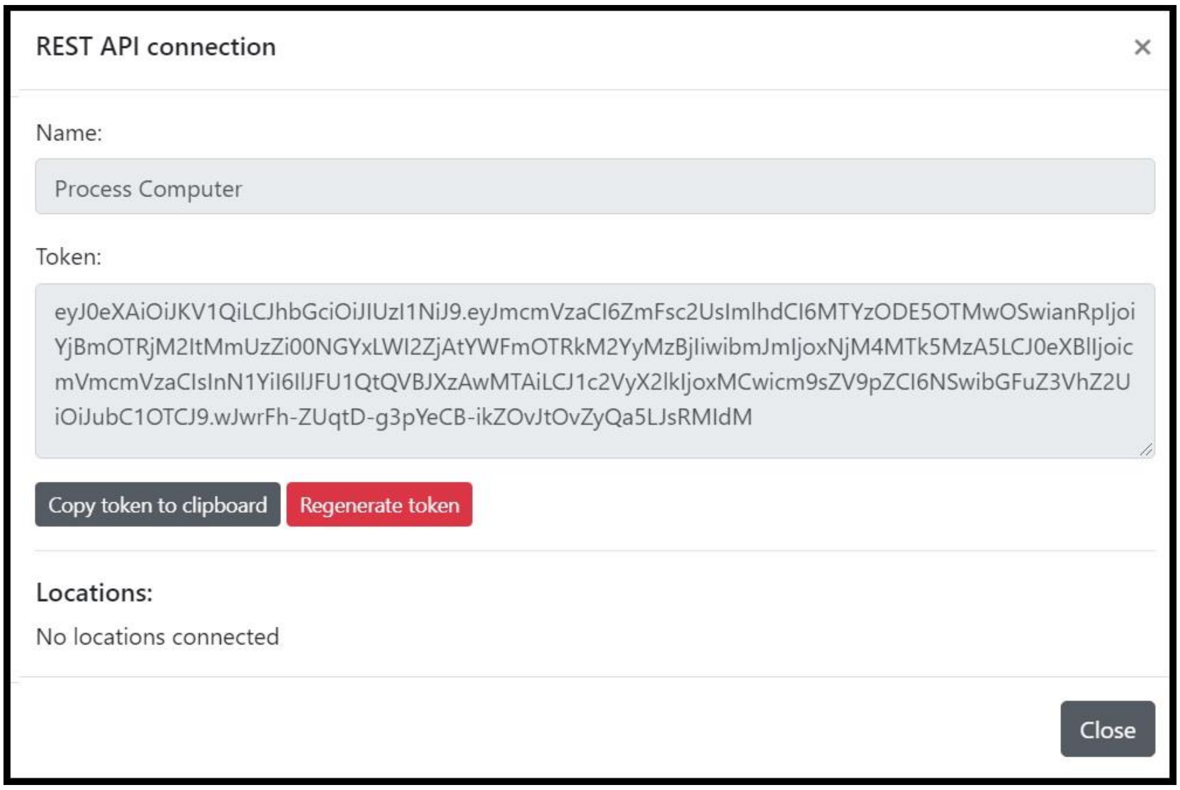
6 REST API connection

External devices can use the REST API to communicate with the Octalarm-Touch Pro. The first step is to create an REST API connection for every external device that must be connected. The best way to create an REST API, is to use the web interface:



The Octalarm-Touch Pro supports multiple REST API's, each with a uniquely generated token.

When the REST API is created, the token can be found by clicking on the info button. Now it's very easy to copy the token to the clipboard.



REST API connection

Name:

Process Computer

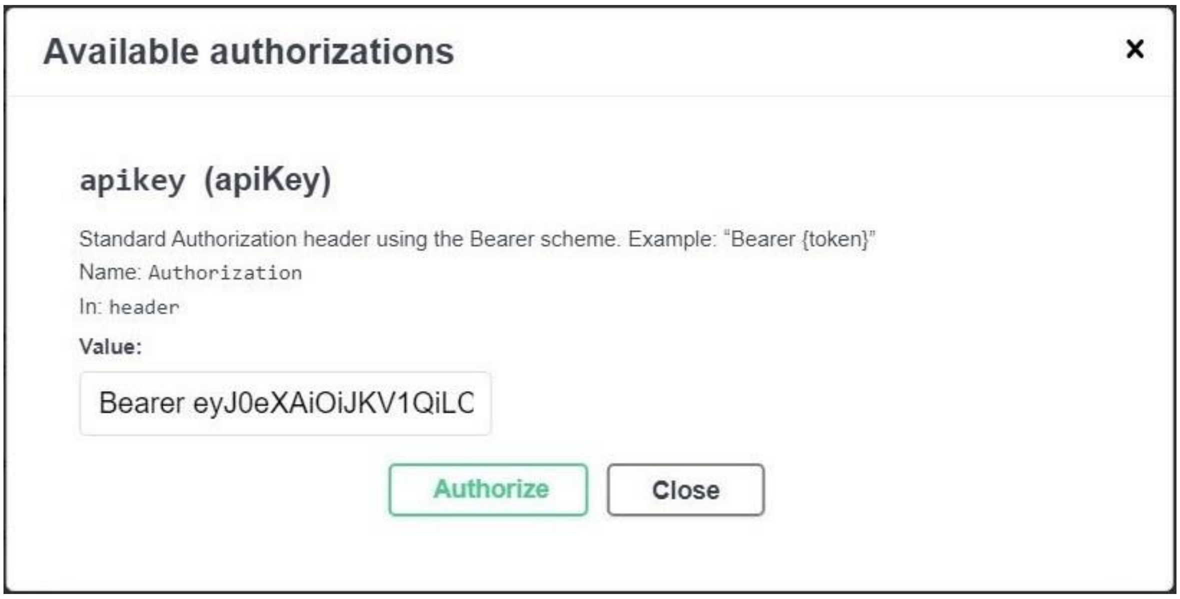
Token:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTYzODE5OTMwOSwianRpIjoi
YjBmOTRjM2ItMmUzZi00NGYxLWI2ZjAtYWFmOTRkM2YyMzBjIiwibmJmIjoxNjM4MTk5MzA5LCJ0eXBlIjoic
mVmcmVzaCIsInN1YiI6IlJFU1QtQVBJXzAwMTAiLCJ1c2VyX2lkIjoxMCwicm9sZV9pZCI6NSwibGFuZ3VhZ2U
iOiJubC1OTCJ9.wJwrFh-ZUqtD-g3pYeCB-ikZOvJtOvZyQa5LJsRMIdM
```

Locations:

No locations connected

This token can also be used for testing with the OpenApi (Swagger) tool. In the authorization screen the string "Bearer" must be entered, followed by a space and the token.



Available authorizations

apikey (apiKey)

Standard Authorization header using the Bearer scheme. Example: "Bearer {token}"

Name: Authorization

In: header

Value:

Bearer eyJ0eXAiOiJKV1QiLC

The connection between the external device and the REST API can be monitored with a watchdog mechanism.

"POST: /Interface/Watchdog"

```
{
  "next_kick_before_secs": 60
}
```

Curl code:

```
curl -X 'POST' \
  'http://192.168.10.72/rest_api/1/Interface/Watchdog' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE2MTI1MzQ0ODgsIm5iZiI6MTYxMjUzNDQ4OCwianRpIjoiNjY0
NzI2NzAtYTRkMi00N2ZmLWFiMjUtNzAyMGYwOTkwNWJhIiwiZXhwIjoxNjEyNTM3OTg4LCJpZGVudGl0eSI6IlJFU1QtQVBJX
zAwMDUiLCJmcmVzaCI6ZmFsc2UsInR5cGUiOiJhY2Nlc3MiLCJ1c2VyX2NsYWltcyI6eyJ1c2VyX2lkIjo1LCJyb2xlX2lkIj
o1LCJsYW5ndWFnZSI6Im5sLU5MIn19.6xlCrMjNsKd9Eyq5ieS-MScZe6P1idstMCKPs5dfOd8' \
  -H 'Content-Type: application/json' \
  -d '{
  "next_kick_before_secs": 60
}'
```

Response:

```
{
  "next_kick_before_secs": 60
}
```

If no new keep-alive message (watchdog API call) is received within the specified time, a system alarm will be generated.

7 Locations

Because alarm handling can be different for every location, these locations must be added to the REST API so that they can report independently.

"POST: /Locations"

```
{
  "custom_id": "loc_01",
  "enabled": true,
  "name": "Tomatoes"
}
```

Curl code:

```
curl -X 'POST' \
  'http://192.168.10.72/rest_api/1/Locations' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE2MTI1MzQ0ODgsIm5iZiI6MTYxMjUzNDQ4OCwianRpIjoiNjY0
NzI2NzAtYTRkMi00N2ZmLWFiMjUtNzAyMGYwOTkwNWJhIiwiZXhwIjoxNjEyNTM3OTg4LCJpZGVudGl0eSI6IlJFU1QtQVBJX
zAwMDUiLCJmcmVzaCI6ZmFsc2UsInR5cGUiOiJhY2Nlc3MiLCJ1c2VyX2NsYWltcyI6eyJ1c2VyX2lkIjo1LCJyb2xlX2lkIj
o1LCJsYW5ndWFnZSI6Im5sLU5MIn19.6xlCrMjNsKd9Eyq5ieS-MScZe6P1idstMCKPs5dfOd8' \
  -H 'Content-Type: application/json' \
  -d '{
  "custom_id": "loc_01",
  "enabled": true,
  "name": "Tomatoes"
}'
```

Response:

```
{
"id": 3,
"name": "Tomatoes",
"custom_id": "loc_01",
"enabled": true
}
```

When all locations are added, we can retrieve all the locations with this API:

"GET: /Locations"

In our example we created two locations: Tomatoes and Peppers.

Curl code:

```
curl -X 'GET' \
  'http://192.168.10.72/rest_api/1/Locations' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE2MTI1MzQ0ODgsIm5iZiI6MTYxMjUzNDQ4OCwianRpIjoiNjY0
NzI2NzAtYTRkMi00N2ZmLWFiMjUtNzAyMGYwOTkwNWJhIiwiZXhwIjoxNjEyNTM3OTg4LCJpZGVudGl0eSI6IlJFU1QtQVBJX
zAwMDUiLCJmcmVzaCI6ZmFsc2UsInR5cGUiOiJhY2Nlc3MiLCJ1c2VyX2NsYWltcyI6eyJ1c2VyX2lkIjo1LCJyb2xlX2lkIj
o1LCJsYW5ndWFnZSI6Im5sLU5MIn19.6xlCrMjNsKd9Eyq5ieS-MScZe6P1idstMCKPs5dfOd8'
```

Response:

```
{
  "locations": [
    {
      "id": 3,
      "name": "Tomatoes",
      "custom_id": "loc_01",
      "enabled": true
    },
    {
      "id": 4,
      "name": "Peppers",
      "custom_id": "loc_02",
      "enabled": true
    }
  ]
}
```

The table below shows the elements with their description:

Element	Description
custom_id	unique location ID defined by customer
enabled	false when system is disabled, for example when maintenance has to be done.
id	unique location database ID, generated by Octalarm Touch.
name	name will be used when alarms are generated. The name of this location will be used in the alarm report (voice or text) and will also appear in the logbook and on the display.

All elements can be changed afterwards, except the "id" element. Changing elements can be done with the PUT method, using the "custom_id".

Example:

“PUT: /Locations/CustomId/loc_01”

```
{
  "name": "Cucumber"
}
```

Curl code:

```
curl -X 'PUT' \
  'http://192.168.10.72/rest_api/1/Locations/CustomId/loc_01' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE2MTI1MzQ0ODgsIm5iZiI6MTYxMjUzNDQ4OCwianRpIjoiNjY0
NzI2NzAtYTRkMi00N2ZmLWFiMjUtNzAyMGYwOTkwNWJhIiwiZXhwIjoxNjEyNTM3OTg4LCJpZGVudGl0eSI6IlJFU1QtQVBJX
zAwMDUiLCJmcmVzaCI6ZmFsc2UsInR5cGUiOiJhY2Nlc3MiLCJ1c2VyX2NsYWltcyI6eyJ1c2VyX2lkIjo1LCJyb2xlX2lkIj
o1LCJsYW5ndWFnZSI6Im5sLU5MIn19.6xlCrMjNsKd9Eyq5ieS-MScZe6P1idstMCKPs5dfOd8' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Cucumber"
}'
```

Response:

```
{
  "id": 3,
  "name": "Cucumber",
  "custom_id": "loc_01",
  "enabled": true
}
```

8 Categories

Alarms often have to be distributed over different call lists. Therefore we have to create multiple categories. In the example used before, there were 4 categories:

1. cultivation urgent
2. cultivation non-urgent
3. technical urgent
4. technical non-urgent

This API can be used to add the categories:

"POST: /Categories"

```
{
  "auto_repeat_delay": 300,
  "auto_repeat_times": 3,
  "buzzer": true,
  "call_list_id": 1,
  "continue_after_restore": false,
  "critical": true,
  "enabled": true,
  "location_custom_id": "loc_01",
  "name": "Cult. Urgent",
  "report_during_daytime": true,
  "report_during_nighttime": true,
  "report_during_weekend": true,
  "topic": "cultivation_urg"
}
```

Curl code:

```
curl -X 'POST' \
  'http://192.168.10.72/rest_api/1/Categories' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE2MTI1MzQ0ODgsIm5iZiI6MTYxMjUzNDQ4OCwianRpIjoiNjY0
NzI2NzAtYTRkMi00N2ZmLWFiMjUtNzAyMGYwOTkwNWJhIiwiZXhwIjoxNjEyNTM3OTg4LCJpZGVudGl0eSI6IlJFU1QtQVBJX
zAwMDUiLCJmcmVzaCI6ZmFsc2UsInR5cGUiOiJhY2Nlc3MiLCJ1c2VyX2NsYWltcyI6eyJ1c2VyX2lkIjo1LCJyb2xlX2lkIj
o1LCJsYW5ndWFnZSI6Im5sLU5MIn19.6xlCrMjNsKd9Eyq5ieS-MScZe6P1idstMCKPs5dfOd8' \
  -H 'Content-Type: application/json' \
  -d '{
  "auto_repeat_delay": 300,
  "auto_repeat_times": 3,
  "buzzer": true,
  "call_list_id": 1,
  "continue_after_restore": false,
  "critical": true,
  "enabled": true,
  "location_custom_id": "loc_01",
  "name": "Cult. Urgent",
  "report_during_daytime": true,
  "report_during_nighttime": true,
  "report_during_weekend": true,
  "topic": "cultivation_urg"
}'
```

Response:

```
{
  "id": 35,
  "location_id": 3,
  "topic": "cultivation_urg",
  "enabled": true,
  "name": "Cult. Urgent",
  "report_during_daytime": true,
  "report_during_nighttime": true,
  "report_during_weekend": true,
  "continue_after_restore": false,
  "call_list_id": 1,
  "auto_repeat_times": 3,
  "auto_repeat_delay": 300,
  "buzzer": true,
  "critical": true,
  "location_custom_id": "loc_01"
}
```

The table below shows the elements with their description:

Element	Description
auto_repeat_delay	delay before an alarm is repeated
auto_repeat_times	amount of repeats after the alarm is first accepted
buzzer	enable or disable the buzzer for this alarm
call_list_id	ID of the selected call list
category_id	unique generated ID for category
continue_after_restore	if enabled the call list must continue after the alarm is restored
critical	if enabled the app volume will be on maximum volume
disabled_until	disable alarm until UTC timestamp YYYY-MM-DDTHH:MM:SSZ
enabled	enable or disable the alarm reporting for this category
location_cat_seq	category sequence number based on location
location_custom_id	unique location ID defined by customer, can be used instead of location_id
location_id	unique generated ID for the location
name	name for the category
report_during_daytime	if enabled this alarm must be reported during daytime
report_during_nighttime	if enabled this alarm must be reported during nighttime
report_during_weekend	if enabled this alarm must be reported during the weekend
topic	unique custom ID for the category

When a category is created, the ID of a call list can also be specified (call_list_id). It is recommended to use call_list_id 1, which is the standard call list in the Octalarm-Touch Pro. If another call list is preferred, it can be selected in the menus of the Octalarm-Touch Pro itself. This can be done on the touch screen or in the web browser.

9 Alarm report

Alarms are sent with all the information needed to start a report. This API starts an alarm based on a category topic:

"POST: / Categories / Alarm"

```
{
  "alarm_value": true,
  "category_topic": "cultivation_urg",
  "location_custom_id": "loc_01",
  "message": "dept. 2 temperature too high",
  "name": "dept. 2 temperature",
  "unique_id": "loc_01_alrm_01a1b4"
}
```

Curl code:

```
curl -X 'POST' \
  'http://192.168.10.72/rest_api/1/Categories/Alarm' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE2MTI1MzQ0ODgsIm5iZiI6MTYxMjUzNDQ4OCwianRpIjoiNjY0
NzI2NzAtYTRkMi00N2ZmLWFiMjUtNzAyMGYwOTkwNWJhIiwiZXhwIjoxNjEyNTM3OTg4LCJpZGVudGl0eSI6IlJFU1QtQVBJX
zAwMDUiLCJmcmVzaCI6ZmFsc2UsInR5cGUiOiJhY2Nlc3MiLCJ1c2VyX2NsYWltcyI6eyJ1c2VyX2lkIjo1LCJyb2xlX2lkIj
o1LCJsYW5ndWFnZSI6Im5sLU5MIn19.6xlCrMjNsKd9Eyq5ieS-MScZe6P1idstMCKPs5dfOd8' \
  -H 'Content-Type: application/json' \
  -d '{
  "alarm_value": true,
  "category_topic": "cultivation_urg",
  "location_custom_id": "loc_01",
  "message": "dept. 2 temperature too high",
  "name": "dept. 2 temperature",
  "unique_id": "loc_01_alrm_01a1b4"
}'
```

Response:

```
{
  "unique_id": "loc_01_alrm_01a1b4",
  "name": "dept. 2 temperature",
  "message": "dept. 2 temperature too high",
  "category_topic": "cultivation_urg",
  "alarm_value": true,
  "location_id": 3,
  "location_custom_id": "loc_01",
  "category_id": 35
}
```

The table below shows the elements with their description:

Element	Description
alarm_value	true for alarm, false for recovered
category_topic	unique category ID defined by customer
location_custom_id	unique location ID defined by customer
location_id	unique location database ID, generated by Octalarm Touch
message	text that will be used for alarm reporting (voice, SMS, app, etc.)
name	short name to display the alarm on display or logbook
unique_id	alarm custom ID

When an unknown location ID is used, the Octalarm-Touch Pro will use the alarm handling of the “unknown location” system alarm. When an unknown category topic is used, the Octalarm-Touch Pro will use the default category of that specific location.

All alarms of the external device must have their own unique ID. This unique ID (unique_id) is used by the external device when an alarm report is started via the REST API. If the alarm is restored, or if the alarm becomes active again, the device must use this unique ID again. The external device is therefore responsible for ensuring that each alarm has its own ID. So all these unique ID’s must be saved by the external device.

Active alarms are stored in volatile memory, which means that these active alarms are gone when the Octalarm-Touch Pro has to restart.

The external device is also responsible for keeping the locations and categories in sync with the locations and categories created in the device. This means that the external device must be able to sync by retrieving the locations and categories (GET method), and add or modify the new locations and categories as needed (POST or POST method).

Resending alarms

Active alarms are stored in volatile memory, which means that these active alarms are gone when the Octalarm-Touch Pro must restart. However, normally the Octalarm-Touch Pro will not restart spontaneously, but there is always the possibility that a user restarts the device.

To prevent those alarms will disappear in that situation, it is recommended to resend the alarm periodically via the REST API (as long as the alarm is active). For example, the period that the alarm is resend could be 1 minute. As soon as the alarm recovers, this periodic repeating should, of course, stop.

By using the exact same values and ID's, resending the alarm has no effect on the current alarm process. But should the device unexpectedly restart, it will result in a new alarm process. Regular resending therefore provides additional operational reliability.

10 Limits

At the moment, the REST API is only available on the Octalarm-Touch Pro and ARA-Touch Pro. Both diallers have two Ethernet ports, Eth0 and Eth1. Both ports can be used to communicate with the REST API.

There are limits for the maximum number of interfaces, locations and categories: 8 REST API's with 16 global locations and 128 global categories. So those 16 locations can be added to just one REST API, or they can be divided among all REST APIs. The same applies for the categories. The maximum of 128 categories can be added to one location, or divided among several locations.

Please contact us if you think these licenses are not sufficient for your application.

Besides this, there is a limit of 100 unique alarms that can be active simultaneously. If more alarms are activated, an "ERROR 500" is returned.

11 Release notes

Known Issues

This is a summary of current active issues:

1/ The REST API always responds with JSON data regardless of the Content-Type setting.
2/ While testing the API using the OpenApi (Swagger), there may be no response if the input is incorrect.

New features/fixes

Software version	Description
2.4.5	The version API has been improved with the ability to check hardware and branding. Very useful to check if the correct configuration is connected.
2.4.4	After updating to version 2.4.4, old tokens are no longer valid. The old tokens contain an error, therefore they need to be replaced with new tokens. Communication can be restored by using a new token, that can be generated in the menu 'Octalarm Link' with the button 'Regenerate token'.
	The limit of the maximum number of categories has been changed: • the limit of maximum categories has been increased from 8 to 128 global categories
2.4.3	After updating to version 2.4.3, old tokens will no longer be valid. This was caused by a update of a library. Communication can be restored by using a new token, that can be generated in the menu 'Octalarm Link' with the button "Regenerate token".
	The maximum limit of some elements of the Alarm API have been changed: • the limit for 'message' has been increased from 40 to 100 characters • the limit for 'name' has been increased from 20 to 40 characters • the limit for 'unique_id' has been increased from 50 to 120 characters