Additional Information				
Stock No.	X9751375.01	Order No.		MICRO-EPSILON
Product	Interface Instructions for colorSENSOR CFO100 CFO100(100) CFO200 CFO200(100)	B-Order No.		
Main Operating Instructions		colorSENSOR CFO – X9751375		

Interface Instructions for the colorSENSOR CFO

Inhalt

1 REST-API communication with the controller

1.1 Basic informations

The HTTP-based API 1 enables the controller configuration or current states to be queried and changed.

It is the primary interface, via which all details of the controller are accessible. For example, the integrated web application, RS232, USB, MEDAQULib and Modbus also communicate with the controller exclusively via the API.

The API is REST-like and supports the following HTTP requests to distinguish actions with different effects:

- GET
  - Status of a resource (e.g. a current sample); GET queries are free of side effects and are sometimes saved in cache memory.
- POST
  - Create a new resource (e.g. teach in a new colour or colour group).Erzeugen einer neuen Ressource (z.B. eine neue Farbe oder Farbgruppe einlernen).
- PUT
  - Changing a resource (e.g. changing a tolerance in a colour group / matcher
- DELETE
  - Deleting, resetting or emptying a resource (e.g. deleting a colour or resetting to the factory setting).

1) Version 1.5.10

1.1.1 API Endpoint

The API endpoints are to be understood here as an addition to the basic URL (http://). As an example, the browser call (GET-request) for the colourSENSOR CFO to query the initial configuration is mentioned here:

http://169.254.168.150/api/peripherals/outputs

Basis-URL | API-Endpunkt

http://{sensor}/api

sensor String, **required**	Hostname or IP adress of he controller

1.1.2 Examples for manual query

The following manual queries are only intended to illustrate the exemplary use of the API.

For integration into own tools, most programming languages and development environments typically provide HTTP programming interfaces for easy access.

The following recommended tools for API-querying are not mandatory, but can be used.

Environment	Tool recommendation
Command line (CLI)	cURL
Windows PowerShell	Invoke-RestMethod
Chrome/Chromium	Postman

1.1.3 Query example

Every web browser uses GET requests to retrieve content. With tools, it is easy to send requests with other HTTP verbs as well. In application to the colourSENSOR CFO, for example, the request results in

```
curl -X GET http://169.254.168.150/api/sensor/samples/current
```

the following response in form of a JSON-object:

```
{
 "data": {
  "timestamp": 3145601368.0,
  "uuid": "0f721d83-f3c0-4584-8913-a51a5b842784",
  "transformed_color": {
   "values": [ 99.953887939453125, -0.0064074993133544922,
0.017380714416503906 ]
  },
  "corrected_color": {
   "values": [ 0.79777300357818604, 0.74252212047576904,
0.28755432367324829 ]
  },
  "detection": {
   "matcher": null,
   "output_pattern": { "states": [ false, false, false ] },
   "distances": [ null, null, null ]
  },
  "representations": {
   "RGB": [ 0.9994870320649919, 0.99951960105113213, 0.99927028464270895 ]
  },
  "inputs": {
   "trigger_0_down": true,
   "trigger_1_down": true,
   "trigger_2_up": false,
   "trigger_0_up": false,
   "trigger_3_down": true,
   "trigger_3_up": false,
   "trigger_2_down": true,
   "trigger_1_up": false
  }
 },
 "errors": []
}
```

1.1.4 Quickstart via API communication

The following sequence of requests configures the sensor for the detection of a wanted color.

The request examples below are sent via the `curl` command (examples prefixed with $) or the Windows Powershell (prefixed with >).

The same requests can be used in all other programming languages or environments supporting HTTP requests, as well.

1. Clear all settings:

```
$ curl -X DELETE http://sensor/api/settings
```

```
> Invoke-RestMethod -Method DELETE -Uri http://<sensorIP>/api/settings
```

2. Place a neutral white color target in front of your sensor's optics.

3. Adjust the sensor settings to the optical setup by initiating the *Autogain* procedure:

```
$ curl -X POST http://<sensorIP>/api/sensor/detection-profiles/current/autogain
```

```
> Invoke-RestMethod -Method POST -Uri http://<sensorIP>/api/sensor/detection-profiles/current/autogain
```

4. Place your wanted target object in front of the sensor's optics.

5. Teach this color:

```
$ curl -X POST http://<sensorIP>/api/sensor/detectables
```

```
> Invoke-RestMethod -Method POST -Uri http://<sensorIP>/api/sensor/detectables
```

6. The sensor should now pull up its first output pin as long as the target is placed in front of it.

    - The first switching output pin is assigned to the first color, by default. This is configurable.

7. Request the current sample data and pick only the color values:

```
# hint: "jq" is a separate tool for querying JSON datasets
$ curl http://<sensorIP>/api/sensor/samples/current | jq .data.transformed_color.values
```

```
> (Invoke-RestMethod -Method GET -Uri http://<sensorIP>/api/sensor/samples/current).data.transformed_color
```

- Result:

```
[ 101.23353576660156, 7.889449596405029, -42.49897003173828 ]
```

1.1.5 REST-API Introduction

1.1.5.1 General Informations

Collections (arrays or lists)

- can be queried via `GET` request
- some collections allow to be filtered via query parameters, outlined in the resource documentation below
- some collections allow `POST` requests to create new items in the collection

  - properties of new items can usually be set in the `POST` request body
  - a failure to create a new item due to malformed data is indicated with a 400 HTTP status code
  - a failure to create a new item caused by too many items in the collection is indicated by a 422 HTTP status code
  - some collections can be depleted by sending a `DELETE` request to them supporting the same query parameters as `GET` requests if any
  - a `DELETE` request on the collection always returns a positive HTTP status code like 200 or 204 with no regard for the number of items that have been deleted (and even if no items have been deleted as a result of the request)

Collection Items

- Individual collection items typically can be accessed via a URL path like `/collection-path/item-id` (e.g. `/api/access/users/barbara`, where `/users` is the collection and `barbara` the item id).

  - If no item with that specific id exists, either because it has been deleted in the meantime in response to a user request or (in case of size- or time-limited collections) the item was removed, the request will be terminated with a 404 HTTP status code.
  - The data of some collections items can be updated with `PUT` requests where the body contains a JSON object with the data that should be updated. Partial updates are supported as well. Changing read-only attributes may be terminated with a 400 HTTP status code. The response for a successful update contains the state of the resource after the modification.
  - The unique id of an item (most often a `uuid`) is invariable.
  - Some collection items can be removed with a `DELETE` request.

- Some collections support more than one URL-compatible item id field. The matcher, detectable and detection profile collection items for example, can also be retrieved by their respective `alias` field.

Data Formatting Conventions

- Identifiers are in American English
- Compound words in resource urls are hyphenated (e.g. `white-reference`)
- Compound words in JSON objects or data accepted as multipart/form-data uses snake case (e.g. `white_reference`)

1.1.5.2 Response Format and Error Handling

The API will return a response containing a body with a JSON object for any given request to any known endpoint of the following form:

```
{
        "errors": [ ...errors ],
        "data": { ...data }
}
```

`data` contains the actual response payload that has been requested whereas `errors` contains any errors that the API encountered while processing the request. An error object found in the `errors` array has the following form:

```
{
        "message": String,
        "mapping": String | null,
        "code": String
}
```

Attribute	Type	Content
message	String	The `message` property contains a human-readable english description of the error. Its primary target audience is the developer working with the API and should guide them on their way to resolve the error.
mapping	String / null	The mapping property contains a valid JavaScript expression as a string that evaluates to the property on the submitted or provided object where the error was encountered. The mapping may be null if no such connection can be made. If it is set an expression like `foo[1].bar` would refer to the `bar` property in the second item of an array named `foo` found on the root object.
code	String	The `code` property can be used as an identifier to distinguish between different error types. This may be helpful if you want to display localized error messages (where you could use `code` as the key to your translation dictionary) or when used in code to change the behaviour of your application in case of an error. The code is a string representing an error class hierarchy where each error class is delimited by a dot. This is helpful when you want to start of with little set of error translations and become more exact with the descriptions at a later point in time. So instead of translating a code like `LPLC.validation.non_negative_float` you could just translate `LPLC.validation` with 'Please check your input' and add a translation for float errors later on.

Error codes are documented for each resource below but the following are the most common ones:

- `LPLC.validation`
- `LPLC.validation.missing_input`
- `LPLC.validation.readonly`
- `LPLC.validation.non_negative_float`
- `LPLC.validation.positive_integer`
- `LPLC.validation.smaller_integer`
- `LPLC.validation.single_character`
- `LPLC.validation.string`
- `LPLC.validation.boolean`
- `LPLC.format.encoding.utf8`
- `LPLC.format.malformed.json`

1.1.5.3 Switching outputs, triggers and hold time settings

Every sample period ends with the selection of the most appropriate matcher ("group of colors"). This matcher is applied to the switching outputs under certain conditions. Relevant configuration for this behavior are the following settings:

- optional triggered update of switching outputs (see *action-triggers*)
- matcher attributes *hold_time* and *reset_output_after_hold_time_expired*

The following situations and actions are used in the behavior specification below:

- *no active hold time*: The *hold_time* attribute of the most recently applied matcher was zero. Thus there is currently no hold time configured until another matcher with a non-zero hold time is applied.
- *hold time is expired*: The most recently applied matcher had a hold time greater than zero, but this hold time elapsed since this matcher was applied. Thus an hold time was previously active, but it expired in the meantime.
- *reset_output_after_hold_time_expired* is on/off: When a matcher is applied (see below), then the attribute *reset_output_after_hold_time_expired* is memorized until another matcher is selected. The on/off state refers to this memorized value.
- *unchanged detected matcher*: The most recently applied matcher and the currently detected matcher are the same.
- *new detected matcher*: The most recently applied matcher and the currently detected matcher are not the same.
- action *do nothing*: The switching outputs and the currently memorized hold time settings stay unchanged.
- action *apply new matcher*: Memorize the matcher attributes *hold_time* and *reset_output_after_hold_time_expired* as current hold time settings and set the switching outputs as specified in the *output_pattern* of the matcher.
- action *apply* '*no match*': Memorize the detection profile attribute *non_matching_hold_time* as the current hold time. Memorize *false* for *reset_output_after_hold_time_expired*. Set the switching outputs as specified in the detection profile attribute *non_matching_output*.

The behaviour with and without *triggered updates of switching outputs* differs significantly. Thus both situations are specified separately below.

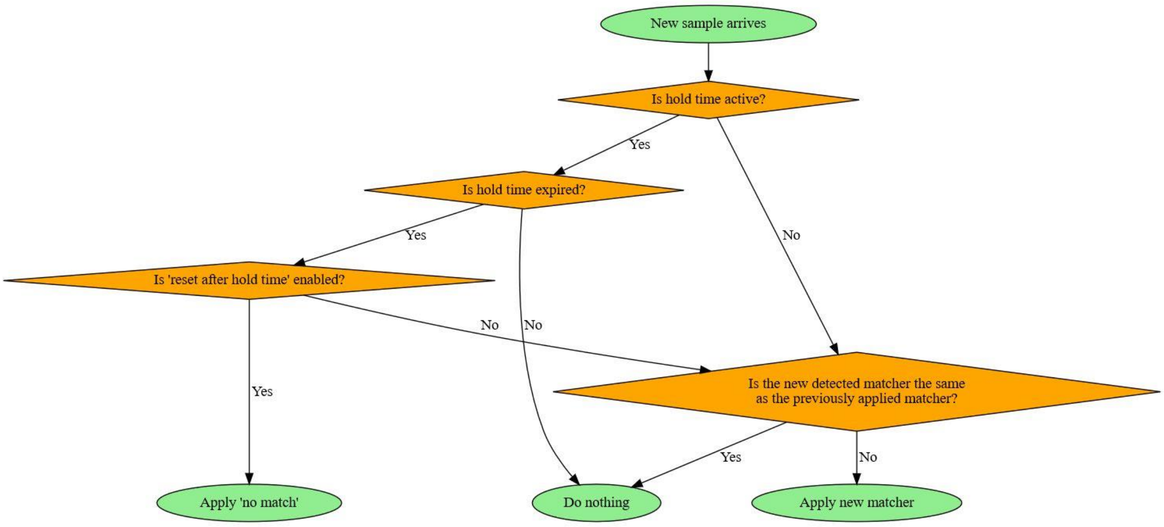
Disabled triggered update of switching outputs

Triggered updates are disabled if none of the trigger events (e.g. a rising edge of the first input line) is configured with the enable_switching_output action (see `/api/actions`).

The following table lists the specific behavior based on the currently active hold time settings and depending on the currently detected matcher.

Currently active hold time settings	**Action for unchanged detected matcher**	**Action for new detected matcher**
no active hold time	do nothing	apply new matcher
hold time is expired; *reset_output_after_hold_time_expired* is off	do nothing	apply new matcher
hold time is expired; *reset_output_after_hold_time_expired* is on	apply *no match*	apply *no match*
hold time is not expired	do nothing	do nothing

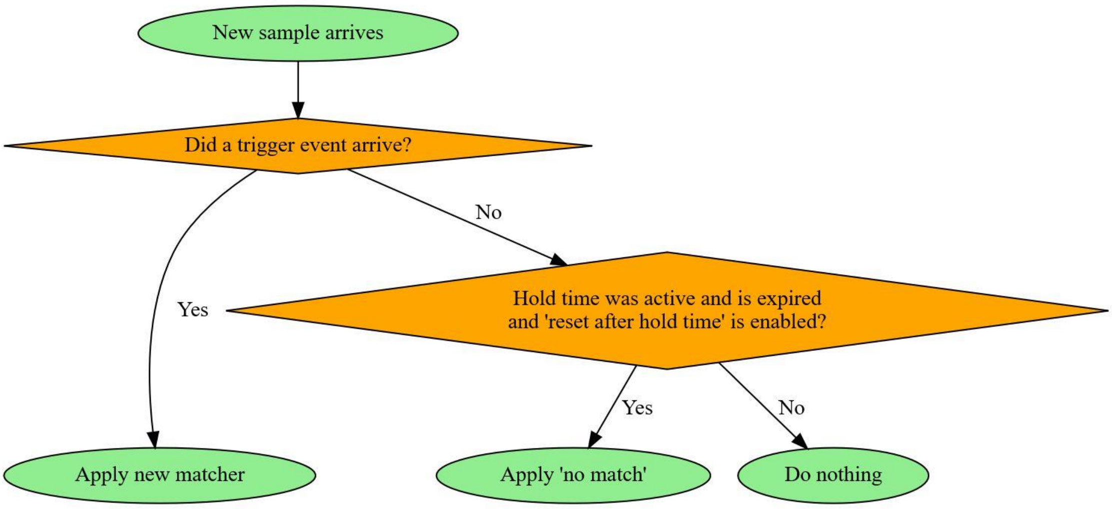
The following flowchart visualizes the decisions and actions when applying a matcher. It is a different representation of the table above.



Enabled triggered update of switching outputs

Triggered updates are enabled if any of the trigger events (e.g. a rising edge of the first input line) is configured with the *enable_switching_output* action (see `/api/actions`)

The following flowchart visualizes the decisions and actions when applying a matcher.


New sample arrives
Did a trigger event arrive?
No
Yes
Hold time was active and is expired and 'reset after hold time' is enabled?
Yes
No
Apply new matcher
Apply 'no match'
Do nothing

1.1.5.4 Websockets

The websocket provided by the API can be used to develop highly interactive frontends for the sensors that work without polling the endpoints of the REST-like resources.

Aside from a stream of samples that the sensors pushes to the websocket it also broadcasts information about events like newly taught colors or a change of the sensors configuration.

The websocket is developed as a complement to the resources and not as a replacement. The primary interface to control the sensor is and always will be the resources outlined below.

Requests to any of the endpoints under `/websocket` do not reset the session timeout.

Overview

You can access the websocket on the `/websocket/notifications/websocket` path. The data transmitted on this channel resembles a stream of information. In order to easily identify relevant packages, every payload transmitted through the socket is encapsulated in a JSON object containing the following properties:

`- id`

- The unique identifier of this particular websocket packet

`- source`

- The source this packet originates from or the reason for its transmission (e.g. `detection_profile.matcher`)

`- timestamp`

- The sensors uptime when the packet is sent over the websocket.

`- payload`

- An object containing an `event` property (with values like `changed` or `created`), and an optional `uuid` and `data` attribute that represent the object described by the `event` from the `source`.

  One of the fields `added_items`, `changed_items` and `removed_items` may be present if `source` refers to a collection. In this case the value of this field is a list of identifiers belonging to the affected items of the collection. This allows clients to synchronize their data model without requesting the full collection after each collection-related notification. If a collection-related notification does not contain any of the fields above, then the scope of changes is unspecific and thus a full retrieval of the collection may be necessary.

Fallback

In order to support older Browsers the /websocket/notifications endpoint is compatible with the SockJS Client-Bibliothek library that implements fallback techniques like XHR Streaming, JSONP, Long Polling and others.

1.1.5.5 Networking and Discovery

Network discovery

The sensor announces itself in the local network via the following protocols:

- zeroconf / avahi broadcasts
- SSDP

The SSDP protocol allows discovery of the sensors via the Windows network neighborhood.
The zeroconf protocol allows discovery on Linux, MacOS and mobile devices.

Automatically assigned link-local address

The sensor is reachable via its explicitly configured IPv4 and IPv6 addresses as well as via its automatically configured link-local address. This address belongs to the subnet "fe80::/10" with the local address part being based on the MAC address of the sensor (see "EUI-64"). The link-local address of the sensor is usable in all networks independent of the sensor configuration. Thus it provides a stable address under all circumstances.

Link-local addresses in general need to be suffixed with the local network identifier.

Examples

- Windows: `fe80::1234:56ff:fe78:90ab%0`
- Linux / MacOS / Android: `fe80::1234:56ff:fe78:90ab%eth0`

The network identifier (suffix after "%") in the examples above need to be adjusted to the local setup of the device connecting to the sensor.

1.2 Resources

1.2.1 Device Information

Some constant properties describe the individual device itself and include information about the model and vendor.

1.2.1.1 Get Device Properties

```
GET / device
```

Return invariable properties of the device.

Response

Code	Body	application/json	
200	Properties (object)	DeviceInformation	
	data DeviceInformation, **required**	**id** DeviceSerialNumber, **required**	Serial Number
		model_name string, **required**	human-readable name of the device model
		model_key string, **required**	unique id of the device model
		variant any of string or null, **required**	indicates a special series of a model
		vendor_key DeviceVendorKey, **required**	Unique key identifying the organization distributing this device
		vendor_name DeviceVendorName, **required**	Name of vendor of this device
		device_id DeviceSerialNumber, optional, Deprecated	Deprecated: use "id" instead.
		model string, optional, Deprecated	Deprecated: use "model_name" instead.
		vendor DeviceVendorName, optional, Deprecated	Deprecated: use "vendor_name" instead.
4XX	**errors** Array of Error, **required**	Error[]	
		code string, optional	machine-readable unique error code
		mapping string, optional	a reference to the parameter that caused the error
		message string, optional	human-readable error description

Example

```
{"data":
     {"model_key": "me_cfo_200",
     "model": "CFO200",
     "vendor": "Micro-Epsilon Eltrotec GmbH",
     "device_id": "7455301813",
     "vendor_key": "eltrotec",
     "model_name": "CFO200",
     "vendor_name": "Micro-Epsilon Eltrotec GmbH",
     "id": "7455301813",
     "variant": "100"},

 "errors": []}
```

1.2.2 Sensor

Query and modify all details of sensoric configuration and operation.

1.2.2.1 Retrieve Sensor Samples

```
GET / sensor / samples
```

Returns a list of samples from the color detection.

When no additional query parameters are passed the collection contains samples from the past. You can activate sample-streaming with the `stream` query parameter. In this case only new samples will be returned, as they become available.

Samples are implemented as ring buffer. Old samples will be removed from the collection as new samples are added.

Request

Query Parameters	
stream number, one of [`0`, `1`] , default: `0`, optional	Controls whether or not stream-mode is activated. When streaming is activated only the sensor will contionously transmit new samples to the client. The number of samples that are transmitted can be controlled with the `stream_count` query parameter. When streaming is deactivated (which is the default), only past samples are returned.
stream_count integer , default: `0`, optional	Determines how many samples should be transmitted before the connection is terminated when stream-mode has been activated. The default is to stream indefinitely.
format string, one of [`json`, `csv`] , default: `json`, optional	Determines the output format of the samples when stream-mode has been activated. If `csv` is selected the first transmitted line are the column headers. Headers are based on the default JSON representation and use the syntax also used by JavaScript. Given the example `representations.RGB[0]` the value for this header would refer to the first item in the RGB representation.
delimiter string , default: `,` , minimum length: 1, maximum length: 1, optional	Determines the column delimiter when `csv` has been selected as output-format. If you want to use a semicolon as delimiter be sure to url-encode it first `(%3B)`. Otherwise it's interpreted as query parameter separator. Be aware that even though unicode characters are allowed by the API you should restrict yourself to one-byte characters as most tools will fail to use delimiters that use two or more bytes.

Response

Code	**Body**	**application/json**				
200	Properties (object)	Data				
	data object, **required**	**samples** Array of ColorDetectionResult, **required**	ColorDetectionResult[]			
			uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
			timestamp TimestampBackendUptime (number), **required**	The timestamp (given in microseconds) is based on the uptime of the internal analog sensor backend. It may get reset to zero under specific conditions.		
			corrected_color CorrectedColor, **required**	Representation of a color in the colorspace XYZ.		
				values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace	
			transformed_color TransformedColor, **required**	A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the		

Code	Body	application/json				
				colorspace.axes property of currently used detection profile.		
				values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace	
			representations ColorRepresentations, **required**	Pre-calculcated visual representations of a color suitable for rendering		
				RGB Array of number, minimum items: 3, maximum items: 3, **required**	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.	
			inputs InputsState, **required**	The state of all inputs during a given period is specified by a list of possible events combined with a boolean value indicating, if the given event occurred within the period.		
				// boolean, **required**	The boolean value indicates whether the named input event occurred during the last period.	
			detection ColorMatchingResult, **required**	After each sampling period the retrieved color value is compared to the stored detectables (color positions). Detectables are ignored, if the tolerance shape of their corresponding matcher does not encompass the current sample. Finally the closes suitable detectable is selected as the winner of the color matching operation. The corresponding matcher determines the state of the sensor for the duration of the next sampling period.		
				matcher any of UUID (string) or null, optional, Deprecated	Deprecated: use "chosen_matcher_id" instead	
				chosen_matcher_id any of UUID (string) or null, **required**	unique identifier of the selected matcher	
				distances Array of any of number or null, **required**	Distance between the sample's color position and the selected matcher's closest color position along the three axes of the color space. The array contains three 'null' values, if no suitable matcher was found for the current color sample.	
				output_pattern CurrentSwitchingOutputsState, **required**	Currently active state of the Switch-	

Code	Body	application/json				
					ing Outputs. Beware that this may deviate from the specified output states of the current best matcher, since settings like *triggered input* or *hold time* influence update process for the Switching Outputs.	
					states Array of any of boolean or null, **required**	List of True/False values describing the current states of the Switching Outputs
			signal_level number, **required**	The signal level indicates the usage of the internal ADC sampling range. This		
	errors Array of Error, **required**	Error[]				
		code String, optional	machine-readable unique error code			
		mapping String, optional	a reference to the parameter that caused the error			
		message String, optional	human-readable error description			

1.2.2.2 Retrieve latest Sensor Sample

`GET / sensor / samples / current`

Returns the latest sample.

Be aware that the same sample may be returned for successive requests if no new samples arrived meanwhile. While the sensor is performing auto-gain or is over-saturated, it will return the last valid sample that was processed.

The result is empty (`null`), while the sensor is processing a configuration change request.

Response

Code	Body	application/json			
200	Properties (object)				
	data Any of ColorDetectionResult or null, **required**	ColorDetectionResult			
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
		timestamp TimestampBackendUptime (number), **required**	The timestamp (given in microseconds) is based on the uptime of the internal analog sensor backend. It may get reset to zero under specific conditions.	.	

Code	Body	application/json			
		corrected_color CorrectedColor, **required**	Representation of a color in the colorspace XYZ.	.	
			values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace	
		transformed_color TransformedColor, **required**	A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the `colorspace.axes` property of currently used detection profile.		
			values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace	
		representations ColorRepresentations, **required**	Pre-calculcated visual representations of a color suitable for rendering		
			RGB Array of number, minimum items: 3, maximum items: 3, **required**	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.	
		inputs InputsState, **required**	The state of all inputs during a given period is specified by a list of possible events combined with a boolean value indicating, if the given event occurred within the period.		
			// boolean, **required**	The boolean value indicates whether the named input event occurred during the last period.	
		detection ColorMatchingResult, **required**	After each sampling period the retrieved color value is compared to the stored detectables (color positions). Detectables are ignored, if the tolerance shape of their corresponding matcher does not encompass the current sample. Finally the closes suitable detectable is selected as the winner of the color matching operation. The corresponding matcher determines the state of the sensor for the duration of the next sampling period.		
			matcher any of UUID (string) or null, optional, Deprecated	Deprecated: use "chosen_matcher_id" instead	
			chosen_matcher_id any of UUID (string) or null, **required**	unique identifier of the selected matcher	
			distances Array of any of number or null, **required**	Distance between the sample's color position and the selected matcher's closest color position along the three axes of the color space. The array contains three 'null' values, if no suitable matcher was found for the current color sample.	
			output_pattern CurrentSwitchingOutputsState, **required**	Currently active state of the Switching Outputs. Beware that this may deviate from the specified output states of the current best matcher, since settings like	

Code	Body	application/json			
				triggered input or *hold time* influence update process for the Switching Outputs.	
				states Array of any of boolean or null, **required**	List of True/False values describing the current states of the Switching Outputs
		signal_level number, **required**	The signal level indicates the usage of the internal ADC sampling range. This		
	errors Array of Error, **required**	Error[]			
		code String, optional	machine-readable unique error code		
		mapping String, optional	a reference to the parameter that caused the error		
		message String, optional	human-readable error description		

1.2.2.3 Retrieve all Matchers (color groups)

`GET / sensor / matchers`

Matchers (color groups) describe a color detection result. A Matcher contains details regarding the wanted behaviour of the Switching Outputs and a list of *Detectables* (color positions).

The most basic setup of a colorsensor could involve only a single color group containing all *positive* detection results. All samples that are not matched by this color group would indicate a problem of the monitored real-world process.

A more advanced usage of Matchers could additionally include a Matcher for the different acceptable background colors between real target objects (e.g. the color of the conveyor belt). Thus the Switching Outputs of the sensor could indicate whether a *positive*, a *neutral* or a *negative* real-world event was sampled.

Of course, every Matcher may also simply contain exactly one color position, in order to allow fine-grained classification of the target's appearance.

The `tolerance` field of a matcher describes the shape and the dimensions of the part of the color-space that is covered by this matcher. Only color positions within this space may cause a match for this matcher. A tolerance is specified by a `shape` and a dictionary of `limits`. Both attributes need to be specified. An empty dictionary of limits is interpreted as the default `limits` for the requested shape.

Request

Query Parameters	
profile_id String, optional	Filter ColorMatchers by the given *Detection Profile ID*. Only ColorMatchers that are part of the given Detection Profile will be returned.

Response

Code	Body	application/json				
200	Properties (object)					
	data Object, **required**	data				
		matchers Array of ColorMatcher, **required**	ColorMatcher[]			

Code	Body	application/json				
			uuid UUID (string), pattern: ^[a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
			alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.		
			name String, **required**	human-readable name of the matcher		
			tolerance Any of InfiniteColorTolerance, SphereColorTolerance, CylinderColorTolerance or BoxColorTolerance, **required**	Specification of a geometric shape and its dimensions in the current colorspaces.		
				InfiniteColorTolerance		
				limits Object, **required**	limits	
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities` .	
				SphereColorTolerance		
				limits Object, **required**	limits **radius** Numer, **required**	
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.	
				CylinderColorTolerance		
				limits Object, **required**	limits **radius** Number, required **half_height** Number, required	
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`	
				BoxColorTolerance		
				limits Object, **required**	limits **half_edges** Array of number, minimum items: 3, maximum items: 3, **required**	

Code	Body	application/json				
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.	
			output_pattern WantedSwitchingOutputsState, **required**	The combination of tri-state values describes a logical state of the switching outputs of the sensor. The states `true` or `false` cause the output to go up or down. The state `null` keeps the previous state of the output unchanged.		
				uuid UUID (string), pattern: ^[a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
				states Array of any of boolean or null, **required**	List of True/False/Null values describing the wanted states of the Switching Outputs	
			hold_time HoldTime (number), maximum: 3153600000, **required**	Minimum duration (in seconds) of a matcher's output setup being applied after detection.		
			reset_output_after_hold_time_expired Boolean, default: false, **required**	Controls if the output should be reset after the hold time passed. This is helpful if you only sample by triggering inputs and wish to reset the outputs afterwards.		
			signal_color Any of string or null, **required**	A custom color name. How and what color will be displayed is defined by the client.		
	errors Array of Error, required	Error []				
		code String, optional	machine-readable unique error code			
		mapping String, optional	a reference to the parameter that caused the error			
		message String, optional	human-readable error description			

1.2.2.4 Create a new Matcher (color group)

`POST / sensor / matchers`

Stores a new matcher (color group) on the sensor. In order to add colors to it, use the `/api/sensor/detectables` endpoint.

Request

Body	application/json		
Properties (ColorMatcher)			
	uuid UUID (string), pattern: ^[a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	

Body	**application/json**		
	alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.	
	name String, **required**	human-readable name of the matcher	
	tolerance Any of InfiniteColorTolerance, SphereColorTolerance, CylinderColorTolerance or BoxColorTolerance, **required**	Specification of a geometric shape and its dimensions in the current colorspaces.	
		InfiniteColorTolerance	
		limits Object, **required**	limits
		shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities` .
		SphereColorTolerance	
		limits Object, **required**	limits **radius** Numer, **required**
		shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities` .
		CylinderColorTolerance	
		limits Object, **required**	limits **radius** Number, required **half_height** Number, required
		shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities` .
		BoxColorTolerance	
		limits Object, **required**	limits **half_edges** Array of number, minimum items: 3, maximum items: 3, required
		shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities` .

Body	application/json		
	output_pattern WantedSwitchingOutputsState, **required**	The combination of tristate values describes a logical state of the switching outputs of the sensor. The states `true` or `false` cause the output to go up or down. The state `null` keeps the previous state of the output unchanged.	
		uuid UUID (string), pattern: ^ [a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
		states Array of any of boolean or null, **required**	List of True/False/Null values describing the wanted states of the Switching Outputs
	hold_time HoldTime (number), maximum: 3153600000, **required**	Minimum duration (in seconds) of a matcher's output setup being applied after detection.	
	reset_output_after_hold_time_expired Boolean, default: false, required	Controls if the output should be reset after the hold time passed. This is helpful if you only sample by triggering inputs and wish to reset the outputs afterwards.	
	signal_color Any of string or null, **required**	A custom color name. How and what color will be displayed is defined by the client.	

Example

```
{
  "uuid": "9ffaa31f-8011-44f5-bb2a-f91e4be50764",
  "alias": 6,
  "name": "clean bottle cap",
  "tolerance": {
    "limits": {
      "radius": 2,
      "half_height": 4
    },
    "shape": "cylinder"
  },
  "output_pattern": {
    "uuid": "1adc74e2-96ac-4761-b9e6-2d93e02d9244",
    "states": [
      true,
      false,
      false
    ]
  },
  "hold_time": 0,
  "reset_output_after_hold_time_expired": false,
  "signal_color": null
}
```

Response

Code	Body	application/json				
200 400	Properties (object)					
	data ColorMatcher, **required**	A matcher represents a distinguished detection result and the wanted behaviour of the sensor whenever it is encountered.				
			ColorMatcher[]			

Code	Body	application/json				
			uuid UUID (string), pattern: ^[a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
			alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.		
			name String, **required**	human-readable name of the matcher		
			tolerance Any of InfiniteColorTolerance, SphereColorTolerance, CylinderColorTolerance or BoxColorTolerance, **required**	Specification of a geometric shape and its dimensions in the current colorspaces.		
				InfiniteColorTolerance		
				limits Object, **required**	limits	
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities` .	
				SphereColorTolerance		
				limits Object, **required**	limits **radius** Numer, **required**	
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities` .	
				CylinderColorTolerance		
				limits Object, **required**	limits **radius** Number, required **half_height** Number, required	
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.	
				BoxColorTolerance		
				limits Object, **required**	limits **half_edges** Array of number, minimum items: 3, maximum items: 3, **required**	

Code	Body	application/json				
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities` .	
			output_pattern WantedSwitchingOutputsState, **required**	The combination of tri-state values describes a logical state of the switching outputs of the sensor. The states true or false cause the output to go up or down. The state null keeps the previous state of the output unchanged.		
				uuid UUID (string), pattern: ^[a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
				states Array of any of boolean or null, **required**	List of True/False/Null values describing the wanted states of the Switching Outputs	
			hold_time HoldTime (number), maximum: 3153600000, **required**	Minimum duration (in seconds) of a matcher's output setup being applied after detection.		
			reset_output_after_hold_time_expired Boolean, default: false, **required**	Controls if the output should be reset after the hold time passed. This is helpful if you only sample by triggering inputs and wish to reset the outputs afterwards.		
			signal_color Any of string or null, **required**	A custom color name. How and what color will be displayed is defined by the client.		
	errors Array of Error, required	Error []				
		code String, optional	machine-readable unique error code			
		mapping String, optional	a reference to the parameter that caused the error			
		message String, optional	human-readable error description			
		May return the following error codes `LPLC.validation.collection_size_exceeded`				

1.2.2.5 Remove multiple or all ColorMatchers

`DELETE / sensor / matchers`

Remove a selection of ColorMatchers either based on a given filter argument (if supported for this collection) or remove all ColorMatchers from the collection.

All delete requests result in an empty success response (204). This is even valid for a non-filtered `DELETE` request against an empty collection or for a filtered `DELETE` request against a collection without ColorMatchers matching the filter.

Response

Code	
204	The empty response indicates success

1.2.2.6 Retrieve Matcher (color group) Details

`GET / sensor / matchers / {itemId}`

Returns the current configuration of a matcher.

Request

Path Variables
itemId String, **required**

Response

Code	**Body**	**application/json**			
200 400	Properties (object)				
	data ColorMatcher, **required**	A matcher represents a distinguished detection result and the wanted behaviour of the sensor whenever it is encountered.			
			ColorMatcher[]		
			uuid UUID (string), pattern: ^ [a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
			alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.	
			name String, **required**	human-readable name of the matcher	
			tolerance Any of InfiniteColorTolerance, SphereColorTolerance, CylinderColorTolerance or BoxColorTolerance, **required**	Specification of a geometric shape and its dimensions in the current colorspaces.	
				InfiniteColorTolerance	
				limits Object, **required**	limits
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities .`
				SphereColorTolerance	
				limits	limits

Code	Body	application/json			
				Object, **required**	**radius** Numer, **required**
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.
				CylinderColorTolerance	
				limits Object, **required**	limits **radius** Number, **required** **half_height** Number, **required**
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.
				BoxColorTolerance	
				limits Object, **required**	limits **half_edges** Array of number, minimum items: 3, maximum items: 3, required
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.
			output_pattern WantedSwitchingOutputsState, **required**	The combination of tri-state values describes a logical state of the switching outputs of the sensor. The states `true` or `false` cause the output to go up or down. The state `null` keeps the previous state of the output unchanged.	
				uuid UUID (string), pattern: ^[a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
				states Array of any of boolean or null, **required**	List of True/False/Null values describing the wanted states of the Switching Outputs
			hold_time HoldTime (number), maximum: 3153600000, **required**	Minimum duration (in seconds) of a matcher's output setup being applied after detection.	
			reset_output_after_hold_time_expired Boolean, default: false, **required**	Controls if the output should be reset after the hold time passed. This is helpful if you only sample by triggering inputs and wish to reset the outputs afterwards.	
			signal_color Any of string or null, **required**	A custom color name. How and what color will be displayed is defined by the client.	
	errors Array of Error, **required**	Error []			
		code String, optional	machine-readable unique error code		

Code	Body	application/json			
		mapping String, optional	a reference to the parameter that caused the error		
		message String, optional	human-readable error description		
		May return the following error codes `LPLC.not_found.collection.item`			

1.2.2.7 Delete a Matcher (color group)

`DELETE / sensor / matchers / {itemId}`

Deletes the matcher and all associated detectables.

Request

Path variables
itemId String, **required**

Response

Code	
204	The empty response indicates success
	May return the following error codes `LPLC.not_found.collection.item`

1.2.2.8 Update the Matcher (color group) Configuration

`PUT / sensor / matchers / {itemId}`

Update the matcher with a new configuration.

Request

Path Variables
itemId String, **required**

Body	application/json		
Properties (Color Matcher)			
	uuid UUID (string), pattern: ^[a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
	alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.	
	name String, **required**	human-readable name of the matcher	
	tolerance Any of InfiniteColorTolerance, SphereColorTolerance, CylinderColorTolerance or BoxColorTolerance, **required**	Specification of a geometric shape and its dimensions in the current colorspaces.	
		InfiniteColorTolerance	
		limits Object, **required**	limits
		shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities` .
		SphereColorTolerance	
		limits	limits

Body	application/json		
		Object, **required**	**radius** Numer, **required**
		shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities .`
		CylinderColorTolerance	
		limits Object, **required**	limits **radius** Number, **required** **half_height** Number, **required**
		shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities .`
		BoxColorTolerance	
		limits Object, **required**	limits **half_edges** Array of number, minimum items: 3, maximum items: 3, **required**
		shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities .`
	output_pattern WantedSwitchingOutputsState, **required**	The combination of tristate values describes a logical state of the switching outputs of the sensor. The states `true` or `false` cause the output to go up or down. The state `null` keeps the previous state of the output unchanged.	
		uuid UUID (string), pattern: ^[a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
		states Array of any of boolean or null, **required**	List of True/False/Null values describing the wanted states of the Switching Outputs
	hold_time HoldTime (number), maximum: 3153600000, **required**	Minimum duration (in seconds) of a matcher's output setup being applied after detection.	
	reset_output_after_hold_time_expired Boolean, default: false, required	Controls if the output should be reset after the hold time passed. This is helpful if you only sample by triggering inputs and wish to reset the outputs afterwards.	
	signal_color Any of string or null, **required**	A custom color name. How and what color will be displayed is defined by the client.	

Examples

```
{
  "uuid": "9ffaa31f-8011-44f5-bb2a-f91e4be50764",
  "alias": 6,
  "name": "clean bottle cap",
  "tolerance": {
    "limits": {
      "radius": 2,
      "half_height": 4
    },
    "shape": "cylinder"
  },
  "output_pattern": {
    "uuid": "1adc74e2-96ac-4761-b9e6-2d93e02d9244",
    "states": [
      true,
      false,
      false
    ]
  },
  "hold_time": 0,
  "reset_output_after_hold_time_expired": false,
  "signal_color": null
}
```

Response

Code	Body	application/json			
200 400 404	Properties (object)				
	data ColorMatcher, **required**	A matcher represents a distinguished detection result and the wanted behaviour of the sensor whenever it is encountered.			
			ColorMatcher		
			uuid UUID (string), pattern: ^ [a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
			alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.	
			name String, **required**	human-readable name of the matcher	
			tolerance Any of InfiniteColorTolerance, SphereColorTolerance, CylinderColorTolerance or BoxColorTolerance, **required**	Specification of a geometric shape and its dimensions in the current colorspaces.	
				InfiniteColorTolerance	
				limits Object, **required**	limits
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.
				SphereColorTolerance	
				limits Object, **required**	limits **radius** Numer, **required**
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.
				CylinderColorTolerance	
				limits	limits

Code	Body	application/json			
				Object, **required**	**radius** Number, **required** **half_height** Number, **required**
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities` .
				BoxColorTolerance	
				limits Object, required	limits **half_edges** Array of number, minimum items: 3, maximum items: 3, **required**
				shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities` .
			output_pattern WantedSwitchingOutputsState, **required**	The combination of tristate values describes a logical state of the switching outputs of the sensor. The states `true` or `false` cause the output to go up or down. The state `null` keeps the previous state of the output unchanged.	
				uuid UUID (string), pattern: ^[a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
				states Array of any of boolean or null, **required**	List of True/False/Null values describing the wanted states of the Switching Outputs
			hold_time HoldTime (number), maximum: 3153600000, **required**	Minimum duration (in seconds) of a matcher's output setup being applied after detection.	
			reset_output_after_hold_time_expired Boolean, default: false, **required**	Controls if the output should be reset after the hold time passed. This is helpful if you only sample by triggering inputs and wish to reset the outputs afterwards.	
			signal_color Any of string or null, **required**	A custom color name. How and what color will be displayed is defined by the client.	
	errors Array of Error, **required**	Error []			
		code String, optional	machine-readable unique error code		
		mapping String, optional	a reference to the parameter that caused the error		
		message String, optional	human-readable error description		
		May return the following error codes `LPLC.not_found.collection.item`			

1.2.2.9 Retrieve ColorDetectables

`GET / sensor / detectables`

Detectables describe positions in the currently selected colorspace. Each detectable is part of a Matcher. Every Matcher may contain zero or more Detectables.

Detectables are used to determine the most suitable Matcher for a sampled color. This closest match defines the result of a sampling period and thus the behaviour of the sensor during the next sampling period.

Request

Query Parameters	
matcher_id UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	Filter detectables by the given Matcher ID. Only Detectables that are part of the given Matcher will be returned.
profile_id String, optional	Filter ColorDetectables by the given Detection Profile ID. Only ColorDetectables that are part of the given Detection Profile will be returned.

Response

Code	**Body**	**application/json**			
200	Properties (object)				
	data Object, **required**	**detectables** Array of ColorDetectable, **required**			
			ColorDetectable[]		
			uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
			alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.	
			matcher_id UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	reference to the Matcher containing this Detectable	
			color TransformedColor, **required**	A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the `colorspace.axes` property of currently used detection profile.	
				color	
				values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace
			representations ColorRepresentations, optional, read-only	Pre-calculcated visual representations of a color suitable for rendering	
				representations	
				RGB Array of number, minimum items: 3, maximum items: 3, **required**	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.
	Errors Array of Error, **required**	Error[]			
		code String, optional	machine-readable unique error code		

		mapping String, optional	a reference to the parameter that caused the error		
		message String, optional	human-readable error description		

1.2.2.10 Remove multiple or all ColorDetectables

`DELETE / sensor / detectables`

Remove a selection of ColorDetectables either based on a given filter argument (if supported for this collection) or remove all ColorDetectables from the collection.

All delete requests result in an empty success response (204). This is even valid for a non-filtered `DELETE` request against an empty collection or for a filtered `DELETE` request against a collection without ColorDetectables matching the filter.

Request

Query Parameters	
matcher_id UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	Remove only detectables with the given *Matcher* ID.
profile_id String, optional	Filter ColorDetectables by the given Detection Profile ID. Only ColorDetectables that are part of the given Detection Profile will be returned.

Response

Code	
204	The empty response indicates success.

1.2.2.11 Create ColorDetectables

`POST / sensor / detectables`

Create a new ColorDetectable.

All supported data attributes in the body of the request are optional.

Request

Body	**application/json**		
Properties (ColorDetectable)	**uuid** UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
	alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.	
	matcher_id UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	reference to the Matcher containing this Detectable	
	color TransformedColor, **required**	A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the `colorspace.axes` property of currently used detection profile.	
		color	
		values	Location in a colorspace

		Array of number, minimum items: 3, maximum items: 3, **required**	
	representations ColorRepresentations, optional, read-only	Pre-calculcated visual representations of a color suitable for rendering	
		representations	
		RGB Array of number, minimum items: 3, maximum items: 3, **required**	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.

Examples

```
{
  "uuid": "9f968e8a-ad9c-45ce-9beb-a55011856a99",
  "alias": 2,
  "matcher_id": "1c7e9725-8753-4b6c-a0b7-a71d7e915cb5",
  "color": {
    "values": [
      0.476731,
      0.381263,
      0.128475
    ]
  },
  "representations": {
    "RGB": [
      0.396114,
      0.479113,
      0.552308
    ]
  }
}
```

Response

Code	Body	application/json		
200 400	Properties (object)			
	data ColorDetectable, **required**	A detectable represents the numeric position in a colorspace. It is connected to a *Matcher*.		
		ColorDetectable		
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
		alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.	
		matcher_id UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	reference to the *Matcher* containing this Detectable	
		color TransformedColor, **required**	A color represented by a coordinate in the colorspace. The array indices of the `values` property	

Code	Body	application/json		
			match the order of the `colorspace.axes` property of currently used detection profile.	
			color	
			values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace
		representations ColorRepresentations, optional, read-only	Pre-calculcated visual representations of a color suitable for rendering	
			representations	
			RGB Array of number, minimum items: 3, maximum items: 3, **required**	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.
	errors Array of Error, **required**	Error[]		
		code String, optional	machine-readable unique error code	
		Mapping String, optional	a reference to the parameter that caused the error	
		Message String, optional	human-readable error description	
		May return the following error codes `LPLC.validation.collection_size_exceeded`		

1.2.2.12 Delete ColorDetectable)

`DELETE / sensor / detectable / {itemId}`

Deletes a single ColorDetectable.

Request

Path Variables
itemId String, **required**

Response

Code	
204	The empty response indicates success
	May return the following error codes `LCOL.samples.unavailable`

1.2.2.13 Modify ColorDetectable

`PUT / sensor / detectable / {itemId}`

Modifies a single ColorDetectable.

Request

Path Variables
itemId String, **required**

Body	application/json		

Properties (ColorDetectable)	**uuid** UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
	alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.	
	matcher_id UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	reference to the *Matcher* containing this Detectable	
	color TransformedColor, **required**	A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the `colorspace.axes` property of currently used detection profile.	
		color	
		values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace
	representations ColorRepresentations, optional, read-only	Pre-calculcated visual representations of a color suitable for rendering	
		representations	
		RGB Array of number, minimum items: 3, maximum items: 3, **required**	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.

Examples

```
{
  "uuid": "9f968e8a-ad9c-45ce-9beb-a55011856a99",
  "alias": 2,
  "matcher_id": "1c7e9725-8753-4b6c-a0b7-a71d7e915cb5",
  "color": {
    "values": [
      0.476731,
      0.381263,
      0.128475
    ]
  },
  "representations": {
    "RGB": [
      0.396114,
      0.479113,
      0.552308
    ]
  }
}
```

Response

Code	Body	application/json		
200 400 404	Properties (object)			
	data ColorDetectable, **required**	A detectable represents the numeric position in a colorspace. It is connected to a *Matcher*.		
		ColorDetectable		
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	

Code	Body	application/json		
		alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.	
		matcher_id UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	reference to the *Matcher* containing this Detectable	
		color TransformedColor, **required**	A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the `colorspace.axes` property of currently used detection profile.	
			color	
			values Array of number, minimum items: 3, maximum items: 3, required	Location in a colorspace
		representations ColorRepresentations, optional, read-only	Pre-calculcated visual representations of a color suitable for rendering	
			representations	
			RGB Array of number, minimum items: 3, maximum items: 3, required	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.
	errors Array of Error, **required**	Error[]		
		code String, optional	machine-readable unique error code	
		mapping String, optional	a reference to the parameter that caused the error	
		message String, optional	human-readable error description	
	May return the following error codes `LCOL.samples.unavailable`			

1.2.2.14 Get ColorDetectable

`GET / sensor / detectable / {itemId}`

Returns a single ColorDetectable.

Request

Path Variables
itemId String, **required**

Response

Code	Body	application/json		

200	Properties (object)			
	data ColorDetectable, **required**	A detectable represents the numeric position in a colorspace. It is connected to a *Matcher*.		
		ColorDetectable		
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
		alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.	
		matcher_id UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	reference to the *Matcher* containing this Detectable	
		color TransformedColor, **required**	A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the `colorspace.axes` property of currently used detection profile.	
			color	
			values Array of number, minimum items: 3, maximum items: 3, required	Location in a colorspace
		representations ColorRepresentations, optional, read-only	Pre-calculcated visual representations of a color suitable for rendering	
			representations	
			RGB Array of number, minimum items: 3, maximum items: 3, **required**	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.
	erros Array of Error, **required**	Error[]		
		code String, optional	machine-readable unique error code	
		mapping String, optional	a reference to the parameter that caused the error	
		message String, optional	human-readable error description	
	May return the following error codesn `LCOL.samples.unavailable`			

1.2.2.15 Switch current Detection Profile

`PUT / sensor / detection-profiles`

Only one of the available Detection Profiles is active at a given time. Write a new Detection Profile ID to the `current_profile_id` field in order to change the currently used profile.

Request

Body	application/json
Examples **Example**	`a014e415-0fec-4734-ac3f-30da0a5f3899`

Response

Code	Body	application/json	
200	Properties (object)		
	data CurrentDetectionProfileID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	The sensor can store multiple Detection Profiles, but it can only apply one at a time. The field `current_profile_id` contains the UUID of the Detection Profile that is currently used by the sensor for its operation. It allows to use the shortcut API endpoint `/api/sensor/detection-profiles/current` instead of specifying a Detection Profile by its UUID.	
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.2.16 Create DetectionProfiles

`POST / sensor / detection-profiles`

Create a new DetectionProfile.

All supported data attributes in the body of the request are optional.

Request

Body	application/json			
Properties (Detection-Profile)	**uuid** UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
	aliFas Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.		
	name String, **required**	Human-readable name of the Detection Profile		
	colorspace Colorspace, **required**	A colorspace describes the numeric conversion of colors under certain circumstances. Different standardized colorspaces are suitable for different detection tasks.		
		colorspace		
		name String, **required**		
		space_id ColorspaceID,**required**	Unique name of a colorspace	
		axes Array of ColorspaceAxis, minimum items: 3, maximum items: 3, **required**	ColorspaceAxis[]	
			id	Unique name

Body	**application/json**			
			String, **required**	
			label String, **required**	Human-readable name
			minimum Number, **required**	lowest expected value of a color along this axis under usual circumstances
			maximum Number, **required**	highest expected value of a color along this axis under usual circumstances
	non_matching_output WantedSwitchingOutputsState, **required**	This state of the Switching Outputs is applied, if the currently sample color does not belong to any of the stored *Matchers*.		
		non_matching_output		
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
		states Array of any of boolean or null, **required**	List of True/False/Null values describing the wanted states of the Switching Outputs	
	non_matching_hold_time HoldTime (number), maximum 3153600000, required	Minimum duration (in seconds) of the *non_matching_output* state being applied to the Switching Outputs of the sensor. This prolonging of a potential *non matching* event may be useful, if the processing period of a connected actor exceeds the sampling period of the sensor.		
	compensation_settings CompensationSettings, **required**	The compensation settings of a Detection Profile describe the configuration of internal sensor components related to the stabilization and compensation algorithms. These values can be determined by issuing a `POST` request against `/api/sensor/detection-profiles/current/autogain`. The result is a suitable set of compensation settings for this sensor under the current circumstances. The content of this data object is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).		
		compensation_settings		
	sampling_settings SamplingSettings, **required**	Sampling Settings describe all details of the sampling process. Its attributes may be queried and inspected (e.g. in order to retrieve the current sample rate). Most values stored within the Sampling Settings should not be modified directly. The related API endpoint `/api/sensor/detection-profiles/current/autogain` should be used instead. The only modifiable attribute within the Sampling Settings is the *averages* value. It is safe to change it, even though the default values calculated during an autogain operation should be optimal for most detection tasks.		
		sampling_settings		

Body	**application/json**			
		led_intensity Number, minimum: 0, maximum: 1, **required**	relative intensity of the internal emitter during the light phase	
		base_sample_rate SampleRate (number), minimum: 0.01, **required**	The base sample rate determines the duration of a sampling period. After each sampling period, the gathered data is processed and a new detection result is calculated (e.g. the most suitable *Matcher* for the given sample). This may affect the state of the Switching Outputs or trigger configured actions. Thus the base sample rate defines the maximum rate of changes for the Switching Outputs. See also the *effective sample rate*.	
		effective_sample_rate SampleRate (number), minimum: 0.01, **required**	The effective sample rate is the numeric product of the *base sample rate* and the number of *averages*. It determines the minimum duration that a target needs to be sampled in order to determine its visual appearance correctly. With the default value of *average* set to one, this value is equal to the base sample rate.	
		minimum_wanted_sample_rate SampleRate (number), minimum: 0.01, **required**	This informational value represents the sample rate that was requested during the most recent *Autogain* operation. The effective sample rate may deviate from the wanted sample rate, if the requested sample rate was not achievable due to limitations of the sensor (e.g. exceeding the supported sample rate) or due to the environment (e.g. not enough light, thus a slower amplification with higher gain was necessary).	
		sample_light_phase Boolean, **required**	defines if the sensor should periodically activate the internal emitter for sampling	
		sample_dark_phase Boolean, **required**	defines if the sensor should periodically	

Body	**application/json**			
			deactivate the internal emitter for sampling	
		averages AverageSampleCount (integer), minimum: 1, **required**	Number of previous samples to be averaged for every sampling result. A rolling averaging algorithm is applied to the samples.	
		amplification AmplificationLevel (integer), **required**	The amplification level specifies the internal configuration of an amplifier. This value is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).	
	white_reference Array of number, **required**	The White Reference attribute is used for indicating a custom color balancing. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference).`		
	normalization_constant Array of number, **required**	Normalization constants are related to the White Reference. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference`).		

Examples

```
{
  "name": "#0",
  "uuid": "2475df8d-85f0-4208-ba60-dce6cb282a96",
  "alias": 1,
  "non_matching_hold_time": 0,
  "colorspace": {
    "name": "L*a*b*",
    "axes": [
      {
        "id": "L",
        "label": "L*",
        "minimum": 0,
        "maximum": 100
      },
      {
        "id": "a",
        "label": "a*",
        "minimum": -500,
        "maximum": 500
      },
      {
        "id": "b",
        "label": "b*",
        "minimum": -200,
        "maximum": 200
      }
    ],
    "space_id": "Lab"
  },
  "compensation_settings": {
    "monitor_integration": {
      "control": 0.32499998807907104,
      "references": [
        0.7283520102500916,
        0.7442666888237,
        0.7066696286201477
      ]
    },
    "use_calibration_samples": true
  },
  "normalization_constant": [
    237.4935277662995,
    242.62655153828055,
    587.8264132734112
  ],
  "white_reference": [
    95.047,
    100,
    108.883
  ],
  "non_matching_output": {
    "uuid": "3f26aff4-8650-42a0-b319-51776c443fbc",
    "states": [
      true,
      true,
      true,
      true,
      true,
      true,
      true,
      true
    ]
  },
  "sampling_settings": {
    "led_intensity": 1,
    "amplification": 1,
    "sample_light_phase": true,
    "minimum_wanted_sample_rate": 1000,
    "averages": 1,
    "base_sample_rate": 1000,
    "sample_dark_phase": true,
    "effective_sample_rate": 1000
  }
}
```

Response

Code	Body	application/json			
200 400	Properties (object)				
	data Detection-Profile, **required**	A Detection Profile contains a complete set of sensor settings for a given detection task. Multiple profiles can be stored in order to switch easily between different detection task or for the incremental development of a refined profile. Some attributes of a Detection Profile expose in-			

Code	Body	application/json			
		ternal details of the sensor, that should be determined indirectly via other means. These attributes are described only superficially, since they should be handled *as is* without changing their value or structure.			
		DetectionProfile			
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
		alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.		
		name String, **required**	Human-readable name of the Detection Profile		
		colorspace Colorspace, **required**	A colorspace describes the numeric conversion of colors under certain circumstances. Different standardized colorspaces are suitable for different detection tasks.		
			colorspace		
			name String, required		
			space_id ColorspaceID, **required**	Unique name of a colorspace	
			axes Array of ColorspaceAxis, minimum items: 3, maximum items: 3, **required**	ColorspaceAxis[]	
				id String, **required**	Unique name
				label String, **required**	Human-readable name
				minimum Number, **required**	lowest expected value of a color along this axis under usual circumstances
				maximum Number, **required**	highest expected value of a color along this axis under usual circumstances
		non_matching_output WantedSwitchingOutputsState, **required**	This state of the Switching Outputs is applied, if the currently sample color does not belong to any of the stored *Matchers*.		
			non_matching_output		
			uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	

Code	Body	application/json			
			states Array of any of boolean or null, **required**	List of True/False/Null values describing the wanted states of the Switching Outputs	
		non_matching_hold_time HoldTime (number), maximum 3153600000, **required**	Minimum duration (in seconds) of the *non_matching_output* state being applied to the Switching Outputs of the sensor. This prolonging of a potential *non matching* event may be useful, if the processing period of a connected actor exceeds the sampling period of the sensor.		
		compensation_settings CompensationSettings, **required**	The compensation settings of a Detection Profile describe the configuration of internal sensor components related to the stabilization and compensation algorithms. These values can be determined by issuing a POST request against `/api/sensor/detection-profiles/current/autogain`. The result is a suitable set of compensation settings for this sensor under the current circumstances. The content of this data object is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).		
			compensation_settings		
		sampling_settings SamplingSettings, **required**	Sampling Settings describe all details of the sampling process. Its attributes may be queried and inspected (e.g. in order to retrieve the current sample rate). Most values stored within the Sampling Settings should not be modified directly. The related API endpoint `/api/sensor/detection-profiles/current/autogain` should be used instead. The only modifiable attribute within the Sampling Settings is the *averages* value. It is safe to change it, even though the default values calculated during an autogain operation should be optimal for most detection tasks.		
			sampling_settings		
			led_intensity Number, minimum: 0, maximum: 1, **required**	relative intensity of the internal emitter during the light phase	
			base_sample_rate SampleRate (number), minimum: 0.01, **required**	The base sample rate determines the duration of a sampling period. After each sampling period, the gathered data is processed and a new detection result is calculated (e.g. the most suitable *Matcher* for the given sample). This may affect the state of the Switching Outputs or trigger configured actions. Thus the base sample rate defines the maximum rate of changes for the Switching Outputs. See also the *effective sample rate*.	

Code	Body	application/json			
			effective_sample_rate SampleRate (number), minimum: 0.01, **required**	The effective sample rate is the numeric product of the *base sample rate* and the number of *averages*. It determines the minimum duration that a target needs to be sampled in order to determine its visual appearance correctly. With the default value of *average* set to one, this value is equal to the base sample rate.	
			minimum_wanted_sample_rate SampleRate (number), minimum: 0.01, **required**	This informational value represents the sample rate that was requested during the most recent *Autogain* operation. The effective sample rate may deviate from the wanted sample rate, if the requested sample rate was not achievable due to limitations of the sensor (e.g. exceeding the supported sample rate) or due to the environment (e.g. not enough light, thus a slower amplification with higher gain was necessary).	
			sample_light_phase Boolean, **required**	defines if the sensor should periodically activate the internal emitter for sampling	
			sample_dark_phase Boolean, **required**	defines if the sensor should periodically deactivate the internal emitter for sampling	
			averages AverageSampleCount (integer), minimum: 1, **required**	Number of previous samples to be averaged for every sampling result. A rolling averaging algorithm is applied to the samples.	
			amplification AmplificationLevel (integer), **required**	The amplification level specifies the internal configuration of an amplifier. This value is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).	
		white_reference Array of number, **required**	The White Reference attribute is used for indicating a custom color balancing. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference)`.		

Code	Body	application/json			
		normalization_constant Array of number, **required**	Normalization constants are related to the White Reference. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference`).		
	errors Array of Error, **required**	Error[]			
		code String, optional	machine-readable unique error code		
		mapping String, optional	a reference to the parameter that caused the error		
		message String, optional	human-readable error description		
		May return the following error codes `LPLC.validation.collection_size_exceeded`			

1.2.2.17 Remove multiple or all DetectionProfiles

`DELETE / sensor / detection-profiles`

Remove a selection of DetectionProfiles either based on a given filter argument (if supported for this collection) or remove all DetectionProfiles from the collection.

All delete requests result in an empty success response (204). This is even valid for a non-filtered `DELETE` request against an empty collection or for a filtered `DELETE` request against a collection without DetectionProfiles matching the filter.

Response

Code	Body	application/json	
200 204	Properties (object)		
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	data Object, **required**	data	
		current_profile_id CurrentDetectionProfileID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	The sensor can store multiple Detection Profiles, but it can only apply one at a time. The field `current_profile_id` contains the UUID of the Detection Profile that is currently used by the sensor for its operation. It allows to use the shortcut API endpoint `/api/sensor/detection-profiles/current` instead of specifying a Detection Profile by its UUID.

1.2.2.18 Retrieve DetectionProfiles

`GET / sensor / detection-profiles`

Retrieves a list of available DetectionProfiles

Response

Code	Body	application/json				
200	Properties (object)					
	data	data				

Code	Body	application/json				
	Object, **required**					
		detection-profiles Array of DetectionProfile, **required**	DetectionProfile[]			
			uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
			alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.		
			name String, **required**	Human-readable name of the Detection Profile		
			colorspace Colorspace, **required**	A colorspace describes the numeric conversion of colors under certain circumstances. Different standardized colorspaces are suitable for different detection tasks.		
				colorspace		
				name String, **required**		
				space_id ColorspaceID, **required**	Unique name of a colorspace	
				axes Array of ColorspaceAxis, minimum items: 3, maximum items: 3, **required**	ColorspaceAxis[]	
					id String, **required**	Unique name
					label String, **required**	Human-readable name
					minimum Number, **required**	lowest expected value of a color along this axis under usual circumstances
					maximum Number, **required**	highest expected value of a color along this axis under usual circumstances
			non_matching_output WantedSwitchingOutputsState, required	This state of the Switching Outputs is applied, if the currently sample color does not belong to any of the stored *Matchers*.		
				non_matching_output		
				uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec.	

Code	Body	application/json				
					X.667, and ISO/IEC 9834-8	
				states Array of any of boolean or null, required	List of True/False/Null values describing the wanted states of the Switching Outputs	
			non_matching_hold_time HoldTime (number), maximum 3153600000, **required**	Minimum duration (in seconds) of the *non_matching_output* state being applied to the Switching Outputs of the sensor. This prolonging of a potential *non matching* event may be useful, if the processing period of a connected actor exceeds the sampling period of the sensor.		
			compensation_settings CompensationSettings, **required**	The compensation settings of a Detection Profile describe the configuration of internal sensor components related to the stabilization and compensation algorithms. These values can be determined by issuing a POST request against `/api/sensor/detection-profiles/current/autogain`. The result is a suitable set of compensation settings for this sensor under the current circumstances. The content of this data object is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).		
				compensation_settings		
			sampling_settings SamplingSettings, **required**	Sampling Settings describe all details of the sampling process. Its attributes may be queried and inspected (e.g. in order to retrieve the current sample rate). Most values stored within the Sampling Settings should not be modified directly. The related API endpoint `/api/sensor/detection-profiles/current/autogain` should be used instead. The only modifiable attribute within the Sampling Settings is the *averages* value. It is safe to change it, even though the default values calculated during an autogain operation should be optimal for most detection tasks.		
				sampling_settings		
				led_intensity Number, minimum: 0, maximum: 1, **required**	relative intensity of the internal emitter	

Code	Body	application/json				
					during the light phase	
				base_sample_rate SampleRate (number), minimum: 0.01, **required**	The base sample rate determines the duration of a sampling period. After each sampling period, the gathered data is processed and a new detection result is calculated (e.g. the most suitable *Matcher* for the given sample). This may affect the state of the Switching Outputs or trigger configured actions. Thus the base sample rate defines the maximum rate of changes for the Switching Outputs. See also the *effective sample rate*.	
				effective_sample_rate SampleRate (number), minimum: 0.01, **required**	The effective sample rate is the numeric product of the *base sample rate* and the number of *averages*. It determines the minimum duration that a target needs to be sampled in order to determine its visual appearance correctly. With the default value of *average* set to one, this value is equal to the base sample rate.	
				minimum_wanted_sample_rate SampleRate (number), minimum: 0.01, **required**	This informational value represents the sample rate that was requested during the most recent *Autogain* operation. The effective sample rate may deviate from the	

Code	Body	application/json				
					wanted sample rate, if the requested sample rate was not achievable due to limitations of the sensor (e.g. exceeding the supported sample rate) or due to the environment (e.g. not enough light, thus a slower amplification with higher gain was necessary).	
				sample_light_phase Boolean, **required**	defines if the sensor should periodically activate the internal emitter for sampling	
				sample_dark_phase Boolean, **required**	defines if the sensor should periodically deactivate the internal emitter for sampling	
				averages AverageSampleCount (integer), minimum: 1, **required**	Number of previous samples to be averaged for every sampling result. A rolling averaging algorithm is applied to the samples.	
				amplification AmplificationLevel (integer), **required**	The amplification level specifies the internal configuration of an amplifier. This value is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).	
			white_reference Array of number, **required**	The White Reference attribute is used for indicating a custom color balancing. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference)`.		

Code	Body	application/json				
			normalization_constant Array of number, **required**	Normalization constants are related to the White Reference. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference`).		
		current_profile_id CurrentDetectionProfileID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	The sensor can store multiple Detection Profiles, but it can only apply one at a time. The field `current_profile_id` contains the UUID of the Detection Profile that is currently used by the sensor for its operation. It allows to use the shortcut API endpoint `/api/sensor/detection-profiles/current` instead of specifying a Detection Profile by its UUID.			
	errors Array of Error, **required**	Error[]				
		code String, optional	machine-readable unique error code			
		mapping String, optional	a reference to the parameter that caused the error			
		message String, optional	human-readable error description			

1.2.2.19 Delete DetectionProfile

`DELETE / sensor / detection-profiles / {itemId}`

Deletes a single DetectionProfile.

Request

Path Variables
itemId String, **required**

Response

Code	
204	The empty response indicates success
	May return the following error codes `LPLC.not_found.collection.item`

1.2.2.20 Modify DetectionProfile

`PUT / sensor / detection-profiles / {itemId}`

Modifies a single DetectionProfile.

Request

Path Variables
itemId

String, **required**

Body	**application/json**			
Properties (Detection-Profile)	**uuid** UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
	alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.		
	name String, **required**	Human-readable name of the Detection Profile		
	colorspace Colorspace, **required**	A colorspace describes the numeric conversion of colors under certain circumstances. Different standardized colorspaces are suitable for different detection tasks.		
		colorspace		
		name String, **required**		
		space_id ColorspaceID, **required**	Unique name of a colorspace	
		axes Array of ColorspaceAxis, minimum items: 3, maximum items: 3, **required**	ColorspaceAxis[]	
			id String, **required**	Unique name
			label String, **required**	Human-readable name
			minimum Number, **required**	lowest expected value of a color along this axis under usual circumstances
			maximum Number, **required**	highest expected value of a color along this axis under usual circumstances
	non_matching_output WantedSwitchingOutputsState, **required**	This state of the Switching Outputs is applied, if the currently sample color does not belong to any of the stored *Matchers*.		
		non_matching_output		
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
		states Array of any of boolean or null, required	List of True/False/Null values describing the wanted states of the Switching Outputs	
	non_matching_hold_time HoldTime (number), maximum 3153600000, **required**	Minimum duration (in seconds) of the *non_matching_output* state being applied to the Switching Outputs of the sensor. This prolonging of a potential *non matching* event may be useful, if the processing period of a connected actor exceeds the sampling period of the sensor.		
	compensation_settings CompensationSettings, **required**	The compensation settings of a Detection Profile describe the configuration of internal sensor components related to the stabilization and compensation algorithms.		

Body	**application/json**			
		These values can be determined by issuing a `POST` request against `/api/sensor/detection-profiles/current/autogain`. The result is a suitable set of compensation settings for this sensor under the current circumstances. The content of this data object is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).		
		compensation_settings		
	sampling_settings SamplingSettings, **required**	Sampling Settings describe all details of the sampling process. Its attributes may be queried and inspected (e.g. in order to retrieve the current sample rate). Most values stored within the Sampling Settings should not be modified directly. The related API endpoint `/api/sensor/detection-profiles/current/autogain` should be used instead. The only modifiable attribute within the Sampling Settings is the *averages* value. It is safe to change it, even though the default values calculated during an autogain operation should be optimal for most detection tasks.		
		sampling_settings		
		led_intensity Number, minimum: 0, maximum: 1, **required**	relative intensity of the internal emitter during the light phase	
		base_sample_rate SampleRate (number), minimum: 0.01, **required**	The base sample rate determines the duration of a sampling period. After each sampling period, the gathered data is processed and a new detection result is calculated (e.g. the most suitable *Matcher* for the given sample). This may affect the state of the Switching Outputs or trigger configured actions. Thus the base sample rate defines the maximum rate of changes for the Switching Outputs. See also the *effective sample rate*.	
		effective_sample_rate SampleRate (number), minimum: 0.01, **required**	The effective sample rate is the numeric product of the *base sample rate* and the number of *averages*. It determines the minimum duration that a target needs to be sampled in order to determine its visual appearance correctly. With the default value of *average* set to one, this value is equal to the base sample rate.	

Body	**application/json**			
		minimum_wanted_sample_rate SampleRate (number), minimum: 0.01, **required**	This informational value represents the sample rate that was requested during the most recent *Autogain* operation. The effective sample rate may deviate from the wanted sample rate, if the requested sample rate was not achievable due to limitations of the sensor (e.g. exceeding the supported sample rate) or due to the environment (e.g. not enough light, thus a slower amplification with higher gain was necessary).	
		sample_light_phase Boolean, **required**	defines if the sensor should periodically activate the internal emitter for sampling	
		sample_dark_phase Boolean, **required**	defines if the sensor should periodically deactivate the internal emitter for sampling	
		averages AverageSampleCount (integer), minimum: 1, **required**	Number of previous samples to be averaged for every sampling result. A rolling averaging algorithm is applied to the samples.	
		amplification AmplificationLevel (integer), **required**	The amplification level specifies the internal configuration of an amplifier. This value is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).	
	white_reference Array of number, **required**	The White Reference attribute is used for indicating a custom color balancing. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference`).		
	normalization_constant Array of number, **required**	Normalization constants are related to the White Reference. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference`).		

Examples

```
{
  "name": "#0",
  "uuid": "2475df8d-85f0-4208-ba60-dce6cb282a96",
  "alias": 1,
  "non_matching_hold_time": 0,
  "colorspace": {
    "name": "L*a*b*",
    "axes": [
      {
        "id": "L",
        "label": "L*",
        "minimum": 0,
        "maximum": 100
      },
      {
        "id": "a",
        "label": "a*",
        "minimum": -500,
        "maximum": 500
      },
      {
        "id": "b",
        "label": "b*",
        "minimum": -200,
        "maximum": 200
      }
    ],
    "space_id": "Lab"
  },
  "compensation_settings": {
    "monitor_integration": {
      "control": 0.32499998807907104,
      "references": [
        0.7283520102500916,
        0.7442666888237,
        0.7066696286201477
      ]
    },
    "use_calibration_samples": true
  },
  "normalization_constant": [
    237.4935277662995,
    242.62655153828055,
    587.8264132734112
  ],
  "white_reference": [
    95.047,
    100,
    108.883
  ],
  "non_matching_output": {
    "uuid": "3f26aff4-8650-42a0-b319-51776c443fbc",
    "states": [
      true,
      true,
      true,
      true,
      true,
      true,
      true,
      true
    ]
  },
  "sampling_settings": {
    "led_intensity": 1,
    "amplification": 1,
    "sample_light_phase": true,
    "minimum_wanted_sample_rate": 1000,
    "averages": 1,
    "base_sample_rate": 1000,
    "sample_dark_phase": true,
    "effective_sample_rate": 1000
  }
```

Response

Code	Body	application/json			
200 400 404	Properties (object)				
	data DetectionProfile, **required**	A Detection Profile contains a complete set of sensor settings for a given detection task. Multiple profiles can be stored in order to switch easily between different detection task or for the incremental development of a refined profile. Some attributes of a Detection Profile expose internal details of the sensor, that should be determined indirectly via other means. These attributes are described only superficially, since they should be handled *as is* without changing their value or structure.			
		DetectionProfile			
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
		alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.		
		name String, **required**	Human-readable name of the Detection Profile		
		colorspace Colorspace, **required**	A colorspace describes the numeric conversion of colors under certain circumstances. Different standardized colorspaces are suitable for different detection tasks.		
			colorspace		
			name String, **required**		
			space_id ColorspaceID, **required**	Unique name of a colorspace	

Code	Body	application/json			
			axes Array of ColorspaceAxis, minimum items: 3, maximum items: 3, **required**	ColorspaceAxis[]	
				id String, **required**	Unique name
				label String, **required**	Human-readable name
				minimum Number, **required**	lowest expected value of a color along this axis under usual circumstances
				maximum Number, **required**	highest expected value of a color along this axis under usual circumstances
		non_matching_output WantedSwitchingOutputsState, **required**	This state of the Switching Outputs is applied, if the currently sample color does not belong to any of the stored *Matchers*.		
			non_matching_output		
			uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
			states Array of any of boolean or null, **required**	List of True/False/Null values describing the wanted states of the Switching Outputs	
		non_matching_hold_time HoldTime (number), maximum 3153600000, **required**	Minimum duration (in seconds) of the *non_matching_output* state being applied to the Switching Outputs of the sensor. This prolonging of a potential *non matching* event may be useful, if the processing period of a connected actor exceeds the sampling period of the sensor.		
		compensation_settings CompensationSettings, **required**	The compensation settings of a Detection Profile describe the configuration of internal sensor components related to the stabilization and compensation algorithms. These values can be determined by issuing a `POST` re-		

Code	Body	application/json			
			quest against `/api/sensor/detection-profiles/current/autogain`. The result is a suitable set of compensation settings for this sensor under the current circumstances. The content of this data object is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).		
			compensation_settings		
		sampling_settings SamplingSettings, **required**	Sampling Settings describe all details of the sampling process. Its attributes may be queried and inspected (e.g. in order to retrieve the current sample rate). Most values stored within the Sampling Settings should not be modified directly. The related API endpoint `/api/sensor/detection-profiles/current/autogain` should be used instead. The only modifiable attribute within the Sampling Settings is the *averages* value. It is safe to change it, even though the default values calculated during an autogain operation should be optimal for most detection tasks.		
			sampling_settings		
			led_intensity Number, minimum: 0, maximum: 1, **required**	relative intensity of the internal emitter during the light phase	
			base_sample_rate SampleRate (number), minimum: 0.01, **required**	The base sample rate determines the duration of a sampling period. After each sampling period, the gathered data is processed and a new detection result is calculated (e.g. the most suitable *Matcher* for the given sample). This may affect the state of the Switching Outputs or trigger configured actions. Thus the base sample rate defines the maximum	

Code	Body	application/json			
				rate of changes for the Switching Outputs. See also the *effective sample rate*.	
			effective_sample_rate SampleRate (number), minimum: 0.01, **required**	The effective sample rate is the numeric product of the *base sample rate* and the number of *averages*. It determines the minimum duration that a target needs to be sampled in order to determine its visual appearance correctly. With the default value of *average* set to one, this value is equal to the base sample rate.	
			minimum_wanted_sample_rate SampleRate (number), minimum: 0.01, **required**	This informational value represents the sample rate that was requested during the most recent *Autogain* operation. The effective sample rate may deviate from the wanted sample rate, if the requested sample rate was not achievable due to limitations of the sensor (e.g. exceeding the supported sample rate) or due to the environment (e.g. not enough light, thus a slower amplification with higher gain was necessary).	
			sample_light_phase Boolean, **required**	defines if the sensor should periodically activate the internal emitter for sampling	
			sample_dark_phase Boolean, **required**	defines if the sensor should	

Code	Body	application/json			
				periodically deactivate the internal emitter for sampling	
			averages AverageSampleCount (integer), minimum: 1, **required**	Number of previous samples to be averaged for every sampling result. A rolling averaging algorithm is applied to the samples.	
			amplification AmplificationLevel (integer), **required**	The amplification level specifies the internal configuration of an amplifier. This value is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).	
		white_reference Array of number, **required**	The White Reference attribute is used for indicating a custom color balancing. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference) .`		
		normalization_constant Array of number, **required**	Normalization constants are related to the White Reference. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference`).		
	errors Array of Error, **required**	Error[]			
		code String, optional	machine-readable unique error code		
		mapping String, optional	a reference to the parameter that caused the error		
		message String, optional	human-readable error description		
		May return the following error codes `LPLC.not_found.collection.item`			

1.2.2.21 Get DetectionProfile

`GET / sensor / detection-profiles / {itemId}`

Returns a single DetectionProfile.

Request

Path Variables
itemId String, **required**

Response

Code	Body	application/json			
200	Properties (object)				
	data DetectionProfile, **required**	A Detection Profile contains a complete set of sensor settings for a given detection task. Multiple profiles can be stored in order to switch easily between different detection task or for the incremental development of a refined profile. Some attributes of a Detection Profile expose internal details of the sensor, that should be determined indirectly via other means. These attributes are described only superficially, since they should be handled *as is* without changing their value or structure.			
		DetectionProfile			
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
		alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.		
		name String, **required**	Human-readable name of the Detection Profile		
		colorspace Colorspace, **required**	A colorspace describes the numeric conversion of colors under certain circumstances. Different standardized colorspaces are suitable for different detection tasks.		
			colorspace		
			name String, **required**		
			space_id ColorspaceID, **required**	Unique name of a colorspace	
			axes Array of ColorspaceAxis, minimum items: 3, maximum items: 3, **required**	ColorspaceAxis[]	
				id String, **required**	Unique name
				label String, **required**	Human-readable name
				minimum Number, **required**	lowest expected value of a color

					along this axis under usual circumstances
				maximum Number, **required**	highest expected value of a color along this axis under usual circumstances
		non_matching_output WantedSwitchingOutputsState, **required**	This state of the Switching Outputs is applied, if the currently sample color does not belong to any of the stored *Matchers*.		
			non_matching_output		
			uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
			states Array of any of boolean or null, **required**	List of True/False/Null values describing the wanted states of the Switching Outputs	
		non_matching_hold_time HoldTime (number), maximum 3153600000, **required**	Minimum duration (in seconds) of the *non_matching_output* state being applied to the Switching Outputs of the sensor. This prolonging of a potential *non matching* event may be useful, if the processing period of a connected actor exceeds the sampling period of the sensor.		
		compensation_settings CompensationSettings, **required**	The compensation settings of a Detection Profile describe the configuration of internal sensor components related to the stabilization and compensation algorithms. These values can be determined by issuing a POST request against `/api/sensor/detection-profiles/current/autogain`. The result is a suitable set of compensation settings for this sensor under the current circumstances. The content of this data object is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).		
			compensation_settings		

		sampling_settings SamplingSettings, **required**	Sampling Settings describe all details of the sampling process. Its attributes may be queried and inspected (e.g. in order to retrieve the current sample rate). Most values stored within the Sampling Settings should not be modified directly. The related API endpoint `/api/sensor/detection-profiles/current/autogain` should be used instead. The only modifiable attribute within the Sampling Settings is the *averages* value. It is safe to change it, even though the default values calculated during an autogain operation should be optimal for most detection tasks.		
			sampling_settings		
			led_intensity Number, minimum: 0, maximum: 1, **required**	relative intensity of the internal emitter during the light phase	
			base_sample_rate SampleRate (number), minimum: 0.01, **required**	The base sample rate determines the duration of a sampling period. After each sampling period, the gathered data is processed and a new detection result is calculated (e.g. the most suitable *Matcher* for the given sample). This may affect the state of the Switching Outputs or trigger configured actions. Thus the base sample rate defines the maximum rate of changes for the Switching Outputs. See also the *effective sample rate*.	
			effective_sample_rate SampleRate (number), minimum: 0.01, **required**	The effective sample rate is the numeric product of the *base sample rate* and the number of *averages*. It determines the minimum	

				duration that a target needs to be sampled in order to determine its visual appearance correctly. With the default value of *average* set to one, this value is equal to the base sample rate.	
			minimum_wanted_sample_rate SampleRate (number), minimum: 0.01, **required**	This informational value represents the sample rate that was requested during the most recent *Autogain* operation. The effective sample rate may deviate from the wanted sample rate, if the requested sample rate was not achievable due to limitations of the sensor (e.g. exceeding the supported sample rate) or due to the environment (e.g. not enough light, thus a slower amplification with higher gain was necessary).	
			sample_light_phase Boolean, **required**	defines if the sensor should periodically activate the internal emitter for sampling	
			sample_dark_phase Boolean, **required**	defines if the sensor should periodically deactivate the internal emitter for sampling	
			averages AverageSampleCount (integer), minimum: 1, **required**	Number of previous samples to be averaged for every sampling result. A rolling averaging algorithm is applied to the samples.	
			amplification AmplificationLevel (integer), **required**	The amplification level specifies the inter-	

				nal configuration of an amplifier. This value is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).	
		white_reference Array of number, **required**	The White Reference attribute is used for indicating a custom color balancing. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference`).		
		normalization_constant Array of number, **required**	Normalization constants are related to the White Reference. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference`).		
	errors Array of Error, **required**	Error[]			
		code String, optional	machine-readable unique error code		
		mapping String, optional	a reference to the parameter that caused the error		
		message String, optional	human-readable error description		
		May return the following error codes `LPLC.not_found.collection.item`			

1.2.2.22 Start Autogain Procedure

`POST / sensor / detection-profiles / {itemId} / autogain`

Execute the autogain procedure in order to determine suitable sampling properties for the current environment. The resulting sampling setup is applied automatically. These new settings are in effect as soon as the response is sent.
The autogain procedure initiates a dynamic recalibration of the internal emitter and all compensation processes. It results in quick changes or flashing of the internal emitter (if enabled). The operation is usually finished within 15 seconds. The response is sent after all related activities are completed.
Later requests for a sample will return values based on the adjusted sampling settings.

Request

Path Variables	
itemId String, **required**	
Body	**application/json**

Properties (AutogainSettings)	
level Number, default: 0.8, minimum: 0.01, maximum: 1, optional	Target value for the auto-gain procedure
minimum_sample_rate SampleRate (number), minimum: 0.02, optional	Desired sample rate (the default is the current sample rate)
enable_internal_emitter Boolean, default: true, optional	controls the power of the internal light source
enable_ambient_light_compensation Boolean, default: true, optional	Control the ambient light compensation procedure. This setting is only relevant if `enable_internal_emitter` is set to true. The ambient light compensation leads to a pulsed usage of the internal light emitter. Samples are collected for alternating light and dark phases. This allows to calculate a color sample of the target excluding any optical interference from external light sources. You should not disable ambient light compensation unless the optical path is perfectly isolated. Otherwise external light will inevitably interfere with the color sampling.
averages AverageSampleCount (integer), minimum: 1, optional	Anzahl zu mittelnder vorheriger Stichproben für jedes Stichprobenergebnis.

Examples

```
{
  "level": 0.7,
  "minimum_sample_rate": 1500,
  "enable_internal_emitter": true,
  "enable_ambient_light_compensation": true
}
```

Response

Code	Body	application/json	
200 400	Properties (object)		
	data **sampling_settings** SamplingSettings, required	Sampling Settings describe all details of the sampling process. Its attributes may be queried and inspected (e.g. in order to retrieve the current sample rate). Most values stored within the Sampling Settings should not be modified directly. The related API endpoint `/api/sensor/detection-profiles/current/autogain` should be used instead. The only modifiable attribute within the Sampling Settings is the *averages* value. It is safe to change it, even though the default values calculated during an autogain operation should be optimal for most detection tasks.	
		SamplingSettings	
		led_intensity Number, minimum: 0, maximum: 1, **required**	relative intensity of the internal emitter during the light phase
		base_sample_rate SampleRate (number), minimum: 0.01, **required**	The base sample rate determines the duration of a sampling period. After each sampling period, the gathered data is processed and a new detection result is calculated (e.g. the most suitable *Matcher* for the given sample). This may affect the state of the Switching Outputs or trigger configured actions. Thus the base sample rate defines the maximum rate of changes for the Switching Outputs. See also the *effective sample rate*.
		effective_sample_rate SampleRate (number), minimum: 0.01, required	The effective sample rate is the numeric product of the *base sample rate* and the number of *averages*.

Code	Body	application/json	
			It determines the minimum duration that a target needs to be sampled in order to determine its visual appearance correctly. With the default value of *average* set to one, this value is equal to the base sample rate.
		minimum_wanted_sample_rate SampleRate (number), minimum: 0.01, **required**	This informational value represents the sample rate that was requested during the most recent *Autogain* operation. The effective sample rate may deviate from the wanted sample rate, if the requested sample rate was not achievable due to limitations of the sensor (e.g. exceeding the supported sample rate) or due to the environment (e.g. not enough light, thus a slower amplification with higher gain was necessary).
		sample_light_phase Boolean, **required**	defines if the sensor should periodically activate the internal emitter for sampling
		sample_dark_phase Boolean, **required**	defines if the sensor should periodically deactivate the internal emitter for sampling
		averages AverageSampleCount (integer), minimum: 1, **required**	Number of previous samples to be averaged for every sampling result. A rolling averaging algorithm is applied to the samples.
		amplification AmplificationLevel (integer), **required**	The amplification level specifies the internal configuration of an amplifier. This value is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).
	compensation_settings CompensationSettings, **required**	The compensation settings of a Detection Profile describe the configuration of internal sensor components related to the stabilization and compensation algorithms. These values can be determined by issuing a POST request against `/api/sensor/detection-profiles/current/autogain`. The result is a suitable set of compensation settings for this sensor under the current circumstances. The content of this data object is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).	
		compensation_settings	
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

	May return the following error codes
	`LCOL.autogain`
	`LCOL.autogain.invalid_target_level`
	`LCOL.autogain.invalid_sample_rate`
	`LPLC.validation.boolean`		

1.2.2.23 Query custom White Reference

`GET / sensor / detection-profiles / {itemId} / white-reference`

Verify the existence of a custom White Reference. A successful response (HTTP Status 200) indicates that a custom White Reference is in use. The *not found* response (HTTP Status 404) indicates that the factory default White Reference is used instead.

The detailed content of the response is not relevant. Instead the related `normalization_constants` field of the Detection Profile is adjusted based on the current White Reference.

Request

Path Variables
itemId String, **required**

Response

Code	**A custom White Reference is in use.**		
200 404	Body	application/json	
	Properties (object)		
	data Array of number, **required**	The White Reference attribute is used for indicating a custom color balancing. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference)`.	
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.2.24 Sample a custom White Reference

`POST / sensor / detection-profiles / {itemId} / white-reference`

Apply a custom White Reference for the color handling of the sensor. The currently sampled color is used for calculating the White Reference. You should pick a neutral white target for this action.

Please note that the change of the White Reference is not in effect immediately. Thus you should wait for three seconds, before sampling new values.

Request

Path Variables
itemId String, **required**

Response

Code	**Body**	**application/json**	
200 406	Properties (object)		
	data Array of number, **required**	The White Reference attribute is used for indicating a custom color balancing. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference)`.	
	errors	Error[]	

	Array of Error, **required**		
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes `LCOL.white_reference.too_dark`		

1.2.2.25 Lese Aktionsauslöser aus

`GET / sensor / action-triggers`

The sensor can be programmed to react on specific external or internal events. The available actions can be either triggered via trigger input lines or via API requests. This allows customized behaviour, e.g. teaching colors via an external button.

Multiple *ActionTrigger* items can be created. Each *ActionTrigger* assigns one or more actions to a specific event (see `trigger_sources` in `/api/sensor/capabilities`). Multiple *ActionTriggers* may refer to the same event (see *order of execution* below for details).

The actions assigned to an *ActionTrigger* are evaluated separately. Thus it is possible to specify the same action (even with the same parameters) multiple times. The list of actions for an *ActionTrigger* may be empty.

The actions within a single *ActionTrigger* are executed successively according to the order of the list items. The order of execution among multiple *ActionTrigger* items is undefined.

Trigger Events describing a state *change* (e.g. `trigger_0_edge_rising`) are emitted only once at the moment of the state change. Thus attached actions are executed only once for every state change.

Trigger Events describing a *state* (e.g. `trigger_0_level_low`) are emitted continuously as long as the state is active. The actions of an *ActionTrigger* attached to such a Trigger Event are executed periodically. After every execution of such an *ActionTrigger* further executions are skipped for a period of one second or until the next state change (whichever comes first). The only exception for this periodically executed actions is the *enable_switching_output* action. If this action is attached to a *state*, then it is re-evaluated whenever the hold time of the currently detected macher expires (i.e. for matchers with hold time zero: in every sample period).

See *actions* for a list of supported actions.

Request

Query Parameters	
event UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	Filter ActionTriggers by the given event name (e.g. *trigger_0_edge_rising*).

Response

Code	**Body**	**application/json**			
200	Properties (object)				
	data Object, **required**	data			
		action-triggers Array of ActionTrigger, **required**	ActionTrigger[]		
			uuid UUID (string), pattern: `^[a-f0-9-`	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	

			`]+$`, **required**, read-only		
			event TriggerEventName (string), **required**	Any of the event names provided by `/api/sensor/capabilities` (attribute `trigger_sources`) is allowed.	
			actions Array of Action, **required**	List of actions to be executed after the given event.	
				Action[]	
				name String, **required**	Unique name of the action
				arguments Object, **required**	arguments
		erros Array of Error, **required**	Error[]		
			code String, optional	machine-readable unique error code	
			mapping String, optional	a reference to the parameter that caused the error	
			message String, optional	human-readable error description	

1.2.2.26 Remove multiple or all Action Triggers

`DELETE / sensor / action-triggers`

Remove a selection of ActionTriggers either based on a given filter argument (if supported for this collection) or remove all ActionTriggers from the collection.

All delete requests result in an empty success response (204). This is even valid for a non-filtered `DELETE` request against an empty collection or for a filtered `DELETE` request against a collection without ActionTriggers matching the filter.

Request

Query Parameters	
event UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	Delete all ActionTriggers assigned to a given event name (e.g. *trigger_0_edge_rising*).

Response

Code	
204	The empty response indicates success

1.2.2.27 Create ActionTriggers

`POST / sensor / action-triggers`

Create a new ActionTrigger.

All supported data attributes in the body of the request are optional.

Request

Body	**application/json**		
Properties (Action Trigger)	**uuid** UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
	event TriggerEventName (string), **required**	Any of the event names provided by `/api/sensor/capabilities` (attribute `trigger_sources`) is allowed.	
	actions Array of Action, **required**	List of actions to be executed after the given event.	
		Action[]	

		name String, **required**	Unique name of the action
		arguments Object, **required**	arguments

Examples

```
{
  "uuid": "3f26aff4-8650-42a0-b319-51776c443fbc",
  "event": "trigger_0_edge_falling",
  "actions": [
    {
      "name": "enable_switching_output",
      "arguments": {}
    }
  ]
}
```

Response:

Code	Body	application/json		
200 400	Properties (object)			
	data ActionTrigger, **required**	An Action Trigger assigns a given set of actions with an event. At the end of each sample period, all events are evaluated. All corresponding actions are executed afterwards.		
		ActionTrigger		
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
		event TriggerEventName (string), **required**	Any of the event names provided by `/api/sensor/capabilities` (attribute `trigger_sources`) is allowed.	
		actions Array of Action, **required**	List of actions to be executed after the given event.	
			Action[]	
			name String, **required**	Unique name of the action
			arguments Object, **required**	arguments
	erros Array of Error, **required**	Error[]		
		code String, optional	machine-readable unique error code	
		mapping String, optional	a reference to the parameter that caused the error	
		message String, optional	human-readable error description	
	May return the following error codes `LPLC.validation.collection_size_exceeded`			

1.2.2.28 Delete ActionTrigger

`DELETE / sensor / action-triggers / {itemId}`

Deletes a single ActionTrigger.

Request

Path Variables

itemId String, **required**

Response

Code	
204	The empty response indicates success
	May return the following error codes `LPLC.not_found.collection.item`

1.2.2.29 Modify ActionTriger

`PUT / sensor / action-triggers / {itemId}`

Modifies a single ActionTrigger.

Request

Path Variables		
itemId String, **required**		
Body	application/json	
Properties (Action Trigger)		
uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
event TriggerEventName (string), **required**	Any of the event names provided by `/api/sensor/capabilities` (attribute `trigger_sources`) is allowed.	
actions Array of Action, **required**	List of actions to be executed after the given event.	
	Action[]	
	name String, **required**	Unique name of the action
	arguments Object, **required**	arguments

Examples

```
{
  "uuid": "3f26aff4-8650-42a0-b319-51776c443fbc",
  "event": "trigger_0_edge_falling",
  "actions": [
    {
      "name": "enable_switching_output",
      "arguments": {}
    }
  ]
}
```

Response

Code	**Body**	**application/json**		
200 400 404	Properties (object)			
	data ActionTrigger, **required**	An Action Trigger assigns a given set of actions with an event. At the end of each sample period, all events are evaluated. All corresponding actions are executed afterwards.		
		ActionTrigger		
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	

		event TriggerEventName (string), **required**	Any of the event names provided by `/api/sensor/capabilities` (attribute `trigger_sources`) is allowed.	
		actions Array of Action, **required**	List of actions to be executed after the given event.	
			Action[]	
			name String, **required**	Unique name of the action
			arguments Object, **required**	arguments
	errors Array of Error, **required**	Error[]		
		code String, optional	machine-readable unique error code	
		mapping String, optional	a reference to the parameter that caused the error	
		message String, optional	human-readable error description	
	May return the following error codes `LPLC.not_found.collection.item`			

1.2.2.30 Get ActionTrigger

`GET / sensor / action-triggers / {itemId}`

Returns a single ActionTrigger

Request

Path Variables
itemId String, **required**

Response

Code	**Body**	**application/json**		
200	Properties (object)			
	data ActionTrigger, **required**	An Action Trigger assigns a given set of actions with an event. At the end of each sample period, all events are evaluated. All corresponding actions are executed afterwards.		
		ActionTrigger		
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
		event TriggerEventName (string), **required**	Any of the event names provided by `/api/sensor/capabilities` (attribute `trigger_sources`) is allowed.	
		actions Array of Action, **required**	List of actions to be executed after the given event.	
			Action[]	
			name String, **required**	Unique name of the action
			arguments Object, **required**	arguments
	erros Array of Error, **required**	Error[]		
		code String, optional	machine-readable unique error code	
		mapping String, optional	a reference to the parameter that caused the error	
		message String, optional	human-readable error description	

		May return the following error codes `LPLC.not_found.collection.item`	

1.2.2.31 Get sensoric Capabilities

`GET / sensor / capabilities`

Response

Code	Body	application/json			
200	Properties (object)				
	data SensorCapabilities, **required**	Provide access to the sensoric details supported by this device (e.g. colorspaces, input and output lines, ...).			
		SensorCapabilities			
		maximum_sample_rate Integer, **required**	the maximum sample rate the sensor supports		
		tolerances Array of ColorTolerance (union), **required**	List of tolerance specifications supported by the sensor		
			InfiniteColorTolerance		
			limits Object, **required**	limits	
			shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities .`	
			SphereColorTolerance		
			limits Object, **required**	limits **radius** Numer, **required**	
			shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities.`	
			CylinderColorTolerance		
			limits Object, **required**	limits **radius** Number, required **half_height** Number, required	
			shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities.`	
			BoxColorTolerance		
			limits Object, **required**	limits **half_edges** Array of number, minimum items: 3, maximum items: 3, **required**	
			shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities .`	

Code	Body	application/json			
		output_drivers Array of SwitchingOutputDriver (string), **required**	List of supported electrical output drivers		
		trigger_sources Array of TriggerSource, **required**	Beinhaltet die Liste verfügbarer Auslösequellen mit ihrem dazugehörigen Auslösefall. Auslösefälle können zum Ausführen bestimmter Aktionen automatisiert werden.		
			TriggerSource[]		
			name String, **required**	Name of the trigger input	
			events Array of TriggerEvent, **required**	TriggerEvent []	
				name TriggerEventName (string), **required**	
		output_pin_count Integer, **required**	Number of available switching output lines		
		Actions Array of Action, **required**, Deprecated	Deprecated: use `/api/actions` instead		
			Action[]		
			name String, **required**	Unique name of the action	
			arguments Object, **required**	arguments	
		colorspaces Array of Colorspace, **required**	List of supported colorspaces.		
			Colorspace[]		
			name String, **required**		
			space_id ColorspaceID, **required**	Unique name of a colorspace	
			axes Array of ColorspaceAxis, minimum items: 3, maximum items: 3, **required**	ColorspaceAxis[]	
				id String, **required**	Unique name
				label String, **required**	Human-readable name
				minimum Number, **required**	lowest expected value of a color along this axis under usual circumstances
				maximum Number, **required**	highest expected value of a color along this axis under usual circumstances
		colorspace_tolerance_maps Array of ColorspaceToleranceMap, **required**	The evaluation of tolerances against positions of detectables		

Code	Body	application/json			
			depends on the currently configured colorspace. For example the tolerance attribute "half_height" refers to the brightness-related axis of a colorspace (e.g. "L*" for the "Lab*" colorspace) and is used for the height of the cylindrical tolerance shape and the first edge of the box tolerance shape. The hue-related attributes (e.g. "a" and "b" for the "Lab*" colorspace) are used for the "radius" of a cylinder tolerance shape and the second and third edges of the box tolerance shape. The *colorspace_tolerance_maps* define these relationships between colorspaces and tolerances.		
			ColorspaceToleranceMap[]		
			colorspace_id ColorspaceID (string), **required**	Unique name of a colorspace	
			tolerance_shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities .`	
			limits_axes_map Object, **required**	limits_axes_map	
				half_height Array of string, optional	
				half_edges Array of string, optional	
				radius Array of string, optional	
		settings_categories Array of string, **required**	List of categories that can be selected during import to control which settings should be applied. See the documentation for the POST request to `/api/seetings.		
		maximum_detectables_count Integer, **required**	Maximum number of color positions (*Detectable*) to be stored in a detection profile.		
		maximum_matchers_count Integer, **required**	Maximum number of detection results (*Matcher*) be stored in a detection profile.		
	errors Array of Error, **required**	Error[]			
		code String, optional	machine-readable unique error code		
		mapping String, optional	a reference to the parameter that caused the error		

Code	Body	application/json			
		message String, optional	human-readable error description		

1.2.2.32 Retrieve Colorspaces

`GET / sensor / colorspaces`

Retrieves a list of available Colorspaces.

Response

Code	Body	application/json			
200	Properties (object)				
	data Object, **required**	data			
		colorspaces Array of Colorspace, **required**	Colorspace[]		
			name String, **required**		
			space_id ColorspaceID, **required**	Unique name of a colorspace	
			axes Array of ColorspaceAxis, minimum items: 3, maximum items: 3, **required**	ColorspaceAxis[]	
				id String, **required**	Unique name
				label String, **required**	Human-readable name
				minimum Number, **required**	lowest expected value of a color along this axis under usual circumstances
				maximum Number, **required**	highest expected value of a color along this axis under usual circumstances
	errors Array of Error, **required**	Error[]			
		code String, optional	machine-readable unique error code		
		mapping String, optional	a reference to the parameter that caused the error		
		message String, optional	human-readable error description		

1.2.2.33 Get Colorspace

`GET / sensor / colorspaces / {space_id}`

Returns a single Colorspace.

Request

Path Variables
space_id String, **required**

Response

Code	Body	application/json		
200	Properties (object)			
	data Colorspace, **required**	A colorspace describes the numeric conversion of colors under certain circumstances. Different standardized colorspaces are suitable for different detection tasks.		
		colorspace		
		name String, **required**		
		space_id ColorspaceID, **required**	Unique name of a colorspace	
		axes Array of ColorspaceAxis, minimum items: 3, maximum items: 3, **required**	ColorspaceAxis[]	
			id String, **required**	Unique name
			label String, **required**	Human-readable name
			minimum Number, **required**	lowest expected value of a color along this axis under usual circumstances
			maximum Number, **required**	highest expected value of a color along this axis under usual circumstances
	errors Array of Error, **required**	Error[]		
		code String, optional	machine-readable unique error code	
		mapping String, optional	a reference to the parameter that caused the error	
		message String, optional	human-readable error description	
		May return the following error codes `LPLC.not_found.collection.item`		

1.2.3 Settings

Management of all device settings

1.2.3.1 Export Settings

`GET / settings`

Export the complete configuration of the device.

Response

Code		
200	Textual representation of the complete device configuration. This configuration export can be uploaded to the same or another sensor without modifications. The configuration data is encoded as Base64. The Base64 encoding is supposed to signal, that the configuration data dump is not meant to be manipulated or inspected automatically. You may not rely on a specific internal structure as it may change over time without further notice.	
	Body	**text/plain**

1.2.3.2 Upload Settings

`POST / settings`

Replace the device configuration with the one being uploaded.

If you only want to partially import the settings you can do so by specifying one or more import categories. If you don't specify at least one import category the default is to import all of them.

Import Category Key	**Will import**
import_category_access	Users, roles and permissions
import_category_firmware	Firmware settings (like the branch, but not the firmware itself)
import_category_keybad	Keypad settings
import_category_network	Network configuration
import_category_outputs	Output driver
import_category_sensor	Matchers (color groups), detectables (colors), colorspace, sample configuration
import_category_system	System settings (like timezone and hostname)

A machine-readable list of import categories is returned by the `/api/sensor/capabilities` endpoint under the `settings_categories` key.

Settings exported from older firmwares will automatically be migrated to the new format required by the sensor. In case a migration fails the response will contain the `LPLC.migration.execution_failed` error code. Settings from more recent firmwares than the one used on the sensor MAY fail on import, if the settings format is no longer compatible. In that case the response will contain the `LPLC.migration.future_version` error code.

Request

Body	**multipart/form-data**
Properties (object)	
Settings_file File, **required**	The settings file containing a Base64 encoded configuration dump. See GET request for `/api/settings`.
/^**import_category_**.+/ Any, optional	Import only a specific subset of the configuration.

Response

Code	
204	The empty response indicates success.

400	**May return the following error codes**
	`LPLC.format.encoding.utf8`
	`LPLC.format.malformed.base64`
	`LPLC.validation.missing_input`
	`LPLC.format.malformed.json`

	LPLC.format.malformed.json.not_dict LPLC.migration.future_version LPLC.migration.execution_failed

1.2.3.3 Import Settings

PUT / settings

Replace the complete device configuration with the uploaded configuration dump.

The content to be uploaded can be retrieved via a GET request on /api/settings. This Base64 encoded configuration dump is expected as the request body.

Settings exported from older firmwares will automatically be migrated to the new format required by the sensor. In case a migration fails the response will contain the LPLC.migration.execution_failed error code. Settings from more recent firmwares than the one used on the sensor MAY fail on import, if the settings format is no longer compatible. In that case the response will contain the LPLC.migration.future_version error code.

Request

Body	**text/plain**

Response

Code	
204	The empty response indicates success.
400	**May return the following error codes** LPLC.format.encoding.utf8 LPLC.format.malformed.base64 LPLC.format.malformed.json LPLC.format.malformed.json.not_dict LPLC.migration.future_version LPLC.migration.execution_failed

1.2.3.4 Reset Settings

DELETE / settings

Reset the device configuration to the factory defaults.

Response

Code	
204	The empty response indicates success

1.2.4 System

Manage the device's system settings:

1.2.4.1 Request System Settings

GET / system

Response

Code	**Body**	**application/json**	
200	Properties (object)		
	data SystemSettings, **required**	SystemSettings	
		hostname Hostname, pattern: ^(?:[a-zA-Z0-9](?:[a-zA-Z0-9\-]*[a-zA-Z0-9])?\.)*[a-zA-Z0-9](?:[a-zA-Z0-9\-]*[a-zA-Z0-9])?$, optional	Human-readable name identifying the device in the network
		uptime any of number or null, optional, read-only	The current system uptime in seconds. Though highly unlikely can be

			nil in case the system reported an invalid value.
	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.4.2 Modify System Settings

`PUT / system`

Request

Body	**application/json**
Properties (SystemSettings)	
hostname Hostname, pattern: `^(?:[a-zA-Z0-9](?:[a-zA-Z0-9\-]*[a-zA-Z0-9])?\.)*[a-zA-Z0-9](?:[a-zA-Z0-9\-]*[a-zA-Z0-9])?$`, optional	Human-readable name identifying the device in the network
uptime any of number or null, optional, read-only	The current system uptime in seconds. Though highly unlikely can be nil in case the system reported an invalid value.

Examples

```
{
  "hostname": "cfo-7454232361"
}
```

Response

Code	**Body**	**application/json**	
200 400 500	Properties (object)		
	data SystemSettings, **required**	SystemSettings	
		hostname Hostname, pattern: `^(?:[a-zA-Z0-9](?:[a-zA-Z0-9\-]*[a-zA-Z0-9])?\.)*[a-zA-Z0-9](?:[a-zA-Z0-9\-]*[a-zA-Z0-9])?$`, optional	Human-readable name identifying the device in the network
		uptime any of number or null, optional, read-only	The current system uptime in seconds. Though highly unlikely can be nil in case the system reported an invalid value.
	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes `LPLC.system.action_failed`		

1.2.4.3 Reset to Factory Firmware and Settings

`POST / system / factory-reset`

Reset the sensor's firmware to its factory default and initiate a reboot. After completion the sensor will use its original ("recovery") firmware and all settings are reset to their defaults. The recovery firmware can be upgraded via "/firmware/upgrade-from-current".

In case you **only** want to reset the settings it is sufficient to send a `DELETE` request to the `/api/settings` endpoint.

Response

Code	
204 500	The empty response indicates success
	May return the following error codes `LPLC.system.action_failed`

1.2.4.4 Initiate Reboot

`POST / system / reboot`

Reboots the device.

The software-triggered reboot is the more polite method to shutdown the sensor compared to unplugging the power supply. However the latter is safe as well.

Response

Code	
204 500	The empty response indicates success
	May return the following error codes `LPLC.system.action_failed`

1.2.4.5 Get time settings

`GET / system / time`

Response

Code	**Body**	**application/json**	
200	Properties (object)		
	data SystemTimeSettings, **required**	SystemTimeSettings	
		now Timestamp (string), optional	current time from the perspective of the sensor
		timezone String, optional	currently configured timezone
		ntp_servers Array of string, optional	one or more network time servers
		default_ntp_servers Array of string, optional, reaed-only	preconfigured network time servers
	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.4.6 Change time settings

`PUT / system / time`

Request

Body	**application/json**
Properties (SystemTimeSettings)	

now Timestamp (string), optional	current time from the perspective of the sensor
timezone String, optional	currently configured timezone
ntp_servers Array of string, optional	one or more network time servers
default_ntp_servers Array of string, optional, reaed-only	preconfigured network time servers

Examples

```
{
  "now": "2018-01-24T15:45:15.694004+01:00",
  "timezone": "Europe/Berlin",
  "ntp_servers": [
    "pool.ntp.org"
  ],
  "default_ntp_servers": [
    "pool.ntp.org"
  ]
}
```

Response

Code	Body	application/json	
200 400 500	Properties (object)		
	data SystemTimeSettings, **required**	SystemTimeSettings	
		now Timestamp (string), optional	current time from the perspective of the sensor
		timezone String, optional	currently configured timezone
		ntp_servers Array of string, optional	one or more network time servers
		default_ntp_servers Array of string, optional, reaed-only	preconfigured network time servers
	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

	May return the following error codes
	`LPLC.system.action_failed`
	`LPLC.validation.readonly`
	`LPLC.validation.readonly`		

1.2.4.7 Retrieve supported Timezones.

`GET / system / time / zones`

The device contains knowledge about an exhaustive list of officially standardized timezones. The sensor should be configured either with a local timezone or with UTC.

Response

Code	Body	application/json	
200	Properties (object)		
	data Object, **required**	Data	
		Timezone_names	List of timezones supported by the device

		Array of string, **required**	
	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.5 Network

The services of the sensor device are accessible via network connections. The network interfaces of the device can be configured for all standard compliant network setups.

1.2.5.1 Reset Network Settings

`DELETE / network`

Reset networking settings to factory defaults

Response

Code	
204	The empty response indicates success.

1.2.5.2 Retrieve status and configuration of all network interfaces

`GET / network / interfaces`

Returns a list of network interfaces.

Each network interface has a MAC-Address (`hardware_address`), a unique interface name (`iface`) and indicates the current physical connection status (`has_link`).

Both IPv4 (`ipv4`) and IPv6 (`ipv6`) are supported, both with their current configuration (`address_configurations`) and actual interface addresses (`current_addresses`).

Interface addresses are in CIDR notation starting with the interface address, followed by slash and ending with a decimal number representing the subnet mask (IPv4) or prefix length (IPv6).

The collection itself is read-only as no new interfaces can be added. You can change the individual interface configuration with PUT requests to a specific interface resource.

Response

Code	**Body**	**application/json**			
200	Properties (object)				
	data Array of NetworkInterfaceAddressConfigurationState, **required**	NetworkInterfaceAddressConfigurationState[]			
		ipv4 NetworkInterfaceAddressFamilyStateIPv4, optional	IPv4 Network address configuration		
			ipv4		
			address_configurations Array of any of NetworkAddress ConfigurationIPv4Static or NetworkAddressConfigurationIPv4DHCP, optional	NetworkAddressConfigurationIPv4Static[]	
				method	Configuration

Code	Body	application/json			
				string, one of [`static`, `dhcp`], **required**	method used for the address.
				address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation
				gateway NetworkAddressIPv4 (string), optional	default gateway for outgoing traffic
				NetworkAddressConfigurationIPv4DHCP[]	
				method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.
		ipv6 NetworkInterfaceAddressFamilyStateIPv6, optional	IPv6 Network address configuration		
			ipv6		
			address_configurations Array of any of NetworkAddressConfigurationIPv6Static, NetworkAddressConfigurationIPv6DHCP or NetworkAddressConfigurationIPv6Auto, optional	NetworkAddressConfigurationIPv6Static[]	
				method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.
				address NetworkInterfaceAddressIPv6 (string), **required**	IPv6 network address in CIDR notation
				gateway NetworkAddressIPv6 (string), optional	default gateway for outgoing traffic
				NetworkAddressConfigurationIPv6DHCP[]	
				method	Configuration method

Code	Body	application/json			
				string, one of `[static, dhcp, auto]`, required	used for the address.
				NetworkAddressConfigurationIPv6Auto[]	
				method string, one of `[static, dhcp, auto]`, required	Configuration method used for the address.
	errors Array of Error, **required**	Error []			
		code String, optional	machine-readable unique error code		
		mapping String, optional	a reference to the parameter that caused the error		
		message String, optional	human-readable error description		

1.2.5.3 Retrieve status and configuration of a single network interface

`GET / network / interfaces /{name}`

Returns information, current status, address configuration, and current addresses for a single interface.

Request

Path Variables
name String, **required**

Response

Code	Body	application/json			
200	Properties (object)				
	data Array of NetworkInterfaceAddressConfigurationState, **required**	NetworkInterfaceAddressConfigurationState[]			
		ipv4 NetworkInterfaceAddressFamilyStateIPv4, optional	IPv4 Network address configuration		
			ipv4		
			address_configurations Array of any of NetworkAddressConfigurationIPv4Static or NetworkAddressConfigurationIPv4DHCP, optional	NetworkAddressConfigurationIPv4Static[]	
				method string, one of `[static, dhcp]`, **required**	Configuration method used for the address.
				address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation

				gateway NetworkAddressIPv4 (string), optional	default gateway for outgoing traffic
				NetworkAddressConfigurationIPv4DHCP[]	
				method string, one of `[static, dhcp]`, **required**	Configuration method used for the address.
		ipv6 NetworkInterfaceAddressFamilyStateIPv6, optional	IPv6 Network address configuration		
			ipv6		
			address_configurations Array of any of NetworkAddressConfigurationIPv6Static, NetworkAddressConfigurationIPv6DHCP or NetworkAddressConfigurationIPv6Auto, optional	NetworkAddressConfigurationIPv6Static[]	
				method string, one of `[static, dhcp]`, required	Configuration method used for the address.
				address NetworkInterfaceAddressIPv6 (string), required	IPv6 network address in CIDR notation
				gateway NetworkAddressIPv6 (string), optional	default gateway for outgoing traffic
				NetworkAddressConfigurationIPv6DHCP[]	
				method string, one of `[static, dhcp, auto]`, **required**	Configuration method used for the address.
				NetworkAddressConfigurationIPv6Auto[]	
				method string, one of `[static, dhcp, auto]`, **required**	Configuration method used for the address.
	errors Array of Error, **required**	Error []			
		code String, optional	machine-readable unique error code		
		mapping String, optional	a reference to the parameter that caused the error		
		message String, optional	human-readable error description		
		May return the following error codes `LPLC.not_found.collection.item`			

1.2.5.4 Modify IPv4 and/or IPv6 address configuration

```
PUT / network / interfaces / {name}
```

Interfaces are readonly except for their IPv6 and IPv4 address configurations.

You can set new address configurations by providing the `address_configurations` key in the respective IP address family object (`ipv4` or `ipv6`). `address_configurations` will replace any existing configuration with the new configuration. If you only want to add a new configuration be sure to submit any existing ones as well.

The `address_configurations` list only affects their respective address family. If you only want to alter the IPv4 address configuration it is sufficient to set the `ipv4.address_configurations` key. Any other address family will remain unaffected as long as you do not alter their own `address_configuration` list as well.

The response to a network configuration change request is returned *before* the new configuration is applied. This ensures that the caller receives an acknowledgment from the API before the network connection may get lost due to the changed configuration. The reconfiguration of the new network setup happens in the background shortly after the response is emitted. Thus the API may close existing connection and will not respond to further requests for a few seconds. Please note that only one configuration change may be requested at a time. Thus the API will delay the response to a second request until all internal processes for the first request are finished. Such a response to a quick subsequent request may be delayed by up to 20 seconds. If too many parallel requests are competing for network setup changes, the HTTP status response 423 (*Locked*) will be returned after a timeout of 20 seconds.

Please Note: Even though the API supports multiple address configurations for each address family only the first will be applied at the moment. This is a pending feature. Therefore you should only provide one address configuration item per address family.

Request

Path Variables
name String, **required**

Body	**application/json**		
Properties (NetworkInterfaceInformation)			
iface NetworkInterfaceName (string), pattern: `^[a-z0-9-]+$`, **required**, read-only	unique name describing a network interface		
hardware_address MacAddress (string), pattern: `^([a-f0-9]{2}:){5}[a-f0-9]{2}$`, **required**, read-only	unique hardware address of a network interface		
has_link boolean, **required**, read-only	current physical connection status (whether a cable is plugged in or not)		
ipv4 NetworkInterfaceAddressFamilyStateIPv4, optional	IPv4 Network address configuration		
	ipv4		
	address_configurations Array of any of NetworkAddressConfigurationIPv4Static or NetworkAddressConfigurationIPv4DHCP, optional	NetworkAddressConfigurationIPv4Static[]	
		method string, one of [`static, dhcp`], **required**	Configuration method used for the address.
		address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation
		gateway NetworkAddressIPv4 (string), optional	default gateway for outgoing traffic

Body	application/json		
		NetworkAddressConfigurationIPv4DHCP[]	
		method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.
ipv6 NetworkInterfaceAddressFamilyStateIPv6, optional	IPv6 Network address configuration		
	ipv6		
	address_configurations Array of any of NetworkAddressConfigurationIPv6Static, NetworkAddressConfigurationIPv6DHCP or NetworkAddressConfigurationIPv6Auto, optional	NetworkAddressConfigurationIPv6Static[]	
		method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.
		address NetworkInterfaceAddressIPv6 (string), **required**	IPv6 network address in CIDR notation
		gateway NetworkAddressIPv6 (string), optional	default gateway for outgoing traffic
		NetworkAddressConfigurationIPv6DHCP[]	
		method string, one of [`static`, `dhcp`, `auto`], **required**	Configuration method used for the address.
		NetworkAddressConfigurationIPv6Auto[]	
		method string, one of [`static`, `dhcp`, `auto`], **required**	Configuration method used for the address.

Examples

Remove all IPv6 address configurations

```
{
  "ipv6": {
    "address_configurations": []
  }
}
```

Replace existing IPv4 configuration with DHCP

```
{
  "ipv4": {
    "address_configurations": [
      {
        "method": "dhcp"
      }
    ]
  }
}
```

Set static and dynamic IPv4 configuration

```
{
  "ipv4": {
    "address_configurations": [
      {
        "method": "dhcp"
      },
      {
        "method": "static",
        "address": "192.168.0.100/24"
      }
    ]
  }
}
```

Response

Code	Body	application/json			
200 400 404 423	Properties (object)				
	data Array of NetworkInterfaceAddressConfigurationState, **required**	NetworkInterfaceAddressConfigurationState[]			
		ipv4 NetworkInterfaceAddressFamilyStateIPv4, optional	IPv4 Network address configuration		
			ipv4		
			address_configurations Array of any of NetworkAddressConfigurationIPv4Static or NetworkAddressConfigurationIPv4DHCP, optional	NetworkAddressConfigurationIPv4Static[]	
				method string, one of [static, dhcp], **required**	Configuration method used for the address.
				address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation
				gateway NetworkAddressIPv4 (string), optiona	default gateway for outgoing traffic
				NetworkAddressConfigurationIPv4DHCP[]	
				method string, one of [static, dhcp], **required**	Configuration method used for the address.
		ipv6 NetworkInterfaceAddressFamilyStateIPv6, optional	IPv6 Network address configuration		
			ipv6		
			address_configurations Array of any of NetworkAddressConfigurationIPv6Static, NetworkAddressConfigurationIPv6DHCP or NetworkAddressConfigurationIPv6Auto, optional	NetworkAddressConfigurationIPv6Static[]	

				method string, one of `[static, dhcp]`, **required**	Configuration method used for the address.
				address NetworkInterfaceAddressIPv6 (string), **required**	IPv6 network address in CIDR notation
				gateway NetworkAddressIPv6 (string), optional	default gateway for outgoing traffic
				NetworkAddressConfigurationIPv6DHCP[]	
				method string, one of `[static, dhcp, auto]`, **required**	Configuration method used for the address.
				NetworkAddressConfigurationIPv6Auto[]	
				method string, one of `[static, dhcp, auto]`, **required**	Configuration method used for the address.
	errors Array of Error, **required**	Error []			
		code String, optional	machine-readable unique error code		
		mapping String, optional	a reference to the parameter that caused the error		
		message String, optional	human-readable error description		

		May return the following error codes
		`LPLC.validation`
		`LPLC.not_found.collection.item`			

1.2.6 Peripherals

1.2.6.1 Get Keypad Information

`GET / peripherals / keypad`

Response

Code	Body	application/json	
200	Properties (KeypadInformation)		
	data KeypadInformation, **required**	Describe the current state of the keypad as well as access to visualization data.	
		KeypadInformation	
		locked boolean, **required**	Boolean flag indicating the state of the key lock (true -> locked, false -> unlocked). All keypad inputs are ignored while the lock is active.
		clear_matcher_before_teach boolean, **required**	The boolean flag controls whether multiple detectables can be stored for a matcher via keypad-based teach operations. A value of true implies, that a teach operation always removes all existing detectables from the currently selected matcher before adding the new detectable. With a value of false previously existing detectables are not deleted before a new one is added.
		visualization_url any of string or null, optional, read-only	The visualization resource location can be used for providing a virtual keypad interface. Its URL may start with a scheme (e.g. *http* or *https*) for a full URL including hostname or it may start with

			a slash, indicating a path provided by the device itself. This attribute cannot be modified.
	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.6.2 Modify Keypad

`PUT / peripherals / keypad`

Modify basic states of the keypad.

Request

Body	application/json	
Properties (KeypadInformation)		
	locked boolean, **required**	Boolean flag indicating the state of the key lock (true -> locked, false -> unlocked). All keypad inputs are ignored while the lock is active.
	clear_matcher_before_teach boolean, **required**	The boolean flag controls whether multiple detectables can be stored for a matcher via keypad-based teach operations. A value of true implies, that a teach operation always removes all existing detectables from the currently selected matcher before adding the new detectable. With a value of false previously existing detectables are not deleted before a new one is added.
	visualization_url any of string or null, optional, read-only	The visualization resource location can be used for providing a virtual keypad interface. Its URL may start with a scheme (e.g. *http* or *https*) for a full URL including hostname or it may start with a slash, indicating a path provided by the device itself. This attribute cannot be modified.

Examples

```
{
  "locked": true,
  "clear_matcher_before_teach": false,
  "visualization_url": "/media/keypad-image.svg"
}
```

Response

Code	**Body**	**application/json**	
200 400	Properties (KeypadInformation)		
	data KeypadInformation, **required**	Describe the current state of the keypad as well as access to visualization data.	
		KeypadInformation	
		locked boolean, **required**	Boolean flag indicating the state of the key lock (true -> locked, false -> unlocked). All keypad inputs are ignored while the lock is active.
		clear_matcher_before_teach boolean, **required**	The boolean flag controls whether multiple detectables can be stored for a matcher via keypad-based teach operations. A value of true implies, that a teach operation always removes all existing detectables from the currently selected matcher before adding the new detectable.

			With a value of false previously existing detectables are not deleted before a new one is added.
		visualization_url any of string or null, optional, read-only	The visualization resource location can be used for providing a virtual keypad interface. Its URL may start with a scheme (e.g. *http* or *https*) for a full URL including hostname or it may start with a slash, indicating a path provided by the device itself. This attribute cannot be modified.
	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes `LPLC.validation`		

1.2.6.3 Retrieve user interactions on the keypad

`GET / peripherals / keypad / events`

Return a list of keypad events.

This collection is implemented as a ring-buffer meaning that older Events will be removed once new events are added.

Response

Code	Body	application/json	
	Properties (object)		
	events Array of KeypadEvent, **required**	KeypadEvent[]	
		source string, **required**	The usual source of events is *inputs*.
		name KeypadEventInput (string), **required**	Name of a keypad input (button) that may trigger events.
		event KeypadEventName (string), **required**	Input peripherals can trigger different events.
		timestamp integer, minimum: 0, **required**	The timestamp is given in milliseconds and should be monotonic increasing.

1.2.6.4 Retrieve a list of available input elements on the keypad

`GET / peripherals / keypad / inputs`

Returns a list of keypad input elements.

Every keypad input element represents a physical button on the keypad.

Response

Code	Body	application/json			
200	Properties (object)				
	data object, **required**	data			
		inputs Array of KeypadInputButton, **required**	KeypadInputButton[]		
			name KeypadEventInput (string), **required**	Name of a keypad input (button) that may trigger events.	

			capabilities Array of object, **required**	object[]	
				name KeypadEventName (string), **required**	Input peripherals can trigger different events.
				url string, **required**	The event can be triggered externally by submitting a POST request against this resource.
		errors Array of Error, **required**	Error []		
			code String, optional	machine-readable unique error code	
			mapping String, optional	a reference to the parameter that caused the error	
			message String, optional	human-readable error description	

1.2.6.5 Simulate a user interaction on the keypad

`POST / peripherals / keypad / inputs / {name} / {event}`

Simulates a button-press by externally triggering the given event for the input.

See the collection of keypad inputs for a list or URLs available for triggering events.

Request

Path Variables
name String, **required**
event String, **required**

Response

Code	
204 **404**	The empty response indicates success

	May return the following error codes
	`LPLC.resource.unspecified`
	`LPLC.resource.invalid`
	`LPLC.illegal_request`

1.2.6.6 Get Output Configuration

`GET / peripherals / outputs`

Response

Code	**Body**	**application/json**	
200	Properties (object)		
	data SwitchingOutputs, **required**	Eletrical output lines can drive external actors in different electrical modes.	
		SwitchingOutputs	
		output_driver SwitchingOutputDriver (string), **required**	The Output Driver defines the electrical behaviour of the switching outputs. The supported output drivers can be retrieved via `/api/sensor/capabilities`.
		count integer, **required**	Number of available output lines

	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.6.7 Modify Output Configuration

`PUT / peripherals / outputs`

Request

Body	**application/json**
Properties (SwitchingOutputsWritable)	
output_driver SwitchingOutputDriver (string), **required**	The Output Driver defines the electrical behaviour of the switching outputs. The supported output drivers can be retrieved via `/api/sensor/capabilities`.

Response

Code	**Body**	**application/json**	
200 400	Properties (object)		
	data SwitchingOutputs, **required**	Eletrical output lines can drive external actors in different electrical modes.	
		SwitchingOutputs	
		output_driver SwitchingOutputDriver (string), **required**	The Output Driver defines the electrical behaviour of the switching outputs. The supported output drivers can be retrieved via `/api/sensor/capabilities`.
		count integer, **required**	Number of available output lines
	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes `LPLC.validation`		

1.2.6.8 Get current interface configuration

`GET / peripherals / rs232`

Response

Code	**Body**	**application/json**	
200 400	Properties (object)		
	data InterfaceRS232, **required**	InterfaceRS232	
		protocol any of SerialModbusSettings or SerialElizaSettings, **required**	SerialModbusSettings
			type string, one of [`none`, `eliza`, `modbus`], default: `eliza`, **required**
			slave_id any of number or null, **required**
			frame_format

			string, one of [rtu, ascii] , default: rtu, **required**
			SerialElizaSettings
			type string, one of [none, eliza, modbus] , default: eliza, **required**
		baud_rate number, one of [9600, 19200, 115200], **required**	
	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.6.9 Modify interface configuration

`PUT / peripherals / rs232`

Request

Body	**application/json**	
Properties (InterfaceRS232)		
protocol any of SerialModbusSettings or SerialElizaSettings, **required**	InterfaceRS232	
	protocol any of SerialModbusSettings or SerialElizaSettings, **required**	SerialModbusSettings
		type string, one of [none, eliza, modbus] , default: eliza, **required**
		slave_id any of number or null, **required**
		frame_format string, one of [rtu, ascii] , default: rtu, **required**
		SerialElizaSettings
		type string, one of [none, eliza, modbus] , default: eliza, **required**
	baud_rate number, one of [9600, 19200, 115200], **required**	

Request

Code	**Body**	**application/json**	
200 400	Properties (object)		
	data InterfaceRS232, **required**	InterfaceRS232	
		protocol any of SerialModbusSettings or SerialElizaSettings, **required**	SerialModbusSettings
			type string, one of [none, eliza, modbus] , default: eliza, **required**
			slave_id any of number or null, **required**
			frame_format string, one of [rtu, ascii] , default: rtu, **required**
			SerialElizaSettings
			type

			string, one of [none, eliza, modbus] , default: eliza, **required**
		baud_rate number, one of [9600, 19200, 115200], **required**	
	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes `LPLC.validation`		

1.2.6.10 Get trigger source event statistics

`GET / peripherals / trigger-sources`

Response

Code	Body	application/json		
200	Properties (object)			
	data TriggerSourcesStatus, **required**	The sensor has a number of input lines that can be used as trigger sources. The event counters are updated periodically (approximately every second).		
		TriggerSourcesStatus		
		trigger_sources Array of object, **required**	object[]	
			name string, **required**	
			event_counters Object, **required**	event_counters
				edge_falling Number, **required**
				edge_rising Number, **required**
				level_low Number, **required**
		errors Array of Error, **required**	Error []	
			code String, optional	machine-readable unique error code
			mapping String, optional	a reference to the parameter that caused the error
			message String, optional	human-readable error description

1.2.6.11 Get current interface configuration

`GET / peripherals / usb`

Response

Code	Body	application/json	
200	Properties (object)		
	data InterfaceUSB, **required**	InterfaceUSB	
		protocol Any of SerialModbusSettings or SerialElizaSettings, **required**	SerialModbusSettings
			type string, one of [none, eliza, modbus] , default: eliza, **required**
			slave_id

			any of number or null, **required**
			frame_format string, one of [`rtu, ascii`], default: `rtu`, **required**
			SerialElizaSettings
			type string, one of [`none, eliza, modbus`], default: `eliza`, **required**
	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.6.12 Modify interface configuration

`PUT / peripherals / usb`

Request

Body	**application/json**
Properties (InterfaceUSB)	
protocol any of SerialModbusSettings or SerialElizaSettings, **required**	SerialModbusSettings
	type string, one of [`none, eliza, modbus`], default: `eliza`, **required**
	slave_id any of number or null, **required**
	frame_format string, one of [`rtu, ascii`], default: `rtu`, **required**
	SerialElizaSettings
	type string, one of [`none, eliza, modbus`], default: `eliza`, **required**

Response

Code	**Body**	**application/json**	
200 400	Properties (object)		
	data InterfaceUSB, **required**	InterfaceUSB	
		protocol Any of SerialModbusSettings or SerialElizaSettings, **required**	SerialModbusSettings
			type string, one of [`none, eliza, modbus`], default: `eliza`, **required**
			slave_id any of number or null, **required**
			frame_format string, one of [`rtu, ascii`], default: `rtu`, **required**
			SerialElizaSettings
			type string, one of [`none, eliza, modbus`], default: `eliza`, **required**
	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes `LPLC.validation`		

1.2.7 Actions

The sensor can be programmed to react on specific external or internal events. The available actions can be either triggered via trigger input lines or via API requests. This allows customized behaviour, e.g. teaching colors via an external button.

This endpoint provides details for all available actions. Additionally actions can be executed for one-time operations.

See *action-triggers* if you want to connect trigger input line events with specific actions for repeated operations.

1.2.7.1 Action descriptions

Every action accepts a a distinct set of optional or mandatory arguments. They are summarized in the `argument` field of each action in the collection. The detailed description of their meaning and the specification of each Action's behaviour is described below.

1.2.7.2 Action „enable_switching_output“

The behavior of the switching outputs changes significantly while this action is configured for at least one *trigger event*. See the introduction chapter about *Switching outputs, triggers and hold time settings* for a detailed specification of the different behaviors.

Arguments: none

1.2.7.3 Action „teach_single“

Sample a new detectable whenever the action is executed.

Arguments:

- `matcher_id` (optional, default: null): the UUID of a matcher or `null`. In case of `null` the new detectable is assigned to a matcher based on `matcher_output_pattern`. In case of the ID belonging to an existing matcher, the new detectable is assigned to this matcher. In case of an ID that does not belong to an existing matcher, a matcher with this ID is created and assigned to the new detectable.

- `matcher_output_pattern` (optional, default: null): output pattern of the target matcher. This field is only considered, if `matcher_id` is `null`. In case of `matcher_output_pattern` being `null`, a new matcher is created whenever the action is executed. Otherwise the `matcher_output_pattern` is expected to be a dictionary containing a `states` field. This `states` field is supposed to contain a list of boolean output states. If a matcher with this output pattern already exists, then the new detectable is added to this matcher. If no matcher with such an output pattern exists, then a new matcher for this output pattern is created before adding the new detectable to it.

- `remove_matcher_detectables_before` (optional, default: true): the boolean value specifies whether detectables belonging to the selected matcher should be removed right before the new detectable is added.

1.2.7.4 Action „keylock“

Control the state of the keypad locking.

A typical approach could be to connect the rising edge of a trigger input to this action with the parameter locked being true and the falling edge of the same trigger input with false. Thus the locking state of the keypad would follow the level of the trigger input.

Arguments:

- `locked` (required): the boolean value specifies the wanted target state of the keypad locking. The `true` value locks the keypad. The `false` value releases the lock.

1.2.7.5 Action „run_autogain“

Start an autogain procedure.

Probably the action "remove_all_detectables" should be executed afterwards, since the color values may not be accurate anymore due to changed sampling settings.

Arguments: none

1.2.7.6 Action „remove_all_detectables“

Clear the detectables collection.

Arguments: none

1.2.7.7 Action „remove_all_matchers“

Clear the matchers collection. This also removes all detectables.

Arguments: none

1.2.7.8 Retrieve Actions

`GET / actions`

Retrieves a list of available Actions

Response

Code	Body	application/json		
200	Properties (object)			
	data object, **required**	data		
		actions Array of Action, **required**	Action []	
			name string, **required**	Unique name of the action
			arguments object, **required**	arguments
	errors Array of Error, **required**	Error []		
		code String, optional	machine-readable unique error code	
		mapping String, optional	a reference to the parameter that caused the error	
		message String, optional	human-readable error description	

1.2.7.9 Get Action

`GET / actions / {itemId}`

Returns a single Action.

Request

Path Variables
itemId String, **required**

Response

Code	Body	application/json	
200	Properties (object)		
	data Action, **required**	The sensor allows the connection of events with actions. Actions can be related to the sensor operations or the information handled by the sensor (e.g. the list of stored detectables). The Action consists of a unique name and a set of optional arguments. The list of available Actions and their possible arguments can be retrieved via `/api/actions`.	
		Action	
		name string, **required**	Unique name of the action
		arguments object, **required**	Arguments
	errors Array of Error, **required**	Error []	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
		May return the following error codes `LPLC.not_found.collection.item`	

1.2.7.10 Software-triggered Actions

`POST / actions / {itemId} / execute`

Execute the given action once. This provides access to all operations that can be connected to trigger input line events.

Additional parameters (if required by the action) can be provided in the body of the request. For example the *keylock* action could be executed by providing a dictionary containing the key *locked* with the wanted boolean target state. See the *arguments* field of each action in the collection above.

The content of the response depends on the specific action that was executed.

The example actions with their respective parameters and responses may not be supported by all sensors. The `Actions` collection contains the authoritative list of supported actions for each sensor.

Request

Path Variables
itemId String, **required**

Body	application/json			
Properties (AnyAction (union))				
ActionEnableSwitchingOutput ActionEnableSwitchingOutput, optional	Apply the *output_pattern* of the currently detected matcher to the switching outputs of the sensor.			
	ActionEnableSwitchingOutput			
	name string, **required**	Unique name of the action		
	arguments	arguments		

Body	**application/json**			
	object, **required**			
ActionTeachDetectable ActionTeachDetectable, optional	Add the currently sampled color as a Detectable to the selected matcher.			
	ActionTeachDetectable			
	arguments object, **required**	arguments		
		matcher_id UUID (string), pattern: `^[a-f0-9-]+$`, optional, read-only	The new Detectable is assigned to the Matcher identified by this UUID. In case this matcher UUID (and "matcher_output_pattern") is undefined, a new matcher is created.	
		matcher_output_pattern: object, optional	Pattern of the switching outputs to be used when selecting the target matcher for the new detectable. A suitable matcher is created, if no matcher with the specified pattern is found. This field is ignored, if "matcher_id" is not null. If no pattern is defined (an no "matcher_id"), then a new matcher is created whenever the corresponding action is executed.	
			matcher_output_pattern:	
			states Array of any of boolean or null, **required**	List of True/False values describing the wanted states of the Switching Outputs
			remove_matcher_detectables_before boolean , default: `true`, optional	Remove all Detectables belonging to the configured Matcher before attaching the new Detectable.
	name string, **required**	Unique name of the action		
ActionKeyLock ActionKeyLock, optional	Change the *locked* state of the keypad. This allows or disallows local access to the sensor via the keypad.			
	ActionKeyLock			
	arguments object, **required**	arguments		
		locked boolean, **required**	Target state of the keypad locking.	
	name string, **required**	Unique name of the action		
ActionRunAutogain ActionRunAutogain, optional	Start an automatic adjustment of the optiocal sensor setup. See `/api/sensor/detection-profiles/current/autogain for details.`			
	ActionRunAutogain			
	name string, **required**	Unique name of the action		
	arguments	arguments		

Body	application/json			
	object, **required**			
ActionRemoveAllDetectables ActionRemoveAllDetectables, optional	Remove all stored Detectables belonging to any Matcher.			
	ActionRemoveAllDetectables			
	name string, **required**	Unique name of the action		
	arguments object, **required**	arguments		
ActionRemoveAllMatchers ActionRemoveAllMatchers, optional	Remove all stored Matchers (including the related detectables).			
	ActionRemoveAllMatchers			
	name string, **required**	Unique name of the action		
	arguments object, **required**	arguments		

Response

Code	Body	application/json			
200 204	Properties (object)				
	data any of ActionResultEnableSwitchingOutput, ActionResultTeachDetectable, ActionResultKeyLock, ActionResultRunAutogain, ActionResultRemoveAllDetectables (string) or ActionResultRemoveAllMatchers (string), **required**	ActionResultEnableSwitchingOutput			
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
		timestamp TimestampBackendUptime (number), **required**	The timestamp (given in microseconds) is based on the uptime of the internal analog sensor backend. It may get reset to zero under specific conditions.		
		corrected_color CorrectedColor, **required**	Representation of a color in the colorspace XYZ.		
			values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace	
		transformed_color TransformedColor, **required**	A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the `colorspace.axes` property of currently used detection profile.		
			values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace	
		representations ColorRepresentations, **required**	Pre-calculated visual representations of a color suitable for rendering		

Code	Body	application/json			
			RGB Array of number, minimum items: 3, maximum items: 3, **required**	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.	
		inputs InputsState, **required**	The state of all inputs during a given period is specified by a list of possible events combined with a boolean value indicating, if the given event occurred within the period.		
			// boolean, **required**	The boolean value indicates whether the named input event occurred during the last period.	
		detection ColorMatchingResult, **required**	After each sampling period the retrieved color value is compared to the stored detectables (color positions). Detectables are ignored, if the tolerance shape of their corresponding matcher does not encompass the current sample. Finally the closes suitable detectable is selected as the winner of the color matching operation. The corresponding matcher determines the state of the sensor for the duration of the next sampling period.		
			detection		
			matcher any of UUID (string) or null, optional, Deprecated	Deprecated: use "chosen_matcher_id" instead	
			chosen_matcher_id any of UUID (string) or null, **required**	unique identifier of the selected matcher	
			distances Array of any of number or null, **required**	Distance between the sample's color position and the selected matcher's closest color position along the three axes of the color space. The array contains three 'null' values, if no suitable matcher was found for the current color sample.	
			output_pattern CurrentSwitchingOutputsState, **required**	Currently active state of the Switching Outputs. Beware	

Code	Body	application/json			
				that this may deviate from the specified output states of the current best matcher, since settings like *triggered input* or *hold time* influence update process for the Switching Outputs.	
				states Array of any of boolean or null, **required**	List of True/False values describing the wanted states of the Switching Outputs
		signal_level number, **required**	The signal level indicates the usage of the internal ADC sampling range. This		
		ActionResultTeachDetectable			
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
		alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.		
		matcher_id UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	reference to the *Matcher* containing this Detectable		
		color TransformedColor, required	A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the `colorspace.axes` property of currently used detection profile.		
			color		
			values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace	
		representations ColorRepresentations, optional, read-only	Pre-calculcated visual representations of a color suitable for rendering		
		ActionResultKeyLock			
		locked boolean, **required**	New state of the keypad locking.		
		ActionResultRunAutogain			
		level	Target value for the auto-gain procedure		

Code	Body	application/json			
		Number, default: `0.8`, minimum: 0.01, maximum: 1, optional			
		minimum_sample_rate SampleRate (number), minimum: 0.02, optional	Desired sample rate (the default is the current sample rate)		
		enable_internal_emitter Boolean, default: `true`, optional	controls the power of the internal light source		
		enable_ambient_light_compensation Boolean, default: `true`, optional	Control the ambient light compensation procedure. This setting is only relevant if `enable_internal_emitter` is set to true. The ambient light compensation leads to a pulsed usage of the internal light emitter. Samples are collected for alternating light and dark phases. This allows to calculate a color sample of the target excluding any optical interference from external light sources. You should not disable ambient light compensation unless the optical path is perfectly isolated. Otherwise external light will inevitably interfere with the color sampling.		
		averages AverageSampleCount (integer), minimum: 1, optional	Number of previous samples to be averaged for every sampling result. A rolling averaging algorithm is applied to the samples.		
	errors Array of Error, **required**	Error []			
		code String, optional	machine-readable unique error code		
		mapping String, optional	a reference to the parameter that caused the error		
		message String, optional	human-readable error description		

1.2.8 Defaults

Collection of defaults and settings for specific tasks. Apart from the custom user-defined values this API also returns factory defaults. Defaults are implicitly applied during specific actions like the creation of matchers or when executing certain behaviours.

Client applications can use this API endpoint to store settings that are independent from their current session or the client itself. Type and validation checks are the responsibility of the client application.

Be aware that defaults may be applied at runtime (like the creation of a matcher) and thus an invalid default value will break the application at a later point in time. Mind the notes below to prevent such problems when operating with the defaults API.

- Default values are not subject to any kind of validation, but are handled as raw data. An invalid hold_time (negative, string instead of a number) for a matcher will become effective during the creation of a new matcher and only if the request for creating the matcher did not contain a hold_time field. In such a case the creation of a matcher would fail. It is therefore paramount to properly validate default values.

- The fields object_type and key are yours to choose. This allows applications to store session- and client-independent data (e.g. an interface theme, color scheme, etc.). In order to avoid name-collisions with internal default-fields you should prefix the object_type or key field with x- (e.g. x-theme instead of theme). The API will never use fields internally that start with x-.
- The API resolves defaults with the following steps. Applications should implement the same behaviour, when resolving default values:
    1. Check if an element in the defaults collection matches both object_type and key
    2. In case it does: use this value
    3. In case it does not: use the value from factory_defaults

1.2.8.1 Retrieve DefaultMapValues

`GET / defaults`

Returns two collections of *DefaultMapValue* objects. `defaults` contains all custom defaults and `factory_defaults` all those that are part of the factory settings. The latter can't be changed but custom defaults take precedence over factory defaults.

Response

Code	Body	application/json		
200	Properties (object)			
	data object, **required**	data		
		defaults Array of DefaultsMapValue, **required**	DefaultsMapValue[]	
			uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
			object_type string, required, read-only	Name of the object the default is meant for
			key string, required, read-only	name of the object's property
			value any, **required**	Actual default value for the object's property
		factory_defaults Array of DefaultsMapValue, **required**	DefaultsMapValue[]	
			uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
			object_type string, **required**, read-only	Name of the object the default is meant for
			key string, required, read-only	name of the object's property
			value any, **required**	Actual default value for the object's property
	errors Array of Error, **required**	Error[]		
		code String, optional	machine-readable unique error code	
		mapping String, optional	a reference to the parameter that caused the error	

Code	Body	application/json		
		message String, optional	human-readable error description	

1.2.8.2 Create DefaultMapValues

`POST / defaults`

All valid attributes for a PUT request of a defaults object are allowed. The attributes `object_type`, `key` and `value` are required. The API ensures that only one combination of `object_type` and `key` is present at a time. A POST request therefore doesn't necessarily increase the number of elements in the collection.

Request

Body	application/json
Properties (DefaultsMapValue)	
uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
object_type string, **required**, read-only	Name of the object the default is meant for
key string, **required**, read-only	name of the object's property
value any, **required**	Actual default value for the object's property

Examples

Matcher: Tolerance

```
{
  "uuid": "a7bd36b3-e9c1-4f60-8d7e-cf47634a28b1",
  "object_type": "matcher",
  "key": "tolerance",
  "value": {
    "shape": "sphere",
    "limits": {
      "radius": 4
    }
  }
}
```

Matcher: Hold Time

```
{
  "uuid": "55b35901-1ea6-4b3d-864a-60af15a9b0c5",
  "object_type": "matcher",
  "key": "hold_time",
  "value": 0
}
```

Matcher: reset output after Hold Time expiry

```
{
  "uuid": "9ba8a7a4-7fa5-4bfc-8883-98d7b6084e91",
  "object_type": "matcher",
  "key": "reset_output_after_hold_time_expired",
  "value": false
}
```

Autogain: number of samples used for averaging

```
{
  "uuid": "eeb46031-10e5-4f13-901a-c7eb16aa0cf9",
  "object_type": "autogain",
  "key": "averages",
  "value": 0
}
```

Response

Code	Body	application/json	
200 400	Properties (object)		
	data DefaultsMapValue, **required**	DefaultsMapValue	
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
		object_type string, **required**, read-only	Name of the object the default is meant for
		key string, **required**, read-only	name of the object's property
		value any, **required**	Actual default value for the object's property
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes LPLC.validation.collection_size_exceeded		

1.2.8.3 Remove multiple oder all DefaultMapValues

`DELETE / defaults`

Remove a selection of DefaultMapValues either based on a given filter argument (if supported for this collection) or remove all DefaultMapValues from the collection.

All delete requests result in an empty success response (204). This is even valid for a non-filtered DELETE request against an empty collection or for a filtered DELETE request against a collection without DefaultMapValues matching the filter.

Response

Code	
204	The empty response indicates success

1.2.8.4 Modify DefaultMapValue

`PUT / defaults / {itemId}`

Modify the default's value. The fields `uuid`, `object_type` and `key` are invariable.

Request

PathVariables		
itemId String, **required**		
Body	**application/json**	
Properties (object)		
data DefaultsMapValue, **required**	DefaultsMapValue	

	uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
	object_type string, **required**, read-only	Name of the object the default is meant for
	key string, **required**, read-only	name of the object's property
	value any, **required**	Actual default value for the object's property

Examples

Matcher: Tolerance

```
{
  "uuid": "a7bd36b3-e9c1-4f60-8d7e-cf47634a28b1",
  "object_type": "matcher",
  "key": "tolerance",
  "value": {
    "shape": "sphere",
    "limits": {
      "radius": 4
    }
  }
}
```

Matcher: Hold Time

```
{
  "uuid": "55b35901-1ea6-4b3d-864a-60af15a9b0c5",
  "object_type": "matcher",
  "key": "hold_time",
  "value": 0
}
```

Matcher: reset output after Hold Time expiry

```
{
  "uuid": "9ba8a7a4-7fa5-4bfc-8883-98d7b6084e91",
  "object_type": "matcher",
  "key": "reset_output_after_hold_time_expired",
  "value": false
}
```

Autogain: number of samples used for averaging

```
{
  "uuid": "eeb46031-10e5-4f13-901a-c7eb16aa0cf9",
  "object_type": "autogain",
  "key": "averages",
  "value": 0
}
```

Response

Code	Body	application/json	
200 400 404	Properties (object)		
	data DefaultsMapValue, **required**	DefaultsMapValue	

		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
		object_type string, **required**, read-only	Name of the object the default is meant for
		key string, **required**, read-only	name of the object's property
		value any, **required**	Actual default value for the object's property
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes LPLC.not_found.collection.item		

1.2.8.5 Delete DefaultMapValue

```
DELETE / defaults / {itemId}
```

Deletes a single DefaultMapValue

Request

PathVariables
itemId String, **required**

Response

Code	
204	The empty response indicates success
	May return the following error codes LPLC.not_found.collection.item

1.2.8.6 Get DefaultMapValue

```
GET / defaults / {itemId}
```

Returns a single DefaultMapValue.

Request

PathVariables
itemId String, **required**

Response

Code	**Body**	**application/json**	
200	Properties (object)		
	data DefaultsMapValue, **required**	DefaultsMapValue	
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
		object_type string, **required**, read-only	Name of the object the default is meant for
		key string, **required**, read-only	name of the object's property
		value any, **required**	Actual default value for the object's property
	errors Array of Error, **required**	Error[]	

		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes LPLC.not_found.collection.item		

1.2.9 Firmware

The firmware is stored on the device and controls all of its aspects. It can be upgraded and safely be reset to the factory defaults.

1.2.9.1 Get Firmware Information

```
GET / firmware
```

Returns information about the currently running firmware.

Response

Code	Body	application/json	
200	Properties (object)		
	data FirmwareInformation, **required**	Information describing a firmware version.	
		FirmwareInformation	
		id FirmwareBuildId (string), pattern: ^[a-f0-9]+$, **required**	unique ID of the currently running firmware image
		channel ReleaseChannel (string), one of [stable, feature] , default: stable, **required**	Describes the kind of a publication Releases on the `stable` channel are generally considered well-tested and are recommended for use in production. Releases on the `feature` add new features but haven't been tested as much as a stable release. Feature releases can but should only be used in production with careful consideration.
		created_on Timestamp (string), **required**	time this firmware build was created
		name string, **required**	human-readable name of this release
		notes string, **required**	Release notes formatted as markdown
		version FirmwareVersion (string), **required**	version of a firmware
		works_with Array of string, **required**	compatible device models (see `model_key` in `/api/device`)
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.9.2 Firmware Image Upload

```
GET / firmware / images
```

Returns the list of all active partial or complete firmware uploads.

Response

Code	Body	application/json	

200	Properties (object)		
	data Array of FirmwareImageUpload, **required**	FirmwareImageUpload []	
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	Unique ID of a firmware upload
		build_id HashDigest (string), pattern: `^[a-f0-9]+$`, **required**	unique ID of the currently running firmware image
		status string, one of `[incomplete, complete, invalid_signature, processing_failure, malformed_content, device_mismatch]`, **required**	Current status of the firmware upload Incomplete the number of bytes received is lower than the number of bytes that have been announced complete the firmware upload is complete and the new firmware can be applied invalid_signature the firmware checksum didn't match the expected value processing_failure an internal undefined error occurred while processing the firmware malformed_content the uploaded firmware image uses an unexpected format or misses essential information device_mismatch the firmware image can not be applied to this device
		uploaded_size integer, minimum: 0, **required**	number of uploaded bytes
		expected_size integer, minimum: 1, **required**	expected total number of bytes for the firmware image
		max_chunk_size integer, minimum: 1, **required**	maximum size for a data chunk uploaded to the device
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.9.3 Upload Firmware

```
POST / firmware / images
```

Upload a new firmware for an upgrade in separate chunks or as a single form-based file upload.

1.2.9.4 Firmware Upload methods

Two different approaches are available for the upload of a firmware image. The client selects the wanted method by using the associated request format:

```
* `multipart` format: Upload the complete image in a single request.
* `JSON` body: Successively upload single blocks of the firmware image.
```

File-based upload ("multipart"):

The file based approach is simple to use and preferable for most situations. It requires the use of the `multipart/form-data` request format. The request transmits the full firmware image file to the API. The API's response to this request is emitted as soon as all related operations are finished.

The request format `multipart/form-data` is commonly used for file based HTML forms. Thus it is also possible to use this firmware upload method with a simple HTML form even without any client side code.

Chunk-based upload (JSON body):

The chunk based approach requires more effort on the client side. This approach may be helpful if you want to achieve advanced flow control or status indications during the firmware upload. Use a JSON formatted request body if you want to use this method.

The initial POST request creates and returns a firmware upload entity (`FirmwareImageUpload`). You may use its UUID for uploading the chunks of the firmware image via subsequent POST requests to `/api/firmware/images/UPLOAD_UUID/upload`. The firmware upload can be finalized and applied by a POST request to `/api/firmware/images/UPLOAD_UUID/apply`.

1.2.9.5 Error handling

In case of an non-recoverable error the API will return a 400 (Bad Request) HTTP status code as early as possible. If the `apply` parameter has been set to a positive value the status code will be 424 (Failed Dependency).

Request

Body	**multipart/form-data**	**application/json**
Properties(object)		
firmware_file FirmwareImageFile (file), **required**	The actual binary firmware image file. Please note that a `filename` (with an arbitrary value) needs to be supplied (in technical terms: the `Content-Disposition` header of this part of the request needs to have a `name` and a `filename` field).	
apply integer, one of [0, 1] , default: 0, **required**	Whether to apply the firmware once it has been received and validated. If this field is set to `1` the firmware will be applied at once, otherwise the API returns the firmware details to allow the application to send a separate request to apply the firmware.	

Response

Code	**Body**	**application/json**	
200 400 424	Properties (object)		
	data FirmwareImageUpload, **required**	A fully or partially uploaded firmware image to be used for upgrading the firmware	
		FirmwareImageUpload []	
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	Unique ID of a firmware upload
		build_id HashDigest (string), pattern: `^[a-f0-9]+$`, **required**	unique ID of the currently running firmware image
		status string, one of `[incomplete, complete, invalid_signature, processing_failure, malformed_content, device_mismatch]`, **required**	Current status of the firmware upload incomplete the number of bytes received is lower than the number of bytes that have been announced complete the firmware upload is complete and the new firmware can be applied invalid_signature

			the firmware checksum didn't match the expected value processing_failure an internal undefined error occurred while processing the firmware malformed_content the uploaded firmware image uses an unexpected format or misses essential information device_mismatch the firmware image can not be applied to this device
		uploaded_size integer, minimum: 0, **required**	number of uploaded bytes
		expected_size integer, minimum: 1, **required**	expected total number of bytes for the firmware image
		max_chunk_size integer, minimum: 1, **required**	maximum size for a data chunk uploaded to the device
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

	May return the following error codes
	`LPLC.validation.missing_input`
	`LPLC.validation.string`
	`LPLC.validation.non_negative_integer`
	`LPLC.validation.positive_integer`
	`LPLC.validation.smaller_integer`
	`LPLC.format.malformed.upload`
	`LPLC.internal_error`		

1.2.9.6 Get firmware image upload

`GET / firmware / images / {itemId}`

Returns a single firmware image upload.

Request

Path Variables
itemId String, **required**

Response

Code	**Body**	**application/json**	
200	Properties (object)		
	data FirmwareImageUpload, **required**	A fully or partially uploaded firmware image to be used for upgrading the firmware	
		FirmwareImageUpload []	
		uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	Unique ID of a firmware upload
		build_id HashDigest (string), pattern: `^[a-f0-9]+$`, **required**	unique ID of the currently running firmware image
		status string, one of `[incomplete, complete, invalid_signature, processing_failure, malformed_content, device_mismatch]`, **required**	Current status of the firmware upload incomplete the number of bytes received is lower than the number of bytes that have been announced complete the firmware upload is complete and the new firmware can be applied invalid_signature the firmware checksum didn't match the expected value processing_failure an internal undefined error occurred while processing the firmware malformed_content the uploaded firmware image uses an unexpected format or misses essential information device_mismatch

			the firmware image can not be applied to this device
		uploaded_size integer, minimum: 0, **required**	number of uploaded bytes
		expected_size integer, minimum: 1, **required**	expected total number of bytes for the firmware image
		max_chunk_size integer, minimum: 1, **required**	maximum size for a data chunk uploaded to the device
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes `LPLC.not_found.collection.item`		

1.2.9.7 Delete firmware image upload

`GET / firmware / images / {itemId}`

Deletes a single firmware image upload.

Request

Path Variables
itemId String, **required**

Response

Code	
204	The empty response indicates success.
	May return the following error codes `LPLC.not_found.collection.item`

1.2.9.8 Switch to new Firmware

`POST / firmware / images / {itemId} / apply`

Applies the firmware to the device thus overwriting the current system image followed by a reboot of the device.

Request

Path Variables
itemId String, **required**

Response

Code	
204 400 424 500	The empty response indicates success.

	May return the following error codes
	`LPLC.format.malformed.upload`
	`LPLC.internal_error`

1.2.9.9 Upload Chunk of Firmware

`POST / firmware / images / {itemId} / upload`

Uploads a data chunck (see `max_chunk_size`) for this firmware image. The `Content-Range` is send by the client and used by the server to determine where the chunk is inserted into the final image. Uploads must happen synchronous so that every chunks start address is defined as (last_chunk_end_address + 1).

New chunks can be uploaded as long as the firmware status is reported as `incomplete`. All other status indicate an either successful or defective firmware upload. In case of a permanent failure all subsequent chunk uploads will be terminated with a HTTP 400 (*Bad Request*) status code.

Request

Path Variables	
itemId String, **required**	
Request Headers	
Content-Range string, pattern: `^bytes\s+\d+-\d+/\d+$`, **required**	Defines where the chunk is positioned in the firmware image file.

Response

Code	
204 400	The empty response indicates success.

	May return the following error codes
	`LPLC.format.malformed.upload`
	`LPLC.header.content_range.conflicts`
	`LPLC.header.content_range.invalid`
	`LPLC.header.content_range.missing`
	`LPLC.upload.missing_chunk`
	`LPLC.payload_too_big`

1.2.9.10 Get Recovery Firmware Information

`GET / firmware / recovery`

Returns information about the current recovery firmware.

Response

Code	**Body**	**application/json**	
200	Properties (object)		
	data FirmwareInformation, **required**	Information describing a firmware version.	
		FirmwareInformation	
		id FirmwareBuildId (string), pattern: ^[a-f0-9]+$, **required**	unique ID of the currently running firmware image
		channel ReleaseChannel (string), one of [stable, feature] , default: stable, **required**	Describes the kind of a publication Releases on the `stable` channel are generally considered well-tested and are recommended for use in production. Releases on the `feature` add new features but haven’t been tested as much as a stable release. Feature releases can but should only be used in production with careful consideration.
		created_on Timestamp (string), **required**	time this firmware build was created
		name string, **required**	human-readable name of this release

		notes string, **required**	Release notes formatted as markdown
		version FirmwareVersion (string), **required**	version of a firmware
		works_with Array of string, **required**	compatible device models (see `model_key` in `/api/device`)
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.9.11 Upgrade Recovery Firmware

`POST / firmware / recovery / upgrade-from-current`

Replaces the stored recovery image with the current system firmware. This is helpful in case you want to update the recovery image to a more recent version.

The factory image merely contains the actual firmware. It does not store the sensors configuration or settings.

The update process will take several minutes.

Response

Code	
204 500	The empty response indicates success

1.2.9.12 Get Firmware Settings

`GET / firmware / settings`

Returns current settings regarding the firmware and possible upgrades.

Response

Code	**Body**	**application/json**	
200	Properties (object)		
	data FirmwareSettings, **required**	Settings related to the device's firmware and upgrades.	
		FirmwareSettings	
		release_channel ReleaseChannel (string), one of [stable, feature] , default: stable, **required**	Describes the kind of a publication Releases on the `stable` channel are generally considered well-tested and are recommended for use in production. Releases on the `feature` add new features but haven’t been tested as much as a stable release. Feature releases can but should only be used in production with careful consideration.
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code

		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.9.13 /firmware/settings

`PUT / firmware / settings`

Request

Body	application/json
Properties (FirmwareSettings)	
release_channel ReleaseChannel (string), one of [stable, feature] , default: stable, **required**	Describes the kind of a publication Releases on the `stable` channel are generally considered well-tested and are recommended for use in production. Releases on the `feature` add new features but haven’t been tested as much as a stable release. Feature releases can but should only be used in production with careful consideration.

Examples

```
{
  "release_channel": "stable"
}
```

Response

Code	Body	application/json	
200 400	Properties (object)		
	data FirmwareSettings, **required**	Settings related to the device's firmware and upgrades.	
		FirmwareSettings	
		release_channel ReleaseChannel (string), one of [stable, feature] , default: stable, **required**	Describes the kind of a publication Releases on the `stable` channel are generally considered well-tested and are recommended for use in production. Releases on the `feature` add new features but haven’t been tested as much as a stable release. Feature releases can but should only be used in production with careful consideration.
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes `LPLC.validation`		

1.2.9.14 Get Firmware Status

`GET / firmware / settings`

Returns information about the currently running firmware version.

Response

Code	Body	application/json	
200	Properties (object)		
	data	Information describing the currently running firmware.	

	FirmwareRunningInformation, **required**		
		FirmwareRunningInformation	
		build_id FirmwareBuildId (string), pattern: `^[a-f0-9]+$`, **required**	unique ID of the currently running firmware image
		source_url any of string or null, optional	Absolute base URL of a firmware repository delivering firmware images suitable for this device
		version FirmwareVersion (string), **required**	version of a firmware
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.10 Access Control

Manage access to data and settings of the sensor.

1.2.10.1 Users

Users are the identities that are allowed to access the API.
A password is required for authenticating a user during login.

1.2.10.2 Roles

Roles describe a set of permissions. Each user may belong to multiple roles.

The role named anonymous is special: it cannot be assigned to users. Instead it describes the set of permissions that are granted to every unauthenticated as well as authenticated request. Thus this special role can be considered the minimum set of permissions that is open for everyone.

1.2.10.3 Auswertung der Zugriffsberechtigungen

All actions can be executed without authentication by default as long as no user account has been created.

If at least one user exists, access control is applied by the API. Thus permissions are checked before an incoming request is processed.

Authentication is conducted via the HTTP Basic Authentication Scheme.

Authorisation for a given action (e.g. *view* or *edit*) targeted at a specific API endpoint is verified as follows:

- Which kind of action is requested by the user: *view* (*GET*) or *edit* (*POST*, *PUT*, *DELETE*)?
- To which access *scope* does the target API endpoint belong (e.g. *network*)?
- Which roles are assigned to the authenticated user (e.g. *operator*)?
- Does at least one of the permissions of any of these roles belong to the requested *scope* and contain the requested *action*?

The last of the questions above decides, whether a request is processed or rejected.

1.2.10.4 HTTP Responses

The following additional HTTP responses can be emitted while access control is active:

HTTP 401 Unauthorized

> is returned if the request requires authentication, but no credentials were supplied or the given credentials were rejected (e.g. unknown user or wrong password). Web applications interfacing the API may want to use the `X-WWW-Authenticate-Scheme-Disable` header (see below) in order to prevent the user's browser from intercepting this error response.

HTTP 403 Forbidden

> is returned if the given credentials were valid, but the associated user is not allowed to request the given *action* in the target *scope*.

The client may specify the `X-WWW-Authenticate-Scheme-Disable` HTTP header in any request. The content of this header is expected to be a comma-separated list of authentication schemes (see RFC 7235). These authentication schemes will *not* be advertised by the API as part of the `WWW-Authenticate` header in its response. The following example header content is suitable for preventing a browser from interfering with authentication related responses: -`WWW-Authenticate-Scheme-Disable: Basic, Digest`.

1.2.10.5 Inspect Access Control Scopes and Actions

`GET / access`

Inspect the available aspects of the access control setup.

Response

Code	Body	application/json	
200	Properties (object)		
	data AccessControlFeatures, **required**	AccessControlFeatues	
		actions Array of AccessAction (string), **required**	Available actions that can be allowed or denied via permissions.
		scopes Array of AccessScope (string), **required**	Available scopes that can be accessed with the different actions.
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description

1.2.10.6 Login into an account

`POST / access / login`

This endpoint can be used to create a session for the current user agent or testing credentials.

If the caller provided valid credentials and didn’t set the `session_timeout` to `0` the response will contain a `Set-Cookie` header that contains a session token used for future authentication and which is automatically handled by `XMLHttpRequest` and `fetch`.

Once a session token has been issued to the user agent any subsequent request to any of the API endpoints will reset the session timeout to the value provided by `session_timeout` as long as the session did not already expire at the point in time when the request was sent to the API. Therefore if the session timeout was set to 15 minutes and a request was made every 10 minutes the session would be valid indefinitely. This does not apply if the endpoint is explicitly excluded from resetting the session timeout (like `GET /api/access/login`).

To test credentials the client may send them along with `session_timeout` set to `0`. The response status code will indicate if the credentials are valid but omits the `Set-Cookie` header thus retaining the currently used session token.

Be aware that the API may start to rate-limit the endpoint if too many invalid credentials have been send to it. Make sure that you implement some kind of user feedback in case of responses with HTTP status code 429 like deactivating the login form and/or displaying the remaining time until a new login may be attempted.

Request

Body	**application/json**
Properties (object)	
username string, **required**	The name of the user that should be authenticated
password string, **required**	The password of the user that should be authenticated
session_timeout number , default: `1200`, optional	The lifetime of the session on the server-side in seconds. Passing `0` will prevent the API from setting the `Set-Cookie` header and allows for checking credentials without creating a new session.

Response

Code				
200 403 429	Authentication with the provided credentials was successful			
	Body	**application/json**		
	Properties (object)			
	data LoginInformation, **required**	Describes the currently active login provided by the user agent		
		LoginInformation		
		logged_in_user any of User or null, **required**	The currently logged in user. Is `null` if the credentials didn't match any known user or have expired.	
			User	
			name string, pattern: `^[\w-]+$`, required, read-only	unique name identifying an account
			password string, optional	Password assigned to this account (only writable; never returned in responses). Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
			password_hash HashDigest (string), pattern: `^[a-f0-9]+$`, optional	Password hash assigned to this account. Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
			roles Array of string, optional	The roles assigned to an account define its set of permissions.
		session_timeout any of number or null, **required**	Number of seconds this session has left before expiring. Is `null` if the provided credentials could not be	

			matched to any active sessions, if the session expired or if the supplied authentication mechanism does not support sessions (e.g. HTTP Authentication).	
	errors Array of Error, **required**	Error[]		
		code String, optional	machine-readable unique error code	
		mapping String, optional	a reference to the parameter that caused the error	
		message String, optional	human-readable error description	

1.2.10.7 Retrieve Information about the currently used Credentials

`GET / access / login`

Response

Code	Body	application/json		
200	Properties (object)			
	data LoginInformation, **required**	Describes the currently active login provided by the user agent		
		LoginInformation		
		logged_in_user any of User or null, **required**	The currently logged in user. Is `null` if the credentials didn't match any known user or have expired.	
			User	
			name string, pattern: `^[\w-]+$`, **required**, read-only	unique name identifying an account
			password string, optional	Password assigned to this account (only writable; never returned in responses). Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
			password_hash HashDigest (string), pattern: `^[a-f0-9]+$`, optional	Password hash assigned to this account. Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
			roles Array of string, optional	The roles assigned to an account define its set of permissions.
		session_timeout any of number or null, **required**	Number of seconds this session has left before expiring. Is `null` if the provided credentials could not be matched to any active sessions, if the session expired or if the supplied authentication mechanism does not support sessions (e.g. HTTP Authentication).	
	errors Array of Error, **required**	Error[]		

		code String, optional	machine-readable unique error code	
		mapping String, optional	a reference to the parameter that caused the error	
		message String, optional	human-readable error description	

1.2.10.8 Logout / Invalidate any current credentials

`DELETE / access / login`

Response

Code	
204 400 403	The provided credentials were successfully invalidated.

1.2.10.9 Create User

`POST / access / users`

Create a new User.

All supported data attributes in the body of the request are optional.

Request

Body	application/json
Properties (User)	
name string, pattern: `^[\w-]+$`, **required**, read-only	unique name identifying an account
password string, optional	Password assigned to this account (only writable; never returned in responses). Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
password_hash HashDigest (string), pattern: `^[a-f0-9]+$`, optional	Password hash assigned to this account. Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
roles Array of string, optional	The roles assigned to an account define its set of permissions.

Examples

```
{
  "name": "alice"
}
```

Response

Code	Body	application/json	
200 400	Properties (object)		
	data User, **required**	User	
		name string, pattern: `^[\w-]+$`, **required**, read-only	unique name identifying an account
		password string, optional	Password assigned to this account (only writable; never returned in responses). Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
		password_hash HashDigest (string), pattern: `^[a-f0-9]+$`, optional	Password hash assigned to this account. Either a `password` or a `password_hash` needs to be

			supplied when creating a new user or changing a password.
		roles Array of string, optional	The roles assigned to an account define its set of permissions.
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes `LPLC.validation.collection_size_exceeded`		

1.2.10.10Remove multiple or all Users

`DELETE / access / users`

Remove a selection of Users either based on a given filter argument (if supported for this collection) or remove all Users from the collection.

All delete requests result in an empty success response (204). This is even valid for a non-filtered DELETE request against an empty collection or for a filtered DELETE request against a collection without Users matching the filter.

Response

Code	
204	The empty response indicates success

1.2.10.11Retrieve User

`GET / access / users`

Retrieves a list of available Users

Response

Code	**Body**	**application/json**		
200	Properties (object)			
	data Object, **required**	data		
		users Array of User, **required**	User[]	
			name string, pattern: `^[\w-]+$`, required, read-only	unique name identifying an account
			password string, optional	Password assigned to this account (only writable; never returned in responses). Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
			password_hash HashDigest (string), pattern: `^[a-f0-9]+$`, optional	Password hash assigned to this account. Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
			roles Array of string, optional	The roles assigned to an account define its set of permissions.
	errors Array of Error, **required**	Error[]		

		code String, optional	machine-readable unique error code	
		mapping String, optional	a reference to the parameter that caused the error	
		message String, optional	human-readable error description	

1.2.10.12 Delete User

`DELETE / access / users / {name}`

Deletes a single User.

Request

Path Variables
name String, **required**

Response

Code	
204	The empty response indicates success
	May return the following error codes `LPLC.not_found.collection.item`

1.2.10.13 Modify User

`PUT / access / users / {name}`

Modifies a single User.

Request

Path Variables	
name String, **required**	
Body	**application/json**
Properties (User)	
name string, pattern: `^[\w-]+$`, **required**, read-only	unique name identifying an account
password string, optional	Password assigned to this account (only writable; never returned in responses). Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
password_hash HashDigest (string), pattern: `^[a-f0-9]+$`, optional	Password hash assigned to this account. Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
roles Array of string, optional	The roles assigned to an account define its set of permissions.

Examples

```
{
  "name": "alice"
}
```

Response

Code	**Body**	**application/json**	
200 400 404	Properties (object)		
	data User, **required**	User[]	
		name	unique name identifying an account

		string, pattern: `^[\w-]+$`, **required**, read-only	
		password string, optional	Password assigned to this account (only writable; never returned in responses). Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
		password_hash HashDigest (string), pattern: `^[a-f0-9]+$`, optional	Password hash assigned to this account. Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
		roles Array of string, optional	The roles assigned to an account define its set of permissions.
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes `LPLC.not_found.collection.item`		

1.2.10.14Get User

`GET / access / users /{name}`

Returns a single User.

Request

Path Variables
name String, **required**

Response

Code	**Body**	**application/json**	
200	Properties (object)		
	data User, **required**	User[]	
		name string, pattern: `^[\w-]+$`, **required**, read-only	unique name identifying an account
		password string, optional	Password assigned to this account (only writable; never returned in responses). Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
		password_hash HashDigest (string), pattern: `^[a-f0-9]+$`, optional	Password hash assigned to this account. Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
		roles Array of string, optional	The roles assigned to an account define its set of permissions.
	errors Array of Error, **required**	Error[]	
		code String, optional	machine-readable unique error code
		mapping String, optional	a reference to the parameter that caused the error
		message String, optional	human-readable error description
	May return the following error codes		

	LPLC.not_found.collection.item		

1.2.10.15Retrieve AccessRoles

`GET / access / roles`

Retrieves a list of available AccessRoles.

Response

Code	Body	application/json			
200	Properties (object)				
	data object, **required**	data			
		roles Array of AccessRole, **required**	AccessRole[]		
			id string, **required**		
			permissions Array of AccessPermission, **required**	AccessPermission[]	
				scope AccessScope (string), **required**	Every API endpoint belongs to an access control scope. A scope combined with a number of actions forms a permission.
				actions Array of AccessAction (string), **required**	
	errors Array of Error, **required**	**Error**[]			
		code String, optional	machine-readable unique error code		
		mapping String, optional	a reference to the parameter that caused the error		
		message String, optional	human-readable error description		

1.3 REST-API Type Reference

1.3.1 AccessAction

Type Information

AccessAction (string)

Examples:

```
view
```

1.3.2 AccessControlFeatures

Properties

actions Array of AccessAction (string), **required**	Available actions that can be allowed or denied via permissions.
scopes Array of AccessScope (string), **required**	Available scopes that can be accessed with the different actions.

Examples

```
{
  "actions": [
    "view",
    "edit"
  ],
  "scopes": [
    "access",
    "miscellaneous",
    "network",
    "notify",
    "peripherals",
    "sensor",
    "settings",
    "system"
  ]
}
```

1.3.3 AccessPermission

A permission defines a set of allowed actions in a specific access control scope.

Properties

scope Array of AccessScope (string), **required**	Every API endpoint belongs to an access control scope. A scope combined with a number of actions forms a permission.
actions Array of AccessAction (string), **required**	

1.3.4 AccessRole

A role describes a set of permissions. Each user may belong to multiple roles.

Properties

id string, **required**		
permissions Array of AccessPermission, **required**	AccessPermission[]	
	scope AccessScope (string), **required**	Every API endpoint belongs to an access control scope. A scope combined with a number of actions forms a permission.
	actions	

	Array of AccessAction (string), **required**	

1.3.5 AccessScope

Every API endpoint belongs to an access control scope. A scope combined with a number of actions forms a permission.

Type Information

AccessScope (string)

Examples

```
network
```

1.3.6 Action

The sensor allows the connection of events with actions. Actions can be related to the sensor operations or the information handled by the sensor (e.g. the list of stored detectables).

The Action consists of a unique name and a set of optional arguments.

The list of available Actions and their possible arguments can be retrieved via `/api/actions` .

Properties

name string, **required**	Unique name of the action
arguments object, **required**	arguments

1.3.7 ActionEnableSwitchingOutput

Apply the *output_pattern* of the currently detected matcher to the switching outputs of the sensor.

Properties

name string, **required**	Unique name of the action
arguments object, **required**	arguments

1.3.8 ActionKeyLock

Change the *locked* state of the keypad. This allows or disallows local access to the sensor via the keypad.

Properties

arguments object, **required**	arguments
name string, **required**	Unique name of the action

Examples

```
{
  "name": "keylock",
  "arguments": {
    "locked": true
  }
}
```

1.3.9 ActionRemoveAllDetectables

Remove all stored Detectables belonging to any Matcher.

Properties

name string, **required**	Unique name of the action
arguments object, **required**	arguments

1.3.10 ActionRemoveAllMatchers

Remove all stored Matchers (including the related detectables).

Properties

name string, **required**	Unique name of the action
arguments object, **required**	arguments

1.3.11 ActionResultEnableSwitchingOutput

After each sampling period a Detection Result is determined based on the currently sampled color and the contents of the color storage (matchers and detectables).

In addition to the sampled color, the Detection Result includes transitions and events on all input lines during the last sample period, as well as the state of the switching outputs during the following sampling period.

Properties

uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
timestamp TimestampBackendUptime (number), **required**	The timestamp (given in microseconds) is based on the uptime of the internal analog sensor backend. It may get reset to zero under specific conditions.		
corrected_color CorrectedColor, **required**	Representation of a color in the colorspace XYZ.		
	values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace	
transformed_color TransformedColor, **required**	A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the `colorspace.axes` property of currently used detection profile.		
	values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace	
representations ColorRepresentations, **required**	Pre-calculcated visual representations of a color suitable for rendering		
	RGB Array of number, minimum items: 3, maximum items: 3, **required**	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.	
inputs InputsState, **required**	The state of all inputs during a given period is specified by a list of possible events combined with a boolean		

	value indicating, if the given event occurred within the period.		
	// boolean, **required**	The boolean value indicates whether the named input event occurred during the last period.	
detection ColorMatchingResult, **required**	After each sampling period the retrieved color value is compared to the stored detectables (color positions). Detectables are ignored, if the tolerance shape of their corresponding matcher does not encompass the current sample. Finally the closes suitable detectable is selected as the winner of the color matching operation. The corresponding matcher determines the state of the sensor for the duration of the next sampling period.		
	matcher any of UUID (string) or null, optional, Deprecated	Deprecated: use "chosen_matcher_id" instead	
	chosen_matcher_id any of UUID (string) or null, **required**	unique identifier of the selected matcher	
	distances Array of any of number or null, **required**	Distance between the sample's color position and the selected matcher's closest color position along the three axes of the color space. The array contains three 'null' values, if no suitable matcher was found for the current color sample.	
	output_pattern CurrentSwitchingOutputsState, **required**	Currently active state of the Switching Outputs. Beware that this may deviate from the specified output states of the current best matcher, since settings like *triggered input* or *hold time* influence update process for the Switching Outputs..	
		states Array of any of boolean or null, **required**	List of True/False values describing the current states of the Switching Outputs
signal_level number, **required**	Der Signalpegel zeigt die Verwendung des internen ADC-Abtastbereichs an		

Examples

```
{
  "uuid": "4575656f-efe4-4a7d-862c-9660c15cdf4e",
  "timestamp": 12455148861,
  "corrected_color": {
    "values": [
      0.419219434261322,
      0.4271118938922882,
      0.18753691017627716
    ]
  },
  "transformed_color": {
    "values": [
      78.10789489746094,
      5.271166801452637,
      -32.290863037109375
    ]
  },
  "representations": {
    "RGB": [
      0.6569499359485452,
      0.7560762577592035,
      0.9910401649653352
    ]
  },
  "inputs": {
    "trigger_0_level_low": true,
    "trigger_0_edge_falling": false,
    "trigger_0_edge_rising": false,
    "trigger_0_level_high": false
  },
  "detection": {
    "chosen_matcher_id": null,
    "distances": [
      null,
      null,
      null
    ],
    "output_pattern": {
      "states": [
        true,
        true,
        true
      ]
    }
  },
  "signal_level": 0.7
}
```

1.3.12 ActionResultKeyLock

Properties

locked boolean, **required**	New state of the keypad locking.

Examples

```
{
  "locked": true
}
```

1.3.13 ActionResultRemoveAllDetectables

The response is empty and returns HTTP status 204.

Type Information

ActionResultRemoveAllDetectables (string)

1.3.14 ActionResultRemoveAllMatchers

The response is empty and returns HTTP status 204.

Type Information

ActionResultRemoveAllDetectables (string)

1.3.15 ActionResultRunAutogain

Optional settings for customizing the Autogain procedure.

Properties

level Number, default: `0.8`, minimum: 0.01, maximum: 1, optional	Target value for the auto-gain procedure
minimum_sample_rate SampleRate (number), minimum: 0.02, optional	Desired sample rate (the default is the current sample rate)
enable_internal_emitter Boolean, default: `true`, optional	controls the power of the internal light source
enable_ambient_light_compensation Boolean, default: `true`, optional	Control the ambient light compensation procedure. This setting is only relevant if `enable_internal_emitter` is set to true. The ambient light compensation leads to a pulsed usage of the internal light emitter. Samples are collected for alternating light and dark phases. This allows to calculate a color sample of the target excluding any optical interference from external light sources. You should not disable ambient light compensation unless the optical path is perfectly isolated. Otherwise external light will inevitably interfere with the color sampling.
averages AverageSampleCount (integer), minimum: 1, optional	Number of previous samples to be averaged for every sampling result. A rolling averaging algorithm is applied to the samples.

Examples

```
{
  "level": 0.7,
  "minimum_sample_rate": 1500,
  "enable_internal_emitter": true,
  "enable_ambient_light_compensation": true
}
```

1.3.16 ActionResultTeachDetectable

A detectable represents the numeric position in a colorspace. It is connected to a *Matcher*.

Properties

uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.	
matcher_id UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	Referenz zu der Farbgruppe (Matcher), welche diese Farbe (Detectable) enthält.	
color TransformedColor, **required**	A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the `colorspace.axes` property of currently used detection profile.	
	color	
	values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace
representations ColorRepresentations, optional, read-only	Vorberechnete visuelle Darstellung einer geeigneten Farbe zur Wiedergabe	
	representations	

	RGB Array of number, minimum items: 3, maximum items: 3, **required**	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.

Examples

```
{
  "uuid": "9f968e8a-ad9c-45ce-9beb-a55011856a99",
  "alias": 2,
  "matcher_id": "1c7e9725-8753-4b6c-a0b7-a71d7e915cb5",
  "color": {
    "values": [
      0.476731,
      0.381263,
      0.128475
    ]
  },
  "representations": {
    "RGB": [
      0.396114,
      0.479113,
      0.552308
    ]
  }
}
```

1.3.17 ActionRunAutogain

Start an automatic adjustment of the optiocal sensor setup. See /api/sensor/detection-profiles/current/autogain for details.

Properties

name string, **required**	Unique name of the action
arguments object, **required**	arguments

1.3.18 ActionTeachDetectable

Add the currently sampled color as a Detectable to the selected matcher.

Properties

arguments object, **required**	arguments		
	matcher_id UUID (string), pattern: `^[a-f0-9-]+$`, optional, read-only	The new Detectable is assigned to the Matcher identified by this UUID. In case this matcher UUID (and "matcher_output_pattern") is undefined, a new matcher is created.	
	matcher_output_pattern: object, optional	Pattern of the switching outputs to be used when selecting the target matcher for the new detectable. A suitable matcher is created, if no matcher with the specified pattern is found. This field is ignored, if "matcher_id" is not null. If no pattern is defined (an no "matcher_id"), then a new matcher is created whenever the corresponding action is executed.	

		matcher_output_pattern:	
		states Array of any of boolean or null, **required**	List of True/False values describing the current states of the Switching Outputs
	remove_matcher_detectables_before boolean , default: `true`, optional	Remove all Detectables belonging to the configured Matcher before attaching the new Detectable.	
name string, **required**	Unique name of the action		

Examples

```
{
  "name": "teach_single",
  "arguments": {
    "matcher_id": "3f26aff4-8650-42a0-b319-51776c443fbc",
    "remove_matcher_detectables_before": false
  }
}
```

1.3.19 ActionTrigger

An Action Trigger assigns a given set of actions with an event.

At the end of each sample period, all events are evaluated. All corresponding actions are executed afterwards.

Properties

uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
event TriggerEventName (string), **required**	Any of the event names provided `by /api/sensor/capabilities` (`attribute trigger_sources`) is allowed.	
actions Array of Action, **required**	List of actions to be executed after the given event.	
	Action[]	
	name String, **required**	Unique name of the action
	arguments Object, **required**	arguments

Examples

```
{
  "uuid": "3f26aff4-8650-42a0-b319-51776c443fbc",
  "event": "trigger_0_edge_falling",
  "actions": [
    {
      "name": "enable_switching_output",
      "arguments": {}
    }
  ]
}
```

1.3.20 Alias

A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.

Type Information

Alias (integer)

Examples

4

1.3.21 AmplificationLevel

The amplification level specifies the internal configuration of an amplifier. This value is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).

Type Information

AmplificationLevel (integer)

1.3.22 AnyAction

Type Information

any of ActionEnableSwitchingOutput, ActionTeachDetectable, ActionKeyLock, ActionRunAutogain, ActionRemoveAllDetectables or ActionRemoveAllMatchers

1.3.23 AnyActionResult

Type Information

any of ActionResultEnableSwitchingOutput, ActionResultTeachDetectable, ActionResultKeyLock, ActionResultRunAutogain, ActionResultRemoveAllDetectables (string) or ActionResultRemoveAllMatchers (string)

1.3.24 AutogainSettings

Optional settings for customizing the Autogain procedure.

Properties

level Number, default: `0.8`, minimum: 0.01, maximum: 1, optional	Target value for the auto-gain procedure
minimum_sample_rate SampleRate (number), minimum: 0.02, optional	Desired sample rate (the default is the current sample rate)
enable_internal_emitter Boolean, default: `true`, optional	controls the power of the internal light source
enable_ambient_light_compensation Boolean, default: `true`, optional	Control the ambient light compensation procedure. This setting is only relevant if `enable_internal_emitter` is set to true. The ambient light compensation leads to a pulsed usage of the internal light emitter. Samples are collected for alternating light and dark phases. This allows to calculate a color sample of the target excluding any optical interference from external light sources. You should not disable ambient light compensation unless the optical path is perfectly isolated. Otherwise external light will inevitably interfere with the color sampling.

averages AverageSampleCount (integer), minimum: 1, optional	Number of previous samples to be averaged for every sampling result. A rolling averaging algorithm is applied to the samples.

Examples

```
{
  "level": 0.7,
  "minimum_sample_rate": 1500,
  "enable_internal_emitter": true,
  "enable_ambient_light_compensation": true
}
```

1.3.25 AverageSampleCount

Number of previous samples to be averaged for every sampling result. A rolling averaging algorithm is applied to the samples.

Type Information

AverageSampleCount (integer), minimum: 1

1.3.26 BaseColorTolerance

Properties

shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.
limits object, **required**	limits

1.3.27 BaseSerialSettings

Properties

Type string, one of [`none, eliza, modbus`] , default: eliza, **required**

1.3.28 BoxColorTolerance

Properties

limits Object, **required**	limits **half_edges** Array of number, minimum items: 3, maximum items: 3, **required**
shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.

1.3.29 ChromaticityCoordinate

Location in a colorspace

Type Information

Array of number, minimum items: 3, maximum items: 3

1.3.30 ColorDetectable

A detectable represents the numeric position in a colorspace. It is connected to a *Matcher*.

Properties

uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.	
matcher_id UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	Referenz zu der Farbgruppe (Matcher), welche diese Farbe (Detectable) enthält.	
color TransformedColor, **required**	A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the `colorspace.axes` property of currently used detection profile.	
	color	
	values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace
representations ColorRepresentations, optional, read-only	Pre-calculcated visual representations of a color suitable for rendering	
	representations	
	RGB Array of number, minimum items: 3, maximum items: 3, **required**	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.

Examples

```
{
  "uuid": "9f968e8a-ad9c-45ce-9beb-a55011856a99",
  "alias": 2,
  "matcher_id": "1c7e9725-8753-4b6c-a0b7-a71d7e915cb5",
  "color": {
    "values": [
      0.476731,
      0.381263,
      0.128475
    ]
  },
  "representations": {
    "RGB": [
      0.396114,
      0.479113,
      0.552308
    ]
  }
}
```

1.3.31 ColorDetectionResult

After each sampling period a Detection Result is determined based on the currently sampled color and the contents of the color storage (matchers and detectables).

In addition to the sampled color, the Detection Result includes transitions and events on all input lines during the last sample period, as well as the state of the switching outputs during the following sampling period.

Properties

uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		

timestamp TimestampBackendUptime (number), **required**	The timestamp (given in microseconds) is based on the uptime of the internal analog sensor backend. It may get reset to zero under specific conditions.		
corrected_color CorrectedColor, **required**	Representation of a color in the colorspace XYZ.		
	values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace	
transformed_color TransformedColor, **required**	A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the `colorspace.axes` property of currently used detection profile.		
	values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace	
representations ColorRepresentations, **required**	Pre-calculcated visual representations of a color suitable for rendering		
	RGB Array of number, minimum items: 3, maximum items: 3, **required**	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.	
inputs InputsState, **required**	The state of all inputs during a given period is specified by a list of possible events combined with a boolean value indicating, if the given event occurred within the period.		
	// boolean, **required**	The boolean value indicates whether the named input event occurred during the last period.	
detection ColorMatchingResult, **required**	After each sampling period the retrieved color value is compared to the stored detectables (color positions). Detectables are ignored, if the tolerance shape of their corresponding matcher does not encompass the current sample. Finally the closes suitable detectable is selected as the winner of the color matching operation. The corresponding matcher determines the state of the sensor for the duration of the next sampling period.		
	matcher any of UUID (string) or null, optional, Deprecated	Deprecated: use "chosen_matcher_id" instead	
	chosen_matcher_id any of UUID (string) or null, **required**	unique identifier of the selected matcher	
	distances Array of any of number or null, **required**	Distance between the sample's color position and the selected matcher's closest color position along the three axes of the color space. The array contains three 'null' values, if no suitable	

		matcher was found for the current color sample.	
	output_pattern CurrentSwitchingOutputsState, **required**	Currently active state of the Switching Outputs. Beware that this may deviate from the specified output states of the current best matcher, since settings like *triggered input* or *hold time* influence update process for the Switching Outputs.	
		states Array of any of boolean or null, **required**	List of True/False values describing the current states of the Switching Outputs
signal_level number, **required**	The signal level indicates the usage of the internal ADC sampling range. This		

Examples

```
{
  "uuid": "4575656f-efe4-4a7d-862c-9660c15cdf4e",
  "timestamp": 12455148861,
  "corrected_color": {
    "values": [
      0.419219434261322,
      0.4271118938922882,
      0.18753691017627716
    ]
  },
  "transformed_color": {
    "values": [
      78.10789489746094,
      5.271166801452637,
      -32.290863037109375
    ]
  },
  "representations": {
    "RGB": [
      0.6569499359485452,
      0.7560762577592035,
      0.9910401649653352
    ]
  },
  "inputs": {
    "trigger_0_level_low": true,
    "trigger_0_edge_falling": false,
    "trigger_0_edge_rising": false,
    "trigger_0_level_high": false
  },
  "detection": {
    "chosen_matcher_id": null,
    "distances": [
      null,
      null,
      null
    ],
    "output_pattern": {
      "states": [
        true,
        true,
        true
      ]
    }
  },
  "signal_level": 0.7
}
```

1.3.32 ColorDetectionResultList

Type Information

Array of ColorDetectionResult

1.3.33 ColorDetectionResultOrNil

Type Information

any of ColorDetectionResult or null

1.3.34 ColorMatcher

A matcher represents a distinguished detection result and the wanted behaviour of the sensor whenever it is encountered.

Properties

uuid UUID (string), pattern: ^[a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.	
name String, **required**	human-readable name of the matcher	
tolerance Any of InfiniteColorTolerance, SphereColorTolerance, CylinderColorTolerance or BoxColorTolerance, **required**	Specification of a geometric shape and its dimensions in the current colorspaces.	
	InfiniteColorTolerance	
	limits Object, **required**	limits
	shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.
	SphereColorTolerance	
	limits Object, **required**	limits **radius** Numer, **required**
	shape ToleranceShapeName (string), required	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.
	CylinderColorTolerance	
	limits Object, **required**	limits **radius** Number, **required** **half_height** Number, **required**

	shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities.`
	BoxColorTolerance	
	limits Object, **required**	limits **half_edges** Array of number, minimum items: 3, maximum items: 3, **required**
	shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities.`
output_pattern WantedSwitchingOutputsState, **required**	The combination of tristate values describes a logical state of the switching outputs of the sensor. The states true or false cause the output to go up or down. The state null keeps the previous state of the output unchanged.	
	uuid UUID (string), pattern: ^[a-f0-9-]+$, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
	states Array of any of boolean or null, **required**	List of True/False/Null values describing the wanted states of the Switching Outputs
hold_time HoldTime (number), maximum: 3153600000, **required**	Minimum duration (in seconds) of a matcher's output setup being applied after detection.	
reset_output_after_hold_time_expired Boolean, default: false, **required**	Controls if the output should be reset after the hold time passed. This is helpful if you only sample by triggering inputs and wish to reset the outputs afterwards.	
signal_color Any of string or null, **required**	A custom color name. How and what color will be displayed is defined by the client.	

Examples

```
{
  "uuid": "9ffaa31f-8011-44f5-bb2a-f91e4be50764",
  "alias": 6,
  "name": "clean bottle cap",
  "tolerance": {
    "limits": {
      "radius": 2,
      "half_height": 4
    },
    "shape": "cylinder"
  },
  "output_pattern": {
    "uuid": "1adc74e2-96ac-4761-b9e6-2d93e02d9244",
    "states": [
      true,
      false,
      false
    ]
  },
  "hold_time": 0,
  "reset_output_after_hold_time_expired": false,
  "signal_color": null
}
```

1.3.35 ColorMatchingResult

After each sampling period the retrieved color value is compared to the stored detectables (color positions). Detectables are ignored, if the tolerance shape of their corresponding matcher does not encompass the current sample. Finally the closes suitable detectable is selected as the winner of the color matching operation. The corresponding matcher determines the state of the sensor for the duration of the next sampling period.

Properties

matcher any of UUID (string) or null, optional, Deprecated	Deprecated: use "chosen_matcher_id" instead	
chosen_matcher_id any of UUID (string) or null, **required**	unique identifier of the selected matcher	
distances Array of any of number or null, **required**	Distance between the sample's color position and the selected matcher's closest color position along the three axes of the color space. The array contains three 'null' values, if no suitable matcher was found for the current color sample.	
output_pattern CurrentSwitchingOutputsState, **required**	Currently active state of the Switching Outputs. Beware that this may deviate from the specified output states of the current best matcher, since settings like *triggered input* or *hold time* influence update process for the Switching Outputs.	
	states Array of any of boolean or null, **required**	List of True/False values describing the current states of the Switching Outputs

Examples

No Match

```
{
  "chosen_matcher_id": null,
  "distances": [
    null,
    null,
    null
  ],
  "output_pattern": {
    "states": [
      true,
      true,
      true
    ]
  }
}
```

Suitable Match

```
{
  "chosen_matcher_id": "4575656f-efe4-4a7d-862c-9660c15cdf4e",
  "distances": [
    1.4,
    0.3,
    null
  ],
  "output_pattern": {
    "states": [
      true,
      false,
      true
    ]
  }
}
```

1.3.36 ColorRepresentations

Pre-calculcated visual representations of a color suitable for rendering

Properties

RGB Array of number, minimum items: 3, maximum items: 3, **required**	RGB color array representing the axes r, g, and b in that order. Values are floats between 0 and 1.

Examples

```
{
  "RGB": [
    0.3197475,
    0.754686,
    0.216748
  ]
}
```

1.3.37 Colorspace

A colorspace describes the numeric conversion of colors under certain circumstances. Different standardized colorspaces are suitable for different detection tasks.

Properties

name String, **required**		
space_id ColorspaceID, **required**	Unique name of a colorspace	
axes Array of ColorspaceAxis, minimum items: 3, maximum items: 3, **required**	ColorspaceAxis[]	
	id String, **required**	Unique name
	label String, **required**	Human-readable name
	minimum Number, **required**	lowest expected value of a color along this axis under usual circumstances
	maximum Number, **required**	highest expected value of a color along this axis under usual circumstances

Examples

```
{
  "name": "L*a*b*",
  "space_id": "Lab",
  "axes": [
    {
      "id": "L",
      "label": "L*",
      "minimum": 0,
      "maximum": 100
    },
    {
      "id": "a",
      "label": "a*",
      "minimum": -500,
      "maximum": 500
    },
    {
      "id": "b",
      "label": "b*",
      "minimum": -200,
      "maximum": 200
    }
  ]
}
```

1.3.38 ColorspaceAxis

Properties

id String, **required**	Unique name
label String, **required**	Human-readable name
minimum Number, **required**	lowest expected value of a color along this axis under usual circumstances
maximum Number, **required**	highest expected value of a color along this axis under usual circumstances

1.3.39 ColorspaceID

Unique name of a colorspace.

Type Information

ColorspaceID (string)

1.3.40 ColorspaceToleranceMap

Specify the usage of the axes of each colorspace for non-trivial tolerance shapes. See "colorspace_tolerance_maps" below "/capabilities" for more details.

Properties

colorspace_id ColorspaceID (string), **required**	Unique name of a colorspace
tolerance_shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.
limits_axes_map Object, **required**	limits_axes_map
	half_height Array of string, optional
	half_edges Array of string, optional
	radius Array of string, optional

Examples

```
{
  "colorspace_id": "Lab",
  "tolerance_shape": "cylindrical",
  "limits_axes_map": {
    "half_height": [
      "L"
    ],
    "radius": [
      "a",
      "b"
    ]
  }
}
```

1.3.41 ColorTolerance

Specification of a geometric shape and its dimensions in the current colorspaces.

Type Information

any of InfiniteColorTolerance, SphereColorTolerance, CylinderColorTolerance or BoxColorTolerance

Examples

Infinite

```
{
  "shape": "infinite",
  "limits": {}
}
```

Sphere

```
{
  "shape": "sphere",
  "limits": {
    "radius": 2
  }
}
```

Cylidner

```
{
  "shape": "cylinder",
  "limits": {
    "radius": 2,
    "half_height": 4
  }
}
```

Box

```
{
  "shape": "box",
  "limits": {
    "half_edges": [
      4,
      2,
      2
    ]
  }
}
```

1.3.42 CompensationSettings

The compensation settings of a Detection Profile describe the configuration of internal sensor components related to the stabilization and compensation algorithms.

These values can be determined by issuing a POST request against `/api/sensor/detection-profiles/current/autogain`. The result is a suitable set of compensation settings for this sensor under the current circumstances.

The content of this data object is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).

1.3.43 CorrectedColor

Representation of a color in the colorspace XYZ.

Properties

values Array of number, minimum items: 3, maximum items: 3, **required**	Location in a colorspace

1.3.44 CurrentDetectionProfileID

The sensor can store multiple Detection Profiles, but it can only apply one at a time. The field `current_profile_id` contains the UUID of the Detection Profile that is currently used by the sensor for its operation. It allows to use the shortcut API endpoint `/api/sensor/detection-profiles/current` instead of specifying a Detection Profile by its UUID.

Type Information

CurrentDetectionProfileID (string), pattern: `^[a-f0-9-]+$`

Examples

```
a014e415-0fec-4734-ac3f-30da0a5f3899
```

1.3.45 CurrentSwitchingOutputsState

Currently active state of the Switching Outputs. Beware that this may deviate from the specified output states of the current best matcher, since settings like *triggered input* or *hold time* influence update process for the Switching Outputs.

Properties

states Array of any of boolean or null, **required**	List of True/False values describing the current states of the Switching Outputs

1.3.46 CylinderColorTolerance

Properties

limits object, **required**	limits
	radius number, **required**
	half_height number, **required**
shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.

1.3.47 Standardwertepaare

Properties

uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
object_type string, **required**, read-only	Name of the object the default is meant for
key string, **required**, read-only	name of the object's property
value any, **required**	Actual default value for the object's property

Examples

Matcher: Tolerance

```
{
  "uuid": "a7bd36b3-e9c1-4f60-8d7e-cf47634a28b1",
  "object_type": "matcher",
  "key": "tolerance",
  "value": {
    "shape": "sphere",
    "limits": {
      "radius": 4
    }
  }
}
```

Matcher: Hold Time

```
{
  "uuid": "55b35901-1ea6-4b3d-864a-60af15a9b0c5",
  "object_type": "matcher",
  "key": "hold_time",
  "value": 0
}
```

Matcher: reset output after Hod Time expiry

```
{
  "uuid": "9ba8a7a4-7fa5-4bfc-8883-98d7b6084e91",
  "object_type": "matcher",
  "key": "reset_output_after_hold_time_expired",
  "value": false
}
```

Autogain: number of samples used for averaging

```
{
  "uuid": "eeb46031-10e5-4f13-901a-c7eb16aa0cf9",
  "object_type": "autogain",
  "key": "averages",
  "value": 0
}
```

1.3.48 DetectionProfile

A Detection Profile contains a complete set of sensor settings for a given detection task.

Multiple profiles can be stored in order to switch easily between different detection tasks or for the incremental development of a refined profile.

Some attributes of a Detection Profile expose internal details of the sensor that should be determined indirectly via other means. These attributes are described only superficially, since they should be handled *as is* without changing their value or structure.

Properties

uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8		
alias Alias (integer), **required**, read-only	A numerical value that can be used to address an item in a collection. If an alias is specified		

	alongside an uuid attribute, that alias can be used as an alternative to address the item in URLs and other protocols like Modbus or serial interfaces.		
name String, **required**	Human-readable name of the Detection Profile		
colorspace Colorspace, **required**	A colorspace describes the numeric conversion of colors under certain circumstances. Different standardized colorspaces are suitable for different detection tasks.		
	colorspace		
	name String, **required**		
	space_id ColorspaceID, **required**	Unique name of a colorspace	
	axes Array of ColorspaceAxis, minimum items: 3, maximum items: 3, **required**	ColorspaceAxis[]	
		id String, **required**	Unique name
		label String, **required**	Human-readable name
		minimum Number, **required**	lowest expected value of a color along this axis under usual circumstances
		maximum Number, **required**	highest expected value of a color along this axis under usual circumstances
non_matching_output WantedSwitchingOutputsState, **required**	This state of the Switching Outputs is applied, if the currently sample color does not belong to any of the stored *Matchers*.		
	non_matching_output		
	uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8	
	states Array of any of boolean or null, **required**	List of True/False/Null values describing the wanted states of the Switching Outputs	
non_matching_hold_time HoldTime (number), maximum 3153600000, **required**	Minimum duration (in seconds) of the *non_matching_output* state being applied to the Switching Outputs of the sensor. This prolonging of a potential *non matching* event may be useful, if the processing period of a connected actor exceeds the sampling period of the sensor.		
compensation_settings CompensationSettings, **required**	The compensation settings of a Detection Profile describe the configuration of internal sensor components related to the stabilization and compensation algorithms. These values can be determined by issuing a POST request against /api/sensor/detection-profiles/current/autogain. The result is a suitable set of compensation settings for this sensor under the current circumstances. The content of this data object is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted		

	and applied without modification or introspection).		
	compensation_settings		
sampling_settings SamplingSettings, **required**	Sampling Settings describe all details of the sampling process. Its attributes may be queried and inspected (e.g. in order to retrieve the current sample rate). Most values stored within the Sampling Settings should not be modified directly. The related API endpoint `/api/sensor/detection-profiles/current/autogain` should be used instead. The only modifiable attribute within the Sampling Settings is the *averages* value. It is safe to change it, even though the default values calculated during an autogain operation should be optimal for most detection tasks.		
	sampling_settings		
	led_intensity Number, minimum: 0, maximum: 1, **required**	relative intensity of the internal emitter during the light phase	
	base_sample_rate SampleRate (number), minimum: 0.01, **required**	The base sample rate determines the duration of a sampling period. After each sampling period, the gathered data is processed and a new detection result is calculated (e.g. the most suitable *Matcher* for the given sample). This may affect the state of the Switching Outputs or trigger configured actions. Thus the base sample rate defines the maximum rate of changes for the Switching Outputs. See also the *effective sample rate*.	
	effective_sample_rate SampleRate (number), minimum: 0.01, **required**	The effective sample rate is the numeric product of the *base sample rate* and the number of *averages*. It determines the minimum duration that a target needs to be sampled in order to determine its visual appearance correctly. With the default value of *average* set to one, this value is equal to the base sample rate.	
	minimum_wanted_sample_rate SampleRate (number), minimum: 0.01, **required**	This informational value represents the sample rate that was requested during the most recent *Autogain* operation. The effective sample rate may deviate from the wanted sample rate, if the requested sample rate was not achievable due to limitations of the sensor (e.g. exceeding the supported sample rate) or due to the environment (e.g. not enough light, thus a slower amplification with higher gain was necessary).	
	sample_light_phase Boolean, **required**	defines if the sensor should periodically activate the internal emitter for sampling	

	sample_dark_phase Boolean, **required**	defines if the sensor should periodically deactivate the internal emitter for sampling	
	averages AverageSampleCount (integer), minimum: 1, **required**	Number of previous samples to be averaged for every sampling result. A rolling averaging algorithm is applied to the samples.	
	amplification AmplificationLevel (integer), **required**	The amplification level specifies the internal configuration of an amplifier. This value is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).	
white_reference Array of number, **required**	The White Reference attribute is used for indicating a custom color balancing. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference`).		
normalization_constant Array of number, **required**	Normalization constants are related to the White Reference. Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference`).		

Examples

```
{
  "name": "#0",
  "uuid": "2475df8d-85f0-4208-ba60-dce6cb282a96",
  "alias": 1,
  "non_matching_hold_time": 0,
  "colorspace": {
    "name": "L*a*b*",
    "axes": [
      {
        "id": "L",
        "label": "L*",
        "minimum": 0,
        "maximum": 100
      },
      {
        "id": "a",
        "label": "a*",
        "minimum": -500,
        "maximum": 500
      },
      {
        "id": "b",
        "label": "b*",
        "minimum": -200,
        "maximum": 200
      }
    ],
    "space_id": "Lab"
  },
  "compensation_settings": {
    "monitor_integration": {
      "control": 0.324999988079071O4,
      "references": [
        0.7283520102500916,
        0.7442666888237,
        0.7066696286201477
      ]
    },
    "use_calibration_samples": true
  },
  "normalization_constant": [
    237.4935277662995,
    242.62655153828055,
    587.8264132734112
  ],
  "white_reference": [
    95.047,
    100,
    108.883
  ],
  "non_matching_output": {
    "uuid": "3f26aff4-8650-42a0-b319-51776c443fbc",
    "states": [
      true,
      true,
      true,
      true,
      true,
      true,
      true,
      true
    ]
  },
  "sampling_settings": {
    "led_intensity": 1,
    "amplification": 1,
    "sample_light_phase": true,
    "minimum_wanted_sample_rate": 1000,
    "averages": 1,
    "base_sample_rate": 1000,
    "sample_dark_phase": true,
    "effective_sample_rate": 1000
  }
}
```

1.3.49 DeviceInformation

Properties

id DeviceSerialNumber, **required**	Serial Number
model_name string, **required**	human-readable name of the device model
model_key string, **required**	unique id of the device model
variant any of string or null, **required**	indicates a special series of a model
vendor_key DeviceVendorKey, **required**	Unique key identifying the organization distributing this device
vendor_name DeviceVendorName, **required**	Name of vendor of this device
device_id DeviceSerialNumber, optional, Deprecated	Deprecated: use "id" instead.

model string, optional, Deprecated	Deprecated: use "model_name" instead.
vendor DeviceVendorName, optional, Deprecated	Deprecated: use "vendor_name" instead.

Examples

```
{
  "vendor_name": "Micro-Epsilon Eltrotec GmbH",
  "vendor_key": "eltrotec",
  "variant": null,
  "model_key": "me_cfo_100",
  "model_name": "CFO100",
  "id": "7454228060"
}
```

1.3.50 DeviceSerialNumber

Serial Number.

Type Information

DeviceSerialNumber (string)

1.3.51 DeviceVendorKey

Unique key identifying the organization distributing this device.

Type Information

DeviceVendorKey (string)

Examples

```
acme
```

1.3.52 DeviceVendorName

Name of vendor of this device.

Type Information

DeviceVendorName (string)

Examples

```
Acme Corporation
```

1.3.53 Error

List of error indicators that are both machine-parseable and human-readable

Properties

code string, optional	machine-readable unique error code
mapping string, optional	a reference to the parameter that caused the error
message string, optional	human-readable error description

1.3.54 FirmwareBuildId

Unique ID of the currently running firmware image.

Type Information

FirmwareBuildId (string), pattern: `^[a-f0-9-]+$`

Examples

```
d985c28e03a4eb39132c02affeb29e71
```

1.3.55 FirmwareImageFile

Type Information

FirmwareImageFile (file)

1.3.56 FirmwareImageSize

Size of the firmware image in bytes.

Type Information

FirmwareImageSize (integer), minimum: 1, maximum: 1073741824

1.3.57 FirmwareImageUpload

A fully or partially uploaded firmware image to be used for upgrading the firmware.

Properties

uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	Unique ID of a firmware upload
build_id HashDigest (string), pattern: `^[a-f0-9]+$`, **required**	unique ID of the currently running firmware image
status string, one of `[incomplete, complete, invalid_signature, processing_failure, malformed_content, device_mismatch]`, **required**	Current status of the firmware upload incomplete the number of bytes received is lower than the number of bytes that have been announced complete the firmware upload is complete and the new firmware can be applied invalid_signature the firmware checksum didn’t match the expected value processing_failure an internal undefined error occurred while processing the firmware malformed_content the uploaded firmware image uses an unexpected format or misses essential information device_mismatch the firmware image can not be applied to this device
uploaded_size integer, minimum: 0, **required**	number of uploaded bytes
expected_size integer, minimum: 1, **required**	expected total number of bytes for the firmware image
max_chunk_size integer, minimum: 1, **required**	maximum size for a data chunk uploaded to the device

Examples

```
{
  "uuid": "78b40d5e-e82c-45a9-8842-9481f889f790",
  "build_id": "e943ce84dbe474bc4d163b44c90070b105fd66bb",
  "expected_size": 335544320,
  "max_chunk_size": 1048576,
  "status": "incomplete",
  "uploaded_size": 24117248
}
```

1.3.58 FirmwareInformation

Information describing a firmware version.

Properties

id FirmwareBuildId (string), pattern: ^[a-f0-9]+$, **required**	unique ID of the currently running firmware image
channel ReleaseChannel (string), one of [stable, feature] , default: stable, **required**	Describes the kind of a publication Releases on the `stable` channel are generally considered well-tested and are recommended for use in production. Releases on the `feature` add new features but haven't been tested as much as a stable release. Feature releases can but should only be used in production with careful consideration.
created_on Timestamp (string), **required**	time this firmware build was created
name string, **required**	human-readable name of this release
notes string, **required**	Release notes formatted as markdown
version FirmwareVersion (string), **required**	version of a firmware
works_with Array of string, **required**	compatible device models (see `model_key` in `/api/device`)

1.3.59 FirmwareRecoveryInformation

Type Information

FirmwareRecoveryInformation (string)

Examples

```
{
  "created_on": "2018-02-13T05:40:39+01:00",
  "name": "CFO",
  "id": "4fab356b5014b5cc82efc4a81bfefbfcdc2d9165",
  "version": "1.3.16",
  "channel": "stable",
  "works_with": [
    "me_cfo_100",
    "me_cfo_200"
  ],
 "notes": "# Release 1.3.16 (2018-02-13 - CFO)\n\n## Veröffentlichungshinweise\n\nWartungsrelease für CFO-Sensoren.\n\n\n## Änderungen\nkeine\n\n\n## Fehlerkorrekturen\n* Hochladen von Konfigurationsdateien mit mehr als 70 Farben ermöglicht\n* Announcierung des korrekten Hostnamen via avahi/zeroconf\n* SSDP: Kommunikation via IPv6-Link-Local-Adresse ermöglicht\n* SSDP: auch die Auto-Konfigurations-IP (via RFC3927) unter \"CurrentAddresses\" announcieren\n* SSDP: nach Konfigurationsänderungen an neue IP-Addressen binden"
}
```

1.3.60 FirmwareRunningInformation

Information describing the currently running firmware.

Properties

build_id FirmwareBuildId (string), pattern: `^[a-f0-9]+$`, **required**	unique ID of the currently running firmware image

source_url any of string or null, optional	Absolute base URL of a firmware repository delivering firmware images suitable for this device
version FirmwareVersion (string), **required**	version of a firmware

Examples

```
{
  "build_id": "4fab356b5014b5cc82efc4a81bfefbfcdc2d9165",
  "source_url": null,
  "version": "1.3.16"
}
```

1.3.61 FirmwareSettings

Settings related to the device's firmware and upgrades.

Properties

channel ReleaseChannel (string), one of [stable, feature] , default: stable, **required**	Describes the kind of a publication Releases on the `stable` channel are generally considered well-tested and are recommended for use in production. Releases on the `feature` add new features but haven't been tested as much as a stable release. Feature releases can but should only be used in production with careful consideration.

Examples

```
{
  "release_channel": "stable"
}
```

1.3.62 FirmwareVersion

Version of a firmware

Type Information

FirmwareVersion (string)

Examples

```
v2.3.42
```

1.3.63 HashDigest

Unique identifier (hexadecimal digest string).

Type Information

HashDigest (string), pattern: `^[a-f0-9]+$`

Examples

```
d985c28e03a4eb39132c02affeb29e71
```

1.3.64 HoldTime

Minimum duration (in seconds) of a matcher's output setup being applied after detection.

Type Information

HoldTime (number), minimum: 0, maximum: 3153600000

1.3.65 Hostname

Human-readable name identifying the device in the network.

Type Information

Hostname (string), pattern: `^(?:[a-zA-Z0-9](?:[a-zA-Z0-9\-]*[a-zA-Z0-9])?\.)*[a-zA-Z0-9](?:[a-zA-Z0-9\-]*[a-zA-Z0-9])?$`

1.3.66 InfiniteColorTolerance

Properties

limits Object, **required**	limits
shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.

1.3.67 InputsState

The state of all inputs during a given period is specified by a list of possible events combined with a boolean value indicating, if the given event occurred within the period.

Properties

// boolean, **required**	The boolean value indicates whether the named input event occurred during the last period.

1.3.68 InterfaceRS232

Properties

protocol any of SerialModbusSettings or SerialElizaSettings, **required**	SerialModbusSettings
	type string, one of [`none, eliza, modbus`] , default: `eliza`, **required**
	slave_id any of number or null, **required**
	frame_format string, one of [`rtu, ascii`] , default: `rtu`, **required**
	SerialElizaSettings
	type string, one of [`none, eliza, modbus`] , default: `eliza`, **required**
baud_rate number, one of [`9600, 19200, 115200`], **required**	

1.3.69 InterfaceUSB

Properties

protocol any of SerialModbusSettings or SerialElizaSettings, **required**	SerialModbusSettings
	type string, one of [`none, eliza, modbus`] , default: `eliza`, **required**
	slave_id any of number or null, **required**
	frame_format string, one of [`rtu, ascii`] , default: `rtu`, **required**
	SerialElizaSettings
	type string, one of [`none, eliza, modbus`] , default: `eliza`, **required**
baud_rate number, one of [`9600, 19200, 115200`], **required**	

1.3.70 KeypadEvent

A keypad event represents a single press or release event of a button at a specific time.

Properties

source string, **required**	The usual source of events is *inputs*.
name KeypadEventInput (string), **required**	Name of a keypad input (button) that may trigger events.
event KeypadEventName (string), **required**	Input peripherals can trigger different events.
timestamp integer, minimum: 0, **required**	The timestamp is given in milliseconds and should be monotonic increasing.

Examples

Intensity Button pressed

```
{
  "source": "inputs",
  "name": "intensity",
  "event": "down",
  "timestamp": 6403500
}
```

Intensity Button released

```
{
  "source": "inputs",
  "name": "intensity",
  "event": "up",
  "timestamp": 6405800
}
```

1.3.71 KeypadEventInput

Name of a keypad input (button) that may trigger events.

Type Information

KeypadEventInput (string)

1.3.72 KeypadEventName

Input peripherals can trigger different events.

Type Information

KeypadEventName (string)

1.3.73 KeypadIndicator

The keypad features multiple LEDs as visual indicators.

The indicators may be lit, blinking or off.

Properties

name string, **required**	Name of the indicator	

type string, **required**	The type describes the possible modes of visualization for this indicator.	
animation Array of object, **required**	The visual state of each indicator is described by an infinite loop of animation steps.	
	object []	
	enabled boolean, **required**	Visual status of the indicator (on or off)
	color string, optional	Name or description of a color
	duration number, **required**	Duration (in seconds) of this part of the looping animation.

Examples

```
{
  "name": "trigger_teach",
  "type": "colored",
  "animation": [
    {
      "enabled": true,
      "color": "green",
      "duration": 0.6
    },
    {
      "enabled": false,
      "duration": 0.4
    }
  ]
}
```

1.3.74 KeypadInformation

Describe the current state of the keypad as well as access to visualization data.

Properties

locked boolean, **required**	Boolean flag indicating the state of the key lock (true -> locked, false -> unlocked). All keypad inputs are ignored while the lock is active.
clear_matcher_before_teach boolean, **required**	The boolean flag controls whether multiple detectables can be stored for a matcher via keypad-based teach operations. A value of true implies, that a teach operation always removes all existing detectables from the currently selected matcher before adding the new detectable. With a value of false previously existing detectables are not deleted before a new one is added.
visualization_url any of string or null, optional, read-only	The visualization resource location can be used for providing a virtual keypad interface. Its URL may start with a scheme (e.g. *http* or *https*) for a full URL including hostname or it may start with a slash, indicating a path provided by the device itself. This attribute cannot be modified.

Examples

```
{
  "locked": true,
  "clear_matcher_before_teach": false,
  "visualization_url": "/media/keypad-image.svg"
}
```

1.3.75 KeypadInputButton

The keypad contains several *inputs* (buttons) that may generate events.

Properties

name	Name of a keypad input (button) that may trigger events.	

KeypadEventInput (string), **required**		
capabilities Array of object, **required**	object[]	
	name KeypadEventName (string), **required**	Input peripherals can trigger different events.
	url string, **required**	The event can be triggered externally by submitting a POST request against this resource.

Examples

```
{
  "name": "intensity",
  "capabilities": [
    {
      "name": "down",
      "url": "/api/peripherals/keypad/inputs/intensity/down"
    },
    {
      "name": "up",
      "url": "/api/peripherals/keypad/inputs/intensity/up"
    }
  ]
}
```

1.3.76 LoginInformation

Describes the currently active login provided by the user agent.

Properties

logged_in_user any of User or null, **required**	The currently logged in user. Is `null` if the credentials didn't match any known user or have expired.	
	User	
	name string, pattern: `^[\w-]+$`, **required**, read-only	unique name identifying an account
	password string, optional	Password assigned to this account (only writable; never returned in responses). Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
	password_hash HashDigest (string), pattern: `^[a-f0-9]+$`, optional	Password hash assigned to this account. Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
	roles Array of string, optional	The roles assigned to an account define its set of permissions.
session_timeout any of number or null, **required**	Number of seconds this session has left before expiring. Is `null` if the provided credentials could not be matched to any active sessions, if the session expired or if the supplied authentication mechanism does not support sessions (e.g. HTTP Authentication).	

1.3.77 MacAddress

Unique hardware address of a network interface.

Type Information

MacAddress (string), pattern: `^([a-f0-9]{2}:){5}[a-f0-9]{2}$`

Examples

```
00:01:2e:7a:dc:23
```

1.3.78 NetworkAddressConfigurationIPv4

Properties

method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.

1.3.79 NetworkAddressConfigurationIPv4DHCP

The Dynamic Host Configuration Protocol requires a router distributing leases on request.

Properties

method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.

1.3.80 NetworkAddressConfigurationIPv4Static

Static address configuration does not depend on network infrastructure.

Properties

method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.
address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation
gateway NetworkAddressIPv4 (string), optional	default gateway for outgoing traffic

1.3.81 NetworkAddressConfigurationIPv6

Properties

method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.

1.3.82 NetworkAddressConfigurationIPv6Auto

Stateless address autoconfiguration (SLAAC) solely relies on the Neighbourhood Discovery Protocol. SLAAC is only available for IPv6.

Properties

method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.

1.3.83 NetworkAddressConfigurationIPv6DHCP

The Dynamic Host Configuration Protocol requires a router distributing leases on request.

Properties

method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.

1.3.84 NetworkAddressConfigurationIPv6Static

Static address configuration does not depend on network infrastructure.

Properties

method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.
address NetworkInterfaceAddressIPv6 (string), **required**	IPv6 network address in CIDR notation

gateway NetworkAddressIPv4 (string), optional	default gateway for outgoing traffic

1.3.85 NetworkAddressIPv4

Type Information

NetworkAddressIPv4 (string)

Examples

```
192.168.1.100
```

1.3.86 NetworkAddressIPv6

Type Information

NetworkAddressIPv6 (string)

Examples

```
fda0:576b:c643:100:40f:10ff:fe02:e6f
```

1.3.87 NetworkInterfaceAddressConfigurationInformation

Properties

ipv4 NetworkInterfaceAddressFamilyInformationIPv4, **required**	IPv4 Network address configuration		
	ipv4		
	address_configurations Array of any of NetworkAddressConfigurationIPv4Static or NetworkAddressConfigurationIPv4DHCP, optional	NetworkAddressConfigurationIPv4Static[]	
		method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.
		address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation
		gateway NetworkAddressIPv4 (string), optional	default gateway for outgoing traffic
		NetworkAddressConfigurationIPv4DHCP[]	
		method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.
	current_addresses Array of WrappedNetworkInterfaceAddressIPv4, **required**	WrappedNetworkInterfaceAddressIPv4[]	
		address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation
ipv6 NetworkInterfaceAddressFamilyInformationIPv6, **required**	IPv6 Network address configuration		
	ipv6		

	address_configurations Array of any of NetworkAddressConfigurationIPv6Static, NetworkAddressConfigurationIPv6DHCP or NetworkAddressConfigurationIPv6Auto, optional	NetworkAddressConfigurationIPv6Static[]	
		method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.
		address NetworkInterfaceAddressIPv6 (string), **required**	IPv6 network address in CIDR notation
		gateway NetworkAddressIPv6 (string), optional	default gateway for outgoing traffic
		NetworkAddressConfigurationIPv6DHCP[]	
		method string, one of [`static`, `dhcp`, `auto`], **required**	Configuration method used for the address.
		NetworkAddressConfigurationIPv6Auto[]	
		method string, one of [`static`, `dhcp`, `auto`], **required**	Configuration method used for the address.
	current_addresses Array of WrappedNetworkInterfaceAddressIPv6, **required**	WrappedNetworkInterfaceAddressIPv6[]	
		address NetworkInterfaceAddressIPv6 (string), **required**	IPv6 network address in CIDR notation

1.3.88 NetworkInterfaceAddressConfigurationIPv4

Type Information

Array of any of NetworkAddressConfigurationIPv4Static or NetworkAddressConfigurationIPv4DHCP

1.3.89 NetworkInterfaceAddressConfigurationIPv6

Type Information

Array of any of NetworkAddressConfigurationIPv6Static, NetworkAddressConfigurationIPv6DHCP or NetworkAddressConfigurationIPv6Auto

1.3.90 NetworkInterfaceAddressConfigurationState

Configurable network address configuration of a network interface

Properties

ipv4 NetworkInterfaceAddressFamilyStateIPv4, optional	IPv4 Network address configuration		
	ipv4		

	address_configurations Array of any of NetworkAddressConfigurationIPv4Static or NetworkAddressConfigurationIPv4DHCP, optional	NetworkAddressConfigurationIPv4Static[]	
		method string, one of [`static, dhcp`], **required**	Configuration method used for the address.
		address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation
		gateway NetworkAddressIPv4 (string), optional	default gateway for outgoing traffic
		NetworkAddressConfigurationIPv4DHCP[]	
		method string, one of [`static, dhcp`], **required**	Configuration method used for the address.
ipv6 NetworkInterfaceAddressFamilyStateIPv6, optional	IPv6 Network address configuration		
	ipv6		
	address_configurations Array of any of NetworkAddressConfigurationIPv6Static, NetworkAddressConfigurationIPv6DHCP or NetworkAddressConfigurationIPv6Auto, optional	NetworkAddressConfigurationIPv6Static[]	
		method string, one of [`static, dhcp`], **required**	Configuration method used for the address.
		address NetworkInterfaceAddressIPv6 (string), **required**	IPv6 network address in CIDR notation
		gateway NetworkAddressIPv6 (string), optional	default gateway for outgoing traffic
		NetworkAddressConfigurationIPv6DHCP[]	
		method string, one of [`static, dhcp, auto`], **required**	Configuration method used for the address.
		NetworkAddressConfigurationIPv6Auto[]	
		method string, one of [`static, dhcp, auto`], **required**	Configuration method used for the address.

Examples

Remove all IPv6 address configurations

```
{
  "ipv6": {
    "address_configurations": []
  }
}
```

Replace existing IPv4 configuration with DHCP

```
{
  "ipv4": {
    "address_configurations": [
      {
        "method": "dhcp"
      }
    ]
  }
}
```

Set static and dynamic IPv4 configuration

```
{
  "ipv4": {
    "address_configurations": [
      {
        "method": "dhcp"
      },
      {
        "method": "static",
        "address": "192.168.0.100/24"
      }
    ]
  }
}
```

1.3.91 NetworkInterfaceAddressesIPv4

Properties

current_addresses Array of WrappedNetworkInterfaceAddressIPv4, **required**	WrappedNetworkInterfaceAddressIPv4[]	
	address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation

1.3.92 NetworkInterfaceAddressesIPv6

Properties

current_addresses Array of WrappedNetworkInterfaceAddressIPv6, **required**	WrappedNetworkInterfaceAddressIPv6[]	
	address NetworkInterfaceAddressIPv6 (string), **required**	IPv6 network address in CIDR notation

1.3.93 NetworkInterfaceAddressFamilyInformationIPv4

IPv4 Network address configuration

Properties

address_configurations Array of any of NetworkAddressConfigurationIPv4Static or NetworkAddressConfigurationIPv4DHCP, optional	NetworkAddressConfigurationIPv4Static[]	

	method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.
	address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation
	gateway NetworkAddressIPv4 (string), optional	default gateway for outgoing traffic
	NetworkAddressConfigurationIPv4DHCP[]	
	method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.
current_addresses Array of WrappedNetworkInterfaceAddressIPv4, **required**	WrappedNetworkInterfaceAddressIPv4[]	
	address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation

Examples

```
{
  "address_configurations": [
    {
      "method": "static",
      "address": "169.254.168.150/16"
    }
  ],
  "current_addresses": [
    {
      "address": "169.254.168.150/16"
    }
  ]
}
```

1.3.94 NetworkInterfaceAddressFamilyInformationIPv6

IPv6 Network address configuration.

Properties

address_configurations Array of any of NetworkAddressConfigurationIPv6Static, NetworkAddressConfigurationIPv6DHCP or NetworkAddressConfigurationIPv6Auto, optional	NetworkAddressConfigurationIPv6Static[]	
	method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.
	address NetworkInterfaceAddressIPv6 (string), **required**	IPv6 network address in CIDR notation
	gateway NetworkAddressIPv6 (string), optional	default gateway for outgoing traffic
	NetworkAddressConfigurationIPv6DHCP[]	
	method string, one of [`static`, `dhcp`, `auto`], **required**	Configuration method used for the address.
	NetworkAddressConfigurationIPv6Auto[]	
	method string, one of [`static`, `dhcp`, `auto`], **required**	Configuration method used for the address.

current_addresses Array of WrappedNetworkInterfaceAddressIPv6, **required**	WrappedNetworkInterfaceAddressIPv6[]	
	address NetworkInterfaceAddressIPv6 (string), **required**	IPv6 network address in CIDR notation

Examples

```
{
  "address_configurations": [
    {
      "method": "static",
      "address": "fda0:576b:c643:100::100/64"
    },
    {
      "method": "auto"
    }
  ],
  "current_addresses": [
    {
      "address": "fda0:576b:c643:100::100/64"
    },
    {
      "address": "fd01::40f:10ff:fe02:e6f/64"
    },
    {
      "address": "fe80::40f:10ff:fe02:e6f/64"
    }
  ]
}
```

1.3.95 NetworkInterfaceAddressFamilyStateIPv4

IPv4 Network address configuration.

Properties

address_configurations Array of any of NetworkAddressConfigurationIPv4Static or NetworkAddressConfigurationIPv4DHCP, optional	NetworkAddressConfigurationIPv4Static[]	
	method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.
	address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation
	gateway NetworkAddressIPv4 (string), optional	Standard Gateway für abgehenden Verkehr
	NetworkAddressConfigurationIPv4DHCP[]	
	method string, one of [`static`, `dhcp`], **required**	Configuration method used for the address.

1.3.96 NetworkInterfaceAddressFamilyStateIPv6

IPv6 Network address configuration.

Properties

address_configurations Array of any of NetworkAddressConfigurationIPv6Static,	NetworkAddressConfigurationIPv6Static[]	

NetworkAddressConfigurationIPv6DHCP or NetworkAddressConfigurationIPv6Auto, optional		
	method string, one of [`static, dhcp`], **required**	Configuration method used for the address.
	address NetworkInterfaceAddressIPv6 (string), **required**	IPv6 network address in CIDR notation
	gateway NetworkAddressIPv6 (string), optional	default gateway for outgoing traffic
	NetworkAddressConfigurationIPv6DHCP[]	
	method string, one of [`static, dhcp, auto`], **required**	Configuration method used for the address.
	NetworkAddressConfigurationIPv6Auto[]	
	method string, one of [`static, dhcp, auto`], **required**	Configuration method used for the address.

1.3.97 NetworkInterfaceAddressIPv4

IPv4 network address in CIDR notation.

Type Information

NetworkInterfaceAddressIPv4 (string)

Examples

```
192.168.1.100/24
```

1.3.98 NetworkInterfaceAddressIPv6

IPv6 network address in CIDR notation.

Type Information

NetworkInterfaceAddressIPv6 (string)

Examples

```
fda0:576b:c643:100:40f:10ff:fe02:e6f/64
```

1.3.99 NetworkInterfaceInformation

Description of the currently active addresses of the interface and its configuration.

Properties

iface NetworkInterfaceName (string), pattern: `^[a-z0-9-]+$`, **required**, read-only	unique name describing a network interface		
hardware_address MacAddress (string), pattern: `^([a-f0-9]{2}:){5}[a-f0-9]{2}$`, required, read-only	unique hardware address of a network interface		
has_link boolean, **required**, read-only	current physical connection status (whether a cable is plugged in or not)		
ipv4 NetworkInterfaceAddressFamilyStateIPv4, optional	IPv4 Network address configuration		
	ipv4		

	address_configurations Array of any of NetworkAddressConfigurationIPv4Static or NetworkAddressConfigurationIPv4DHCP, optional	NetworkAddressConfigurationIPv4Static[]	
		method string, one of [`static, dhcp`], **required**	Configuration method used for the address.
		address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation
		gateway NetworkAddressIPv4 (string), optional	default gateway for outgoing traffic
		NetworkAddressConfigurationIPv4DHCP[]	
		method string, one of [`static, dhcp`], **required**	Configuration method used for the address.
ipv6 NetworkInterfaceAddressFamilyStateIPv6, optional	IPv6 Network address configuration		
	ipv6		
	address_configurations Array of any of NetworkAddressConfigurationIPv6Static, NetworkAddressConfigurationIPv6DHCP or NetworkAddressConfigurationIPv6Auto, optional	NetworkAddressConfigurationIPv6Static[]	
		method string, one of [`static, dhcp`], **required**	Configuration method used for the address.
		address NetworkInterfaceAddressIPv6 (string), **required**	IPv6 network address in CIDR notation
		gateway NetworkAddressIPv6 (string), optional	default gateway for outgoing traffic
		NetworkAddressConfigurationIPv6DHCP[]	
		method string, one of [`static, dhcp, auto`], **required**	Configuration method used for the address.
		NetworkAddressConfigurationIPv6Auto[]	
		method string, one of [`static, dhcp, auto`], **required**	Configuration method used for the address.

Examples

Remove all IPv6 address configurations

```
{
  "ipv6": {
    "address_configurations": []
  }
}
```

Replace existing IPv4 configuration with DHCP

```
{
  "ipv4": {
    "address_configurations": [
      {
        "method": "dhcp"
      }
    ]
  }
}
```

Set static and dynamic IPv4 configuration

```
{
  "ipv4": {
    "address_configurations": [
      {
        "method": "dhcp"
      },
      {
        "method": "static",
        "address": "192.168.0.100/24"
      }
    ]
  }
}
```

1.3.100 NetworkInterfaceName

Unique name describing a network interface.

Type Information

NetworkInterfaceName (string), pattern: `^[a-z0-9-]+$`

Examples

```
eth0
```

1.3.101 NetworkInterfaceStaticData

Properties

iface NetworkInterfaceName (string), pattern: `^[a-z0-9-]+$`, **required**, read-only	unique name describing a network interface
hardware_address MacAddress (string), pattern: `^([a-f0-9]{2}:){5}[a-f0-9]{2}$`, **required**, read-only	unique hardware address of a network interface
has_link boolean, **required**, read-only	current physical connection status (whether a cable is plugged in or not)

1.3.102 NormalizationConstant

Normalization constants are related to the White Reference.

Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (`e.g. /api/sensor/detection-profiles/{itemId}/white-reference`).

Type Information

Array of number

1.3.103 ReleaseChannel

Describes the kind of a publication

Releases on the `stable` channel are generally considered well-tested and are recommended for use in production.

Releases on the `feature` add new features but haven't been tested as much as a stable release. Feature releases can but should only be used in production with careful consideration.

Type Information

ReleaseChannel (string), one of [`stable`, `feature`] , default: `stable`

Examples

```
stable
```

1.3.104 SampleRate

Type Information

SampleRate (number), minimum: 0.01

1.3.105 SamplingSettings

Sampling Settings describe all details of the sampling process.

Its attributes may be queried and inspected (e.g. in order to retrieve the current sample rate).

Most values stored within the Sampling Settings should not be modified directly. The related API endpoint `/api/sensor/detection-profiles/current/autogain` should be used instead.

The only modifiable attribute within the Sampling Settings is the *averages* value. It is safe to change it, even though the default values calculated during an autogain operation should be optimal for most detection tasks.

Properties

led_intensity Number, minimum: 0, maximum: 1, required	relative intensity of the internal emitter during the light phase
base_sample_rate SampleRate (number), minimum: 0.01, **required**	The base sample rate determines the duration of a sampling period. After each sampling period, the gathered data is processed and a new detection result is calculated (e.g. the most suitable *Matcher* for the given sample). This may affect the state of the Switching Outputs or trigger configured actions. Thus the base sample rate defines the maximum rate of changes for the Switching Outputs. See also the *effective sample rate*.
effective_sample_rate SampleRate (number), minimum: 0.01, **required**	The effective sample rate is the numeric product of the *base sample rate* and the number of *averages*. It determines the minimum duration that a target needs to be sampled in order to determine its visual appearance correctly. With the default value of *average* set to one, this value is equal to the base sample rate.
minimum_wanted_sample_rate SampleRate (number), minimum: 0.01, **required**	This informational value represents the sample rate that was requested during the most recent *Autogain* operation. The effective sample rate may deviate from the wanted sample rate, if the requested sample rate was not achievable due to limitations of the sensor (e.g. exceeding the supported sample rate) or due to the environment (e.g. not enough light, thus a slower amplification with higher gain was necessary).
sample_light_phase Boolean, **required**	defines if the sensor should periodically activate the internal emitter for sampling

sample_dark_phase Boolean, **required**	defines if the sensor should periodically deactivate the internal emitter for sampling
averages AverageSampleCount (integer), minimum: 1, **required**	Number of previous samples to be averaged for every sampling result. A rolling averaging algorithm is applied to the samples.
amplification AmplificationLevel (integer), **required**	The amplification level specifies the internal configuration of an amplifier. This value is not meant to be manipulated by regular users. It should be handled *as is* (stored, transmitted and applied without modification or introspection).

Example

```
{
  "led_intensity": 0.7,
  "base_sample_rate": 1000,
  "effective_sample_rate": 1000,
  "minimum_wanted_sample_rate": 1000,
  "sample_light_phase": true,
  "sample_dark_phase": true,
  "averages": 1,
  "amplification": 5
}
```

1.3.106 SensorCapabilities

Provide access to the sensoric details supported by this device (e.g. colorspaces, input and output lines, ...).

Properties

maximum_sample_rate Integer, **required**	the maximum sample rate the sensor supports		
tolerances Array of ColorTolerance (union), **required**	List of tolerance specifications supported by the sensor		
	InfiniteColorTolerance		
	limits Object, **required**	limits	
	shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.	
	SphereColorTolerance		
	limits Object, **required**	limits **radius** Numer, **required**	
	shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.	
	CylinderColorTolerance		
	limits Object, **required**	limits **radius** Number, **required** **half_height** Number, **required**	
	shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.	
	BoxColorTolerance		
	limits Object, **required**	limits **half_edges**	

		Array of number, minimum items: 3, maximum items: 3, **required**	
	shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities.`	
output_drivers Array of SwitchingOutputDriver (string), **required**	List of supported electrical output drivers		
trigger_sources Array of TriggerSource, **required**	Beinhaltet die Liste verfügbarer Auslösequellen mit ihrem dazugehörigen Auslösefall. Auslösefälle können zum Ausführen bestimmter Aktionen automatisiert werden.		
	TriggerSource[]		
	name String, **required**	Name of the trigger input	
	events Array of TriggerEvent, **required**	TriggerEvent []	
		name TriggerEventName (string), **required**	
output_pin_count Integer, **required**	Number of available switching output lines		
Actions Array of Action, **required**, Deprecated	Deprecated: use /api/actions instead		
	Action[]		
	name String, **required**	Unique name of the action	
	arguments Object, **required**	arguments	
colorspaces Array of Colorspace, **required**	List of supported colorspaces.		
	Colorspace[]		
	name String, **required**		
	space_id ColorspaceID, **required**	Unique name of a colorspace	
	axes Array of ColorspaceAxis, minimum items: 3, maximum items: 3, **required**	ColorspaceAxis[]	
		id String, **required**	Unique name
		label String, **required**	Human-readable name
		minimum Number, **required**	lowest expected value of a color along this axis under usual circumstances
		maximum Number, **required**	highest expected value of a color along this axis under usual circumstances
colorspace_tolerance_maps Array of ColorspaceToleranceMap, **required**	The evaluation of tolerances against positions of detectables depends on the currently configured colorspace. For example the tolerance attribute "half_height" refers to the brightness-related axis of a colorspace (e.g. "L*" for the "L*a*b*" colorspace) and is used for the height of the cylindrical tolerance shape and the first edge of the box tolerance shape.		

	The hue-related attributes (e.g. "a" and "b" for the "Lab*" colorspace) are used for the "radius" of a cylinder tolerance shape and the second and third edges of the box tolerance shape. The *colorspace_tolerance_maps* define these relationships between colorspaces and tolerances.		
	ColorspaceToleranceMap[]		
	colorspace_id ColorspaceID (string), **required**	Unique name of a colorspace	
	tolerance_shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities .`	
	limits_axes_map Object, **required**	limits_axes_map	
		half_height Array of string, optional	
		half_edges Array of string, optional	
		radius Array of string, optional	
settings_categories Array of string, **required**	List of categories that can be selected during import to control which settings should be applied. See the documentation for the POST request to `` ` ``/api/seetings.		
maximum_detectables_count Integer, **required**	Maximum number of color positions (*Detectable*) to be stored in a detection profile.		
maximum_matchers_count Integer, **required**	Maximum number of detection results (*Matcher*) be stored in a detection profile.		

Examples

```
{
  "output_pin_count": 8,
  "tolerances": [
    {
      "shape": "infinite",
      "limits": {}
    },
    {
      "shape": "sphere",
      "limits": {
        "radius": 2
      }
    },
    {
      "shape": "cylinder",
      "limits": {
        "half_height": 4,
        "radius": 2
      }
    },
    {
      "shape": "box",
      "limits": {
        "half_edges": [
          4,
          2,
          2
        ]
      }
    }
  ],
  "actions": [
    {
      "name": "enable_switching_output",
      "arguments": {}
    },
    {
      "name": "teach_single",
      "arguments": {}
    }
  ],
  "maximum_sample_rate": 20000,
  "maximum_detectables_count": 256,
  "maximum_matchers_count": 256,
  "trigger_sources": [
    {
      "name": "trigger_0",
      "events": [
        {
          "name": "trigger_0_level_high"
        },
        {
          "name": "trigger_0_level_low"
        },
        {
          "name": "trigger_0_edge_rising"
        },
        {
          "name": "trigger_0_edge_falling"
        }
      ]
    },
    {
      "name": "trigger_1",
      "events": [
        {
          "name": "trigger_1_level_high"
        },
        {
          "name": "trigger_1_level_low"
        },
        {
          "name": "trigger_1_edge_rising"
        },
        {
          "name": "trigger_1_edge_falling"
        }
      ]
    },
    {
      "name": "trigger_2",
      "events": [
        {
          "name": "trigger_2_level_high"
        },
        {
          "name": "trigger_2_level_low"
        },
        {
          "name": "trigger_2_edge_rising"
        },
        {
          "name": "trigger_2_edge_falling"
        }
      ]
    },
    {
      "name": "trigger_3",
      "events": [
        {
          "name": "trigger_3_level_high"
        },
        {
          "name": "trigger_3_level_low"
        },
        {
          "name": "trigger_3_edge_rising"
        },
        {
          "name": "trigger_3_edge_falling"
        }
      ]
    }
  ],
  "colorspaces": [
    {
      "axes": [
        {
          "id": "L",
          "label": "L*",
          "minimum": 0,
          "maximum": 100
        },
        {
          "id": "a",
          "label": "a*",
          "minimum": -500,
          "maximum": 500
        },
        {
          "id": "b",
          "label": "b*",
          "minimum": -200,
          "maximum": 200
        }
      ],
      "name": "L*a*b*",
      "space_id": "Lab"
    },
    {
      "axes": [
        {
          "id": "L",
          "label": "L*",
          "minimum": 0,
          "maximum": 100
        },
        {
          "id": "u",
          "label": "u*",
          "minimum": 0,
          "maximum": 100
        },
        {
          "id": "v",
          "label": "v*",
          "minimum": 0,
          "maximum": 100
        }
      ],
      "name": "L*u*v*",
      "space_id": "Luv"
    },
    {
      "axes": [
        {
          "id": "X",
          "label": "X",
          "minimum": 0,
          "maximum": 120
        },
        {
          "id": "Y",
          "label": "Y",
          "minimum": 0,
          "maximum": 100
        },
        {
          "id": "Z",
          "label": "Z",
          "minimum": 0,
          "maximum": 120
        }
      ],
      "name": "XYZ",
      "space_id": "XYZ"
    },
    {
      "axes": [
        {
          "id": "x",
          "label": "x",
          "minimum": 0,
          "maximum": 1
        },
        {
          "id": "y",
          "label": "y",
          "minimum": 0,
          "maximum": 1
        },
        {
          "id": "Y",
          "label": "Y",
          "minimum": 0,
          "maximum": 100
        }
      ],
      "name": "xyY",
      "space_id": "xyY"
    },
    {
```

```
      "axes": [
        {
          "id": "L",
          "label": "L*",
          "minimum": 0,
          "maximum": 100
        },
        {
          "id": "u",
          "label": "u'",
          "minimum": 0,
          "maximum": 1
        },
        {
          "id": "v",
          "label": "v'",
          "minimum": 0,
          "maximum": 1
        }
      ],
      "name": "L*u'v'",
      "space_id": "uvL"
    }
  ],
  "output_drivers": [
    "off",
    "npn",
    "pnp",
    "push-pull"
  ],
  "colorspace_tolerance_maps": [
    {"colorspace_id": "Lab", "tolerance_shape": "box",
     "limits_axes_map": {"half_edges": ["L", "a", "b"]}},
    {"colorspace_id": "Lab", "tolerance_shape": "cylinder",
     "limits_axes_map": {"half_height": ["L"], "radius": ["a", "b"]}},
    {"colorspace_id": "Luv", "tolerance_shape": "box",
     "limits_axes_map": {"half_edges": ["L", "u", "v"]}},
    {"colorspace_id": "Luv", "tolerance_shape": "box",
     "limits_axes_map": {"half_height": ["L"], "radius": ["u", "v"]}}
  ],
  "settings_categories": [
    "access",
    "defaults",
    "emitters",
    "firmware",
    "keypad",
    "network",
    "outputs",
    "peripherals",
    "sensor",
    "system"
  ]
}
```

1.3.107 SerialElizaSettings

Properties

type
string, one of [none, eliza, modbus] , default: eliza, **required**

1.3.108 SerialModbusSettings

Properties

type string, one of [`none, eliza, modbus`] , default: `eliza`, **required**
slave_id any of number or null, **required**
frame_format string, one of [`rtu, ascii`] , default: `rtu`, **required**

1.3.109 SignalColor

A custom color name. How and what color will be displayed is defined by the client.

Type Information

any of string or null

1.3.110 SphereColorTolerance

Properties

limits Object, **required**	limits
	radius number, **required**
shape ToleranceShapeName (string), **required**	Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.

1.3.111 SupportedTimezones

List of timezones supported by the device.

Type Information

Array of string

Examples

```
 [
  "Africa/Casablanca",
  "Antarctica/Troll",
  "Europe/Berlin",
  "UTC"
]
```

1.3.112 SwitchingOutputDriver

The Output Driver defines the electrical behaviour of the switching outputs. The supported output drivers can be retrieved via `/api/sensor/capabilities`.

Type Information

SwitchingOutputDriver (string)

Examples

```
push-pull
```

1.3.113 SwitchingOutputs

Eletrical output lines can drive external actors in different electrical modes.

Properties

output_driver SwitchingOutputDriver (string), **required**	The Output Driver defines the electrical behaviour of the switching outputs. The supported output drivers can be retrieved via `/api/sensor/capabilities`.
count integer, **required**	Number of available output lines

Examples

```
{
  "count": 8,
  "output_driver": "push-pull"
}
```

1.3.114 SwitchingOutputsWritable

Eletrical output lines can drive external actors in different electrical modes.

Properties

output_driver SwitchingOutputDriver (string), **required**	The Output Driver defines the electrical behaviour of the switching outputs. The supported output drivers can be retrieved via `/api/sensor/capabilities`.

1.3.115 SystemSettings

Properties

hostname Hostname, pattern: `^(?:[a-zA-Z0-9](?:[a-zA-Z0-9\-]*[a-zA-Z0-9])?\.)*[a-zA-Z0-9](?:[a-zA-Z0-9\-]*[a-zA-Z0-9])?$`, optional	Human-readable name identifying the device in the network
uptime any of number or null, optional, read-only	The current system uptime in seconds. Though highly unlikely can be nil in case the system reported an invalid value.

Examples

```
{
  "hostname": "cfo-7454232361"
}
```

1.3.116 SystemTimeSettings

Properties

now Timestamp (string), optional	current time from the perspective of the sensor
timezone String, optional	currently configured timezone
ntp_servers Array of string, optional	one or more network time servers
default_ntp_servers Array of string, optional, reaed-only	preconfigured network time servers

Examples

```
{
  "now": "2018-01-24T15:45:15.694004+01:00",
  "timezone": "Europe/Berlin",
  "ntp_servers": [
    "pool.ntp.org"
  ],
  "default_ntp_servers": [
    "pool.ntp.org"
  ]
}
```

1.3.117 Timestamp

Timestamp (Format: ISO 8601)

Type Information

Timestamp (string)

Examples

```
2018-01-24T14:04:26+01:00
```

1.3.118 TimestampBackendUptime

The timestamp (given in microseconds) is based on the uptime of the internal analog sensor backend. It may get reset to zero under specific conditions.

Type Information

TimestampBackendUptime (number), minimum: 0

1.3.119 ToleranceShapeName

Name of the geometrical shape of the tolerance. The supported tolerance shapes can be retrieved via `/api/sensor/capabilities`.

Type Information

ToleranceShapeName (string)

1.3.120 TransformedColor

A color represented by a coordinate in the colorspace. The array indices of the `values` property match the order of the `colorspace.axes` property of currently used detection profile.

1.3.121 TriggerEvent

Trigger Events can be emitted by their trigger source. Actions can be attached to a Trigger Event (see `/api/sensor/action-triggers`).

Properties

name TriggerEventName (string), **required**

1.3.122 TriggerEventName

Type Information

TriggerEventName (string)

Examples

```
trigger_0_level_high
```

1.3.123 TriggerSource

Each Trigger Source is a peripheral input with the ability to emit one or more Trigger Events.

Properties

name String, **required**	Name des Auslöseeingangs
events Array of TriggerEvent, **required**	TriggerEvent []
	name TriggerEventName (string), **required**

Examples

```
{
  "name": "trigger_0",
  "events": [
    {
      "name": "trigger_0_level_high"
    },
    {
      "name": "trigger_0_level_low"
    },
    {
      "name": "trigger_0_edge_rising"
    },
    {
      "name": "trigger_0_edge_falling"
    }
  ]
}
```

1.3.124 TriggerSourcesStatus

The sensor has a number of input lines that can be used as trigger sources. The event counters are updated periodically (approximately every second).

Properties

trigger_sources Array of object, **required**	object[]	
	name string, **required**	
	event_counters Object, **required**	event_counters
		edge_falling Number, **required**
		edge_rising Number, **required**
		level_low Number, **required**

Examples

```
{
  "trigger_sources": [
    {
      "name": "trigger_0",
      "event_counters": {
        "edge_falling": 22,
        "edge_rising": 23,
        "level_low": 35124823,
        "level_high": 15
      }
    },
    {
      "name": "trigger_1",
      "event_counters": {
        "edge_falling": 0,
        "edge_rising": 0,
        "level_low": 35124832,
        "level_high": 0
      }
    }
  ]
}
```

1.3.125 User

Properties

name string, pattern: `^[\w-]+$`, **required**, read-only	unique name identifying an account
password string, optional	Password assigned to this account (only writable; never returned in responses). Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
password_hash HashDigest (string), pattern: `^[a-f0-9]+$`, optional	Password hash assigned to this account. Either a `password` or a `password_hash` needs to be supplied when creating a new user or changing a password.
roles Array of string, optional	The roles assigned to an account define its set of permissions.

Examples

```
{
  "name": "alice"
}
```

1.3.126 UUID

Unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8.

Type Information

UUID (string), pattern: `^[a-f0-9-]+$`

Examples

```
a014e415-0fec-4734-ac3f-30da0a5f3899
```

1.3.127 WantedSwitchingOutputsState

The combination of tristate values describes a logical state of the switching outputs of the sensor.

The states `true` or `false` cause the output to go up or down. The state `null` keeps the previous state of the output unchanged.

Properties

uuid UUID (string), pattern: `^[a-f0-9-]+$`, **required**, read-only	unique identifier (UUID) as defined by RFC 4122, ITU-T Rec. X.667, and ISO/IEC 9834-8
states Array of any of boolean or null, **required**	List of True/False/Null values describing the wanted states of the Switching Outputs

Examples

```
{
  "uuid": "3f26aff4-8650-42a0-b319-51776c443fbc",
  "states": [
    true,
    false,
    false,
    false,
    false,
    false,
    false,
    false
  ]
}
```

1.3.128 WhiteReference

The White Reference attribute is used for indicating a custom color balancing.

Its content is subject to internal use. Thus it should not be accessed directly, but only through the related API endpoints (e.g. `/api/sensor/detection-profiles/{itemId}/white-reference`).

Type Information

Array of number

1.3.129 WrappedNetworkInterfaceAddressIPv4

Network addresses (IPv4) in CIDR notation.

Properties

address NetworkInterfaceAddressIPv4 (string), **required**	IPv4 network address in CIDR notation

1.3.130 WrappedNetworkInterfaceAddressIPv6

Network addresses (IPv6) in CIDR notation.

Properties

address NetworkInterfaceAddressIPv6 (string), **required**	IPv6 network address in CIDR notation

2 Terminal Documentation

The text-based terminal is accessible via the following interfaces of the sensor:

- RS-232 (SYS connector pins)
- USB (optional)

2.1 Preface

2.1.1 Connection Details

The sensor uses the following configuration to communicate via the serial interface:

Baud Rate	19200 (RS-232)
Data Bits	`8`
Stop Bits	`1`
Parity	None
Line Feed	`0x0A`/LF/`\n`
Encoding	UTF-8

2.1.2 Syntax

The serial console uses color, markup, and font weight to outline how commands can and should be used. As not all terminals support color and font formatting rules you may only see the described markup in your terminal.

Output is formatted as following:

- lowercase letters indicate a keyword
- uppercase letters indicate a variable
- square brackets indicate an optional keyword or variable
- the `[...]` character sequence indicates an unlimited number of variables
- optional keywords and variables are colored in grey
- required variables are bold & white

Please mind that all input is case-sensitive. `matcher select` is not the same as `MATCHER SELECT`.

BEISPIEL

matcher select `MATCHER` set tolerance `SHAPE` [limits]

- *set a matcher's tolerance*

In the example above the syntax expresses the following command structure:

1. a required keyword "`matcher`"
2. a required keyword "`select`"
3. a required variable "`MATCHER`"
4. a required keyword "`set`"
5. a required keyword "`tolerance`"
6. a required variable "`SHAPE`"
7. an optional variable "`LIMITS`"

2.1.3 Output Formats & Message Parsing

When used in automation it's recommended to use the JSON output format. You can switch between output formats with the `set output-format` command. The available formats are `json` and `human`. Please be advised that human output format is subject to change and is currently not recommended for parsing.

Once a command is executed it will output one or more data packets. Independent of the output format these packets end with a specific two-byte sequence. The first byte indicates if the packet was generated by a successful command and is either `0x20` in case of success or `0x07` in case of a failure. The second byte marks the end of the packet and is guaranteed to be `0x00`, the ASCII NUL character.

2.1.4 Nomenclature

The serial interface uses a set of names and identifiers that are closely modelled after the sensors REST API.

UUID

UUID is short for "Universally unique identifier" and is a 36-characters long character sequence with 5 alpha-numeric groups separated by dashes. Most collections use a UUID as unique keys for their items. UUIDs used by the serial interface and the underlying REST API are UUIDv4 as specified in RFC 4122.

Detectable position

The commands for adding and editing a detectable use a parameter called the position.

This value refers to the color property in the REST API and describes the three-dimensional location of the color in the currently activate colorspace. The format regex is `^(\d+(?:\.\d*)?),(\d+(?:\.\d*)?),(\d+(?:\.\d*)?)$` (Example: `3.14,7.6,8`).

2.1.5 Common Patterns

Some tasks in the serial interface share common or similar behaviours. These recurring patterns are outlined here.

Collections and the `select` keyword:

Collections are a set of items of a specific type that the sensor controls. Most collections handled by the sensor use a UUID as unique key, by which they can be referenced.

In the context of the serial interface collections can easily be spotted by the use of the "select" keyword followed by a variable as part of the command syntax. Even though collections usually use a UUID or another unique key as identifier, the variable can take other arguments. You may also use a list index, as defined by the corresponding list command for this collection or the special index '-1' to reference the last item of this type that you've created.

Property Commands using `show [PROPERTY]`:

Most commands that display detailed information on an object (may it be a collection item or any other resource) using the the `show` keyword support the output of any of the attributes individually. The help command for the command will show you the allowed choices for the property value.

2.1.6 Differences compared to the REST API

The serial protocol is based on the REST API but there are some differences that are outlined in this section.

- There is no feature parity. The REST API is the primary configuration method for the sensor. Not all features implemented in the REST API are available via the serial interface.
- Some commands like `samples stream` implement parameters that don't match the corresponding API endpoint. This may be the case if the behaviour controlled by these parameters is specific to the HTTP or serial protocol.
- While the REST API uses plural resources names like `matchers` the serial protocol uses singular command names (in this case `matcher`).
- There is no general `/defaults` API endpoint implementation available on the serial interface. Instead commands like `matcher default set hold_time` implement a more fine-grained access

2.2 Command Reference

2.2.1 Access Command

Handle security related tasks like logins and session management.

Supported Subcommands

```
access login USERNAME [PASSWORD]
```

- Login with username and password to access privileged commands.

```
access logout
```

- Logout and invalidate the current session.

```
access session
```

- Show current session information.

2.2.2 Device Command

Supported Subcommands

```
device [show] [PROPERTY]
```

[PROPERTY]: `id | model_key | model_name | variant | vendor_key | vendor_name`

2.2.3 Firmware Command

The sensor firmware provides the connectivity of the sensor and its services, as well as the functionality of sensor backend.

Supported Subcommands

```
firmware [version]
```

- Show the current firmware version.

```
firmware recovery [show] [PROPERTY]
```

- Show information about the recovery firmware.
  [PROPERTY]: `channel | created_on | id | name | version`

```
firmware recovery restore
```

- Restore the system from the recovery image. All settings are reset to their defaults.

```
firmware recovery upgrade
```

- Store the currently running firmware image as a recovery image. You may use this operation after verifying a successful firmware upgrade.

2.2.4 Help Command

List and describe all available commands.

Supported Subcommands

```
help [COMMAND]
```

2.2.5 Keypad Command

The keypad provides local access to most basic sensor actions.

Supported Subcommands

```
keypad [show] [PROPERTY]
```

- Show the keypad status.
  [PROPERTY]: `lock`

```
keypad lock [STATE]
```

- Change the state of the keypad lock mechanism.

2.2.6 Matcher Command

A matcher specifies the sensor behaviour based on the sensor input.

Supported Subcommands

Matcher

A matcher contains multiple detectables (colors) and the desired sensor behaviour (e.g. out put states and hold time) that should be applied when one of the colors is detected.

```
matcher[list]
```

- Show the list of configured matchers for the detection profile.

```
matcher add [OUTPUT_PATTERN]
```

- Add a matcher to the detection profile.

```
matcher remove all
```

- Remove all matchers.

```
matcher select MATCHER [show] [PROPERTY]
```

- Show detailed information about a matcher.
  [PROPERTY]: `hold_time | name | num_detectables | output_pattern | reset_output_after_hold_time_expired | signal_color | tolerance | uuid`

```
matcher select MATCHER remove
```

- Remove a single matcher.

```
matcher select MATCHER set name NAME
```

- Set a matcher’s name.

```
matcher select MATCHER set hold_time DURATION
```

- Set a matcher’s hold time.

```
matcher select MATCHER set output_pattern BITMASK
```

- Set a matcher's output bitmask.

```
matcher select MATCHER set tolerance SHAPE [LIMITS]
```

- Set a matcher's tolerance.
  SHAPE: One of the shapes defined by the API.
  `box / cylinder / infinite / sphere`

[LIMITS]: A string describing the limits of the tolerance (e.g. "2r" for a sphere, "4h/6r" for a cylinder, or "1/2/3" for a box). Optional for infinite, but required for every other shape.

Matcher Defaults

Whenever a new matcher is created a few properties are set to predefined default values. You can change these defaults to your liking in order to reduce the number of changes needed afterwards.

```
matcher default [show] [PROPERTY]
```

- Display matcher default values.
  [PROPERTY]: *hold_time* | *tolerance*

```
matcher default set hold_time DURATION
```

- Set the default hold time for new matchers.

```
matcher default set tolerance SHAPE [LIMITS]
```

- Set the default tolerance for new matchers.
  SHAPE: One of the shapes defined by the API.
  `box | cylinder | infinite | sphere`
  [LIMITS]: A string describing the limits of the tolerance (e.g. "2r" for a sphere, "4h/6r" for a cylinder, or "1/2/3" for a box). Optional for infinite, but required for every other shape.

Detectable

Multiple detectables can belong to a matcher. Each detectable represents a position within the currently active colorspace.

```
matcher select MATCHER detectable [list]
```

- List all detectables belonging to a matcher.

```
matcher select MATCHER detectable remove all
```

- Remove the detectables belonging to a matcher.

```
matcher select MATCHER detectable add [POSITION]
```

- Add a detectable to a matcher. Sample the current detectable, if no position is given.
  [POSITION]: A position in the current color space.
  Expected format: 23.918,6,17.29113

```
matcher select MATCHER detectable select DETECTABLE [show] [PROPERTY]
```

- show detailed information about a detectable
  [PROPERTY]: matcher | position | rgb | uuid

```
matcher select MATCHER detectable select DETECTABLE remove
```

- Remove a single detectable from a matcher.

```
matcher select MATCHER detectable select DETECTABLE set position POSITION
```

- Modify the detectable's position in the colorspace.

[POSITION]:	Eine Position in dem aktuellen Farbraum. Erwartetes Format: 23.918,6,17.29113

2.2.7 Network Command

The network configuration allows the use of network-based sensor features (e.g. API or the web interface).

Supported Subcommands

```
network [list]
```

- Show the connection state and active addresses of all network interfaces.

```
network reset
```

- Reset the network configuration to its factory default.

```
network select INTERFACE [show] [PROPERTY]
```

- Show the connection state and active addresses of a network interface.

  [PROPERTY]: ipv4_addresses | ipv4_config | ipv6_addresses | ipv6_config | link | mac | name

```
network select INTERFACE set ipv4 dhcp
```

- Request a dynamically assigned IP address (via DHCP) for a network interface.

```
network select INTERFACE set ipv4 static ADDRESS [GATEWAY]
```

- Define a static IPv4 address for the network interface.

```
network select INTERFACE set ipv4 disabled
```

- Disable IPv4 connectivity for the network interface.

```
network select INTERFACE set ipv6 auto
```

- Enable IPv6 state-less auto network configuration (SLAAC) for the interface.

```
network select INTERFACE set ipv6 dhcp
```

- Request a dynamically assigned IP address (via DHCPv6) for a network interface.

```
network select INTERFACE set ipv6 static ADDRESS [GATEWAY]
```

- Define a static IPv6 address for the network interface.

```
network select INTERFACE set ipv6 disabled
```

- Disable IPv6 connectivity for the network interface.

2.2.8 Repeat Command

Conveniently execute a command multiple times (e.g. following changes of the color sampling results).

Supported Subcommands

```
repeat REPETITIONS DELAY [ARGUMENTS [...]]
```

- Repeat a single command for a number of times with a given delay. You may stop execution with CTRL-C.

REPETITIONS: number of repetitions, 0 for infinite
DELAY: delay in seconds, 0 for no delay

2.2.9 Sample Command

Request sample results from the sensor.

Supported Subcommands

```
sample [show] [PROPERTY]
```

- Show the current color sample.
  [PROPERTY]: color | detection | output_pattern | timestamp | trigger

```
sample stream [COUNT] [FREQUENCY]
```

- Retrieve a continuous stream of color samples from the sensor
  [COUNT]: number of records to retrieve, default 0 for infinite
  [FREQUENCY]: speed of samples in hertz, default infinite

2.2.10 Sensor Command

Sensor settings influence the sampling and processing of sensor signals. Changed settings may invalidate previously sampled detectables.

Supported Subcommands

```
sensor colorspace [show]
```

- Show the currently configured colorspace.

```
sensor colorspace list
```

- List available colorspaces.

```
sensor colorspace set COLORSPACE
```

- Switch to a different colorspace.

```
sensor autogain [SAMPLE_RATE] [TARGET_LEVEL]
```

- Perform the autogain procedure in order to adjust the sensor to the current optical environment (distance, light intensity, target appearance).

```
sensor white-reference reset
```

- Reset the white reference to the factory default. The factory default works well with a commonly used optical path (fiber and optics). Use the factory default if a proper white reference target is not available.

```
sensor white-reference sample
```

- Sample a new white reference from the current target. The target should be neutral white. This may improve the calculation of absolute color values within the given colorspace.

2.2.11 SET Command

Change properties of the console interface.

Supported Subcommands

`set echo STATE`

- enable/disable any output of prompts or typed text
  STATE: off | on

`set output-format FORMAT`

- switch the response output format
  FORMAT: human | json

2.2.12 System Command

Interact with the system hosting the sensor.

Supported Subcommands

`system settings reset`

- Reset all settings to their factory defaults.

`system hostname [show]`

- Show the system's hostname.

`system hostname set HOSTNAME`

- Define the hostname of the system.

`system timezones [list]`

- List all timezones supported by the system

`system time [show] [PROPERTY]`

- Show the system's current time settings
  [PROPERTY]: now | timezone

`system time set now TIME`

- Set the system's current time in ISO8601 format.

`system time set timezone TIMEZONE`

- Set the system's timezone

`system reboot`

- Reboot the device

3 Modbus Documentation

Modbus protocol is only available for the colorSENSOR CFOXXX(100) Option 100.

3.1 Introduction

The modbus protocol is a single-master protocol. Data is exchanged over a serial or via network (TCP/IP) interface. The Controller acts as a Modbus slave: It responds to requests from a master.

The Modbus protocol allows partial access to the most relevant features of the controller. Internally it uses the HTTP-based API of the controller for all operations.

The colorSENSOR Modbus interface supports the following protocol features:

- Transport via TCP (IPv4 and IPv6)

- Transport via RS232 and USB using RTU (default) or ASCII format

- Serial baudrates: 9600, 19200 (default), 115200

The Modbus slave address (relevant only for serial connections) is configurable. By default the colorSENSOR CFO is not bound to a specific address, but responds to every packet.

The full set of supported commands is available as a JSON dump. This structured dataset is supposed to ease the generation of a vendor-specific Modbus mapping for the controller.

3.2 Quickstart

The following configuration details and hints should ease the first steps with the controllers Modbus protocol implementation:

- Connect to the sensors Modbus protocol via RS232 (Baudrate: 19200), USB or TCP (port 502).

- Use Big-Endian (byte-order and word-order) when interpreting data in Modbus responses.

- Consider the 1-based addressing scheme when accessing registers. For example a documented address of 501 is transmitted over the wire as 500. Most Modbus client implementations will apply this translation implicitly. Only very few implementations use the on-wire address instead. In this case the documented address needs to be decremented for these specific clients.

- Retrieve the Input Registers from 500 up to 508 via a Modbus request. These registers contain fixed values in different formats (e.g. float, 32 bit and 64 bit integer). Ensure that your client implementation interpretes these values properly according to their documented value (see the register content documentation). In case of misinterpretations you may need to adjust the endianness or the address offset of your client implementation.

3.3 Supported colorSENSOR Features

The Modbus interface of the colorsensors provides most features of the following API endpoints:

- `/defaults (only matcher-releated defaults)`
- `/device`
- `/firmware (only status retrieval; no upgrade)`
- `/firmware/recovery`
- `/firmware/recvery/upgrade-from-current`
- `/sensor/samples/current`
- `/sensor/matchers`
- `/sensor/detectable`
- `/sensor/detection-profiles`
- `/sensor/detection-profiles/autogain`
- `/sensor/detection-profiles/white-reference`
- `/sensor/capabilities`
- `/system`
- `/system/factory-reset`
- `/system/reboot`
- `/peripherals/outputs`
- `/peripherals/rs232`
- `/peripherals/usb`
- `/settings`

The following API endpoints are not supported due to the volatile nature of their data or their complexity (hard to express within the modbus protocol):

- `/access`
- `/action-triggers`
- `/actions`
- `/firmware/images`
- `/firmware/settings`
- `/network/interfaces/`
- `/peripherals/keypad`
- `/peripherals/trigger-sources`
- `/system/time`
- `/system/time/zones`

3.4 Data Types and Register Addressing

3.4.1 Data Types and Modbus Functions

The Modbus protocol specifies different functions for accessing and manipulating values.

The following functions (and their respective function codes) are used for the different types of data:

Function	Code	Function name
Read-only bits	2	Read Discrete I
Writable bits	1	Read Coils
	5	Write single coils
	15	Write multiple coils
Read-only words	4	Read input registers
Writable words	3	Read Multiple Holding Registers
	6	Write Single Holding Register
	16	Write Multiple Holding Register
	23	Read/Write Multiple Registers

3.4.2 Register Addresses

The addressing of data via the Modbus protocol is not strictly specified. Different implementations use a variety of name schemes and offsets. The relevant details of this Modbus implementation are:

- All addresses written in this documentation are register offsets relative to the specific Modbus function.
- All addresses are 1-based. This approach is used by most Modbus implementations.

For example the register for the float test value is documented as a read-only word at address 501. This address could also be written as 30501 (based on a traditional Modbus addressing scheme mapping the functions to specific address ranges). The content of this register can be retrieved with the Read Input Registers function (function identifier „4“). The internal address of this value (as used for the on-wire format of Modbus) is 500 (due to the 1-based register addressing). This internal address is only used by very few Modbus client implementations. Most implementations use the 1-based address, instead.

Clients without support for address offsets may need to decrement every address (as documented here) when assembling the Mod-bus data frame.

3.4.3 Simple Data Types

The Modbus specification describes simple data types (bits and 16-bit-words). Additionally the following data types are used by the Modbus implementation of the colorSENSOR:

- Float values: two registers (32 bit), IEEE-754, big-endian word-order and byte-order.
- Integer values with 32 bit (two registers) or 64 bit (four registers): big-endian word-order and byte-order.
- Strings: the first word contains the length; all following bytes contain the ASCII characters. Each "word" register (after the length) contains two characters (first: upper, second: lower byte).Reading past the end of the string length is allowed and returns null bytes.Thus usually a trailing null byte is present at at the end of the string.But you may not rely on this, as the trailing null byte is missing, if the string uses exactlythe maximum number of allowed characters for this string.
- Bytes: a raw byte array is used for binary data transfers. Each register contains two characters (first: upper, second: lower byte).Reading past the end of the binary data is allowed and returns null bytes.The length of the raw data should be handled via a separate register.

- Bitmask: 16 bit words are used to represent or manipulate boolean fields. Each bit represents a single boolean value. The description of each bitmask data field maps bit positions to the boolean state described by this bit. A value of zero is considered to be “false” (not active). A value of one is true. The bit positions start with zero with the least significant bit.

3.5 Session State, Concurrency and Multiple Interfaces

Multiple interfaces of the sensor can communicate via the modbus protocol. Each hardware interface (e.g. RS232, USB) manages its own state. This is relevant for stateful operations (e.g. access to a collection), that require a sequence of read or write requests. The Ethernet interface accepts TCP connections. Each connection tracks its own state for the duration of the connection.

3.5.1 Functions

3.5.1.1 Autogain Procedure API Endpoint: `/sensor/detection-profiles/current/autogain`

Execute the autogain procedure in order to determine suitable sampling properties for the current optical environment. The resulting sampling setup is applied automatically. These new settings are in effect as soon as the response is sent. The success or failure of an autogain procedure can be verified as soon as the autogain_is_running flag is cleared.

<table>
<tr><th>Address</th><th>Type</th><th>Operation</th><th colspan="2">Description</th><th>FC</th></tr>
<tr><td>00020</td><td>Bit</td><td>write</td><td colspan="2">Start an autogain procedure</td><td>5, 15</td></tr>
<tr><td rowspan="5">00302</td><td rowspan="5">Bitmask</td><td rowspan="5">read</td><td colspan="2">Status of the most recently started autogain procedure</td><td rowspan="5">4</td></tr>
<tr><td><b>Position</b></td><td><b>Description</b></td></tr>
<tr><td>0</td><td>Is still running</td></tr>
<tr><td>1</td><td>Finished successfully</td></tr>
<tr><td>2</td><td>Failed: Target is too dark</td></tr>
<tr><td>00410</td><td>Float</td><td>read / write</td><td colspan="2">Minimum wanted sample rate</td><td>3, 4, 6, 16</td></tr>
<tr><td>00412</td><td>Float</td><td>read / write</td><td colspan="2">Target analog input level</td><td>3, 4, 6, 16</td></tr>
<tr><td>00414</td><td>Uint16</td><td>read / write</td><td colspan="2">Number of samples used for averaging</td><td>3, 4, 6, 16</td></tr>
<tr><td rowspan="4">00415</td><td rowspan="4">Bitmask</td><td rowspan="4">read / write</td><td colspan="2">Boolean flags for autogain procedure
Default value: 65535</td><td rowspan="4">3, 4, 6, 16</td></tr>
<tr><td><b>Position</b></td><td><b>Description</b></td></tr>
<tr><td>0</td><td>Enable internal emitter</td></tr>
<tr><td>1</td><td>Enable ambient tight compensation</td></tr>
<tr><td rowspan="5">00416</td><td rowspan="5">Bitmask</td><td rowspan="5">read / write</td><td colspan="2">Override default autogain settings with custom values</td><td rowspan="5">3, 4, 6, 16</td></tr>
<tr><td><b>Position</b></td><td><b>Description</b></td></tr>
<tr><td>0</td><td>Overwrite minimum wanted sample rate</td></tr>
<tr><td>1</td><td>Overwrite target analog input level</td></tr>
<tr><td>2</td><td>Overwrite number of samples used for averaging</td></tr>
</table>

3.5.1.2 White reference API Endpoint: `/sensor/detection-profiles/current/white-reference`

The white reference is used for calculating accurate color positions in the colorspaces. The factory default white reference is suitable for a special set of sensor and optics. A custom white reference can be sampled. A reference white target is recommended for this.

Address	Type	Operation	Description	FC
00021	Bit	write	Reset the custom white reference	5
00022	Bit	write	Sample a custom white reference	5

3.5.1.3 Add Color to Color Table API Endpoint: `/sensor/matchers`

Address	Type	Operation	Description	FC
00024	Bit	write	Create a new matcher and assign the current color position to it (as a detectable).	5,15
00451	Uint16	read	Retrieve the identifier of the most recently created matcher.	4

3.5.1.4 Manage Color Positions of a Color Group API Endpoint: `/sensor/matchers`

Each color group (matcher) may refer to one or more color positions (detectables).

Address	Type	Operation	Description	FC
00025	Bit	write	Add a new detectable to an existing matcher (color group).	5
00026	Bit	write	Delete all detectables of an existing matcher (color group).	5
00027	Bit	read	Indicate whether the currently selected matcher exists.	1
00311	Uint16	read	Current number of detectables (color positions) assigned to the matcher.	4

00450	Uint16	read / write	Specify the matcher (color group) when adding or removing detectables (color positions).	3, 6

3.5.1.5 Read Sensor Capabilities API Endpoint: `/sensor/capabilities`

Inspect the available features of the controller.

Address	Type	Operation	Description		FC
00300	Uint16	read	Number of available switching outputs		4
00301	Bitmask	read	Colorspaces supported by the sensor		4
			Position	**Description**	
			0	XYZ	
			1	L*a*b*	
			2	xyY	
			3	L*u*v*	
			4	L*u‘v‘	
00303	Bitmask	read	Available tolerance shapes		4
			Position	**Description**	
			0	Infinite (classification)	
			1	Sphere	
			2	Cylinder	
			3	Box	
00304	Bitmask	read	Available switching output drivers		4
			Position	**Description**	
			0	Disabled	
			1	NPN	
			2	PNP	
			3	Push-Pull	
00305	Float	read	Maximum sample rate		4
00307	Uint16	read	Maximum number of detectables		4
00308	Uint16	read	Maximum number of matchers		4

3.5.1.6 Get Current Sample API Endpoint: `/sensor/samples/current`

Retrieve the latest color detection sample. A single read operation covering the complete memory range of the sample is guaranteed to be consistent. Multiple read operations in series will probably result in a combination of values from the different samples gathered during the time between the first and the last request.

Address	Type	Operation	Description	FC
00150	Uint64	read	Timestamp of the current sample	4
00154	Float	read	Signal level of the current sample	4
00156	Float	read	Representation of the color in the XYZ colorspace (X)	4
00158	Float	read	Representation of the color in the XYZ colorspace (Y)	4
00160	Float	read	Representation of the color in the XYZ colorspace (Z)	4
00162	Float	read	Representation of the color in the currently active colorspace L	4
00164	Float	read	Representation of the color in the currently active colorspace a	4
00166	Float	read	Representation of the color in the currently active colorspace b	4
00168	Float	read	Representation of the color as RGB values red (between 0.0 and 1.0)	4
00170	Float	read	Representation of the color as RGB values green (between 0.0 and 1.0)	4
00172	Float	read	Representation of the color as RGB values blue (between 0.0 and 1.0)	4
00174	Uint16	read	Inputs with a high level event during the last sample period (bit 0 -> IN0)	4
00175	Uint16	read	Inputs with a low level event during the last sample period (bit 0 -> IN0)	4
00176	Uint16	read	Inputs with a rising edge event during the last sample period (bit 0 -> IN0)	4
00177	Uint16	read	Inputs with a falling edge event during the last sample period (bit 0 -> IN0)	4
00178	Uint16	read	ID of the closest matcher in range of the last sample‘s color position. The value 65535 is returned if the sampled color position was not in range of any of the available matchers.	4
00179	Uint16	read	Currently active state of the Switching Outputs (bit 0 -> OUT0)	4
00180	Float	read	Distance (based on the axes of the currently configured colorspace) between the last sampled color position and the closest suitable matcher (if any). A negative value (-1) indicates that no matcher is in range. Distance 1	4
00182	Float	read	Distance (based on the axes of the currently configured colorspace) between the last sampled color position and the closest suitable matcher (if any). A negative value (-1) indicates that no matcher is in range. Distance 2	4
00184	Float	read	Distance (based on the axes of the currently configured colorspace) between the last sampled color position and the closest suitable matcher (if any). A negative value (-1) indicates that no matcher is in range. Distance 3	4

3.5.1.7 Status of the Color Table API Endpoint: `/sensor/dtection-profiles/current`

Retrieve the current usage of the color table.

Address	Type	Operation	Description	FC
00309	Uint16	read	Current number of matchers (color groups) stored in the color table	4
00310	Uint16	read	Current number of detectables (color positions) stored in the color table	4

3.5.1.8 Clear Color Table API Endpoint: `/sensor/matchers`

Delete all colors that are stored in the color table.

Address	Type	Operation	Description	FC
00023	Bit	read	Remove all stored colors	5, 15

3.5.1.9 Switching Outputs Driver API Endpoint: `/peripherals/outputs`

Electrical output lines can drive external actors in different electrical modes. The currently active mode can be retrieved and changed.

<table>
<tr><th>Address</th><th>Type</th><th>Operation</th><th colspan="2">Description</th><th>FC</th></tr>
<tr><td rowspan="6">00400</td><td rowspan="6">Uint16</td><td rowspan="6">read / write</td><td colspan="2">Retrieve and change the current switching output driver.</td><td rowspan="6">3,4,6,16</td></tr>
<tr><td><b>Description</b></td><td><b>Values</b></td></tr>
<tr><td>off</td><td>0</td></tr>
<tr><td>npn</td><td>1</td></tr>
<tr><td>pnp</td><td>2</td></tr>
<tr><td>push-pull</td><td>3</td></tr>
</table>

3.5.1.10 Firmware Version API Endpoint: `/firmware`

Read information about the firmware.

Address	Type	Operation	Beschreibung	FC
00100	Uint16	read	Firmware Version (Major: X.0.0)	4
00101	Uint16	read	Firmware Version (Major: 0.X.0)	
00102	Uint16	read	Firmware Version (Major: 0.0.X)	

3.5.1.11 Device Information API Endpoint: `/device`

Read information about the device.

Address	Type	Operation	Description	FC
00103	String	read	Device serial	4
00114	String	read	Vendor of device	
00123	String	read	Device model	
00132	String	read	Device variant	

3.5.1.12 Configure access lock API Endpoint

Lock or unlock certain methods of accessing the sensor.

<table>
<tr><th>Address</th><th>Type</th><th>Operation</th><th colspan="2">Description</th><th>FC</th></tr>
<tr><td rowspan="5">00460</td><td rowspan="5">Bitmask</td><td rowspan="5">read / write</td><td colspan="2">Lock or unlock certain access actions</td><td rowspan="5">3, 4, 6, 16</td></tr>
<tr><td><b>Position</b></td><td><b>Description</b></td></tr>
<tr><td>0</td><td>Lock keypad (ignore any keypress event)</td></tr>
<tr><td>1</td><td>Reject read access for the API</td></tr>
<tr><td>2</td><td>Reject write access for the API</td></tr>
</table>

3.5.1.13 Manage API users API Endpoint: `/access/user`

Manage the user accounts used by the HTPI API.

Address	Type	Operation	Description	FC
00028	Bit	write	Remove all existing API users (i.e. disable API access control).	1, 5, 15
00461	Uint16	read	Number of API users	4

3.5.1.14 Settings Reset API Endpoint: `/settings`

Reset the controller settings to their factory defaults.

Address	Type	Operation	Description	FC
00006	Bit	write	Reset all settings	5

3.5.1.15 Factory Reset API Endpoint: `/system/factory-reset`

Reset the controller firmware to its factory default and initiate a reboot.

Address	Type	Operation	Description	FC
00002	Bit	write	Trigger a factory reset of the firmware and the settings	5

3.5.1.16 Reboot the Device API Endpoint: `/system/reboot`

Trigger a reboot of all controller components.

Address	Type	Operation	Description	FC
00001	Bit	write	Trigger a reboot	5

3.5.1.17 Upgrade Recovery Firmware API Endpoint: `/system/factory-reset`

Replace the stored recovery image with the current system firmware. This is helpful if you want to update the recovery image to a more recent firmware version.

Address	Type	Operation	Description	FC
00003	Bit	write	Upgrade the recovery firmware to the currently running firmware version	5

3.5.1.18 RS232 Interface Configuration API Endpoint: `/peripherals/rs232`

Inspect or change the settings address of the controller for the RS232 interface. Some settings refer to the Modbus slave protocol. The Modbus slave ID is used for serial communication if more than one Modbus device is connected to the same bus. The frame format may be changed according to the needs of the Modbus master.

Address	Type	Operation	Description		FC
00430	Uint16	read / write	Baud rate of RS232 interface		3,4,6,16
			Werte	**Description**	
			9600	0	
			19200	1	
			115200	2	
00431	Uint16	read / write	Protocol to be used for the RS232 interface		3,4,6,16
			Values	**Description**	
			eliza	0	
			modbus	1	
00432	Uint16	read / write	Slave ID to be used for the Modbus protocol (1..247)		3,4,6,16
00433	Uint16	read / write	Frame format to be used for the Modbus protocol possible values		3,4,6,16
			Values	**Description**	
			rtu	0	
			ascii	1	

3.5.1.19 USB Interface Configuration API Endpoint: `/peripherals/usb`

Inspect or change the settings address of the controller for the USB interface. Some settings refer to the Modbus slave protocol. The Modbus slave ID is used for serial communication if more than one Modbus device is connected to the same bus. The frame format may be changed according to the needs of the Modbus master.

Adress	Type	Operation	Description		FC
00440	Uint16	read / write	Protocol to be used for the USB interface possible values		3, 4, 6, 16
			Values	**Description**	
			eliza	0	
			modbus	1	

<table>
<tr><td>00441</td><td>Uint16</td><td>read / write</td><td colspan="2">Slave ID to be used for the Modbus protocol (1..247)</td><td>3, 4, 6, 16</td></tr>
<tr><td rowspan="4">00442</td><td rowspan="4">Uint16</td><td rowspan="4">read / write</td><td colspan="2">Frame format to be used for the Modbus Protocol possible values</td><td rowspan="4">3, 4, 6, 16</td></tr>
<tr><td><b>Values</b></td><td><b>Description</b></td></tr>
<tr><td>rtu</td><td>0</td></tr>
<tr><td>ascii</td><td>1</td></tr>
</table>

3.5.1.20 Data Format Test API Endpoint: None

Some registers respond with specified fixed values in order to allow clients to verify the correctness of the configured data format easily.

Address	Type	Operation	Description	FC
00500	Uint16	read	A 16 bit integer value containing the number 1234.	4
00501	Float	read	A float value containing the number -1.0.	4
00503	Uint32	read	A 32 bit integer value containing the number 12345678.	4
00505	Uint64	read	A 64 bit integer value containing the number 123456789012.	4

MICRO-EPSILON

MICRO-EPSILON Eltrotec GmbH
Manfred-Wörner-Straße 101 · 73037 Göppingen / Germany
Tel. +49 (0) 7161 / 98872-300 · Fax +49 (0) 7161 / 98872-303
info@micro-epsilon.com · www.micro-epsilon.com
Your local contact: www.micro-epsilon.com/contact/worldwide/

X9751375.01-A021012HDR