# Sensorlab.io API documentation Documentation

**Sensorlab.io**

**Oct 08, 2018**

# Contents

CHAPTER 1

# Sensorlab.io REST API

## 1.1 Basic information

REST API provides number of endpoints to work with sensorlab.io

You work with your sensors, applications, measurements, etc via REST API using your own user credentials or your can work with sensors and measurements using application credentials.

REST API exchange information with JSON objects. Result depends on endpoint type and HTTP status.

Here are the main HTTP statuses you should handle:

- *200* - Normal response. Result depends on endpoint type.
- *400* - This status will appear if your JSON object in request is malformed.

Example:

```
HTTP/1.1 400 Bad Request
Content-Type: applications/json

{
    "success": false,
    "code": 400,
    "message": "Please, check data you send and try again"
}
```

- *401* - REST API returns this status if there's problem with authentication with token or authentication credentials are wrong.

Example:

```
HTTP/1.1 401 Unauthorized
Content-Type: applications/json

{
```

(continues on next page)

(continued from previous page)

```
    "success": false,
    "code": 401,
    "message": "Unauthorized"
}
```

- *403* - REST API returns this status if there's no access to the requested object (for example, sensor is not public and you are trying to access it using only public api key)

Example:

```
HTTP/1.1 403 Forbidden
Content-Type: text/javascript

{
    "success": false,
    "code": 403,
    "message": "Sensor measurements are not public"
}
```

- *404* - REST API will return this error if there's no requested endpoint or object for this endpoint doesn't exist.

You can identify is it missing endpoint or object by parameter *type*. In case of *object* value for *type* there will be *object_type* field that will help to detect what type of object is missing.

Missing endpoint example:

```
HTTP/1.1 404 Not Found
Content-Type: applications/json

{
    "success": false,
    "code": 404,
    "type": "endpoint",
    "message": "Endpoint doesn't exist"
}
```

Missing object example:

```
HTTP/1.1 404 Not Found
Content-Type: applications/json

{
    "success": false,
    "code": 404,
    "type": "object",
    "object_type": "sensor",
    "message": "Sensor not found"
}
```

Object types available:

- user
- sensor
- application
- measurement
- sensor_alert

- *422* - validation error. Some endpoints require correct request params or correct fields in JSON object.

Some fields can be required or require special format. All those errors will be available in the *errors* field.

Example:

```
HTTP/1.1 422 Unprocessable Entity
Content-Type: applications/json

{
    "success": false,
    "code": 422,
    "message": "There are validation errors found.",
    "errors": [
        {
            "code": 3,
            "message": "`threshold_type` is required",
            "param": "threshold_type"
        }
    ]
}
```

- *500* - If there are any problems with API you will get response with this status.

Example:

```
HTTP/1.1 500 Internal Server Error
Content-Type: applications/json

{
    "error": true,
    "status": 500,
    "message": "Internal error, please, try again"
}
```

## 1.2 REST API Endpoints

### 1.2.1 Authentication endpoints

#### Get user authentication token

**POST /api/v1/auth/user/token**

**Request**:

```
POST /api/v1/auth/user/token HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json

{
  "email": "test@email.com",
  "password": "password"
}
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↪eyJfaWQiOiI1YWJhMTY1ZTQ4MmJlYzJkZjg4N2M2YTMiLCJpYXQiOjE1MjIxNDY0MTYsImV4cCI6MTUyMjIzMjgxNn0.
↪-6kJm1Rbd_SPbuwc6kg6FHuJnUii8FtKI9DXR0J5-Ig"
}
```

**Request Headers**

- Content-Type – application/json

**Status Codes**

- 200 OK – No errors.
- 401 Unauthorized – Wrong authorization credentials
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- code=1 - field=email - Please, provide email. This cannot be empty
- code=2 - field=password - Please, provide password to get token
- code=3 - field=email - Email is invalid
- code=4 - field=email - User does not exist
- code=5 - field=email - Email is not verified
- code=6 - field=password - Password is incorrect

---

**Note:** Not every endpoint is available for application token. You will find more information in the endpoint description itself.

---

### Get application authentication token

**POST /api/v1/auth/application/token**

**Request**:

```
POST /api/v1/auth/application/token HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json

{
  "public_api_key": "sensorlab:application:62fa02f38ff6100dbbd1cdff2339ccf3",
  "private_api_key": "9a57d7e74ead4ec610539dcf5124db0d"
}
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json
```

(continues on next page)

(continued from previous page)

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
↪eyJfaWQiOiI1YWJhMTY1ZTQ4MmJlYzJkZjg4N2M2YTMiLCJpYXQiOjE1MjIxNDY0MTYsImV4cCI6MTUyMjIzMjgxNn0.
↪-6kJm1Rbd_SPbuwc6kg6FHuJnUii8FtKI9DXR0J5-Ig"
}
```

**Request Headers**

- Content-Type – application/json

**Status Codes**

- 200 OK – No errors.
- 401 Unauthorized – Wrong authorization credentials
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- code=1 - field=public_api_key - Please, provide public api key. This cannot be empty
- code=2 - field=private_api_key - Please, provide private api key. This cannot be empty

---

**Note:** Not every endpoint is available for application token. You will find more information in the endpoint description itself.

---

There are two main authentication endpoints - one for user (you) and one for applications.

You will retrieve a JSON Web Token (JWT) which you will pass later on to other endpoints.

With user auth you can work with most endpoints - your profile, application, sensors, measurements. For example, you can create you own custom dashboard with any programming language you prefer and use you email and password to access that dashboard.

Application authentication has access to sensors, connected to application and measurements of those sensors.

You can create as many different applications you want and use only Public and Private API keys to authenticate applications - absolutely no need to pass email or password from your account.

### 1.2.2 Profile endpoints

#### Get profile

**`POST /api/v1/profile`**

**Synopsis** Get user profile. This can also be used to check authorization token.

**Request**:

```
GET /api/v1/profile HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "email": "test@test.com"
}
```

**Request Headers**

- Authorization – "Bearer: {token}" from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors.
- 401 Unauthorized – Wrong authorization token

---

**Note:** Available for:

- User token

---

### Change password

**`POST /api/v1/profile/change_password`**

**Synopsis** Change password for authenticated user.

**Request**:

```
GET /api/v1/profile/change_password HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json

{
    "old_password": "old_password",
    "new_password": "new_password",
    "new_password_check": "new_password"
}
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "success": true,
    "code": 100,
    "message": "New password is set for user."
}
```

**Validation error response**

```
HTTP/1.1 422 OK
Content-type: application/json

{
    "success": false,
    "code": 422,
    "message": "There are validation errors found.",
    "errors": [
        {
            "code": 1,
            "message": "Please, provide old password. This cannot be empty.",
            "param": "old_password"
        },
        {
            "code": 2,
            "message": "You must provide new password.",
            "param": "new_password"
        },
        {
            "code": 3,
            "message": "You must provide new password check.",
            "param": "new_password_check"
        }
    ]
}
```

**Request Headers**

- Authorization – "Bearer: {token}" from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors.
- 401 Unauthorized – Wrong authorization token
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- code=1 - field=old_password - Please, provide old password. This cannot be empty
- code=2 - field=new_password - You must provide new password
- code=3 - field=new_password_check - You must provide new password check
- code=4 - field=old_password - Password is incorrect. Please provide you current password
- code=5 - field=new_password_check - Both "new password" and "new password check" values must be equal

---

**Note:** Available for:

- User token

---

### 1.2.3 Applications endpoints

#### Get list of applications

**GET /api/v1/applications**

**Request**:

```
GET /api/v1/applications HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "result": [
        {
            "id": "d8d7f14a-9a16-11e8-9eb6-529269fb1459",
            "name": "Sensors Application 1",
            "description": "",
            "created": "2018-05-09T10:20:01.352Z",
            "public_api_key":
↪"sensorlab:application:62fa02f38ff6100dbbd1cdff2339ccf3",
        },
        {
            "id": "f6ca0a08-9a16-11e8-9eb6-529269fb1459",
            "name": "Sensors Application 2",
            "description": "Application description",
            "created": "2018-05-09T10:19:59.363Z",
            "public_api_key":
↪"sensorlab:application:cc7fddc2434a1773a360c84b1be71b16"
        }
    ],
    "count": 27,
    "pages": 1
}
```

**Query Parameters**

- **page** – used for pagination. Default is 1.
- **name** – search by name.
- **sort** – sorting parameter.
- **show_deleted** – if *true* - show deleted only applications.

  You must provide string with sorting field name and sorting type like this:

  *type,asc*

  **Available fields:**

  – *name* - sort by creation date

  – *created* - sort by measurement type

  **Available sort types:**

- *asc* - Ascending sort
- *desc* - Descending sort

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Response JSON Object**

- **id** (*string*) – Application ID.
- **name** (*string*) – Application name.
- **description** (*string*) – Application description.
- **public_api_key** (*string*) – Public API key.

**Status Codes**

- 200 OK – No errors, will return result with applications list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- code=1 - *page* must be an integer

---

**Note:** Available for:

- User token

---

### Get application

**GET /api/v1/applications/** (*id*)

**Request**:

```
GET /api/v1/applications/d8d7f14a-9a16-11e8-9eb6-529269fb1459 HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "id": "d8d7f14a-9a16-11e8-9eb6-529269fb1459",
    "name": "Sensors Application",
    "description": "Sensor Description",
    "created": "2018-05-09T10:20:01.352Z",
    "public_api_key": "sensorlab:application:62fa02f38ff6100dbbd1cdff2339ccf3"
 }
```

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Response JSON Object**

- **id** (*string*) – Application ID.
- **name** (*string*) – Application name.
- **description** (*string*) – Application description.
- **public_api_key** (*string*) – Public API key.

**Status Codes**

- 200 OK – No errors, will return result with applications list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 404 Not Found – Application not found

---

**Note:** Available for:

- User token

---

### Create application

**POST /api/v1/applications**

**Request**:

```
POST /api/v1/applications HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json

{
    "name": "Application Name",
    "description": "Application Description"
}
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "success": true,
    "code": 100,
    "message": "Application created.",
    "application": {
        "id": "7e970572-9a16-11e8-9eb6-529269fb1459",
        "name": "Application Name",
        "description": "Application Description",
        "public_api_key": "sensorlab:application:62fa02f38ff6100dbbd1cdff2339ccf3
↪",
        "private_api_key": "146d3ac9d9900c8151a73ca09b39ca7f"
    }
}
```

**Request JSON Object**

- **name** (*string*) – Name for your application
- **description** (*string*) – Description for your application

**Response JSON Object**

- **application** (*object*) – Application data.
- **application.id** (*string*) – Application ID.
- **application.name** (*string*) – Application name.
- **application.description** (*string*) – Application description.
- **application.public_api_key** (*string*) – Public API key.
- **application.private_api_key** (*string*) – Generated private API key. Please note, that Private API Key will appear only once after app creation. You will be able to generate new one though.

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with applications list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- *code=1 - Please, provide name field. This cannot be empty*

---

**Note:** Available for:

- User token

---

### Update application

**PATCH /api/v1/applications/** (*id*)

**Request**:

```
PATCH /api/v1/applications/f6ca0a08-9a16-11e8-9eb6-529269fb1459 HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json

{
    "name": "Updated Application Name",
    "description": "Updated Application Description"
}
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "success": true,
    "code": 100,
    "message": "Application updated.",
}
```

**Request JSON Object**

- **name** (*string*) – Name for your application
- **description** (*string*) – Description for your application

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with applications list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 404 Not Found – Application not found.
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- *code=1 - Please, provide name field. This cannot be empty.*

---

**Note:** Available for:

- User token

---

### Delete application

**GET /api/v1/applications/** (*id*)

**Request**:

```
DELETE /api/v1/applications/93cf3f40-9a16-11e8-9eb6-529269fb1459 HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "success": true,
    "code": 100,
    "message": "Application deleted."
}
```

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with applications list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 404 Not Found – Application not found

---

**Note:** We don't actually delete any data, just hide it.

---

**Note:** Available for:

- User token

---

### Generate new Private Api Key for application

**POST /api/v1/applications/** (*id*) **/private_api_key/generate**

**Request**:

```
POST /api/v1/applications/d8d7f14a-9a16-11e8-9eb6-529269fb1459/private_api_key/
↪generate HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "success": true,
    "code": 100,
    "message": "Application created.",
    "application": {
        "id": "d8d7f14a-9a16-11e8-9eb6-529269fb1459",
        "name": "Application Name",
        "description": "Application Description",
        "public_api_key": "sensorlab:application:62fa02f38ff6100dbbd1cdff2339ccf3
↪",
        "private_api_key": "146d3ac9d9900c8151a73ca09b39ca7f"
    }
}
```

**Response JSON Object**

- **application** (*object*) – Application data.
- **application.id** (*string*) – Application ID.
- **application.name** (*string*) – Application name.

- **application.description** (*string*) – Application description.
- **application.public_api_key** (*string*) – Public API key.
- **application.private_api_key** (*string*) – Generated private API key. Please note, that Private API Key will appear only once after app creation. You will be able to generate new one though.

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with applications list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- *code=1 - Please, provide name field. This cannot be empty.*

---

**Note:** Available for:

- User token

---

### 1.2.4 Sensors Endpoints

#### Get sensors list

**GET /api/v1/sensors**

**Request**:

```
GET /api/v1/sensors HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "result": [
        {
            "id": "8dbc5460-9a18-11e8-9eb6-529269fb1459",
            "imei": "863911091619316",
            "name": "Esta",
            "application": "9c5b5c32-9a18-11e8-9eb6-529269fb1459",
            "batteryCharge": 27.7169,
            "isBatteryCharging": false,
            "isOnline": true
        },
        {
```

(continues on next page)

(continued from previous page)


```
            "id": "876cc47b-5379-40de-83d0-10cf96720566",
            "imei": "980098461327809",
            "name": "Christelle",
            "application": "9c5b5fde-9a18-11e8-9eb6-529269fb1459",
            "batteryCharge": 6.7315,
            "isBatteryCharging": false,
            "isOnline": true
        }
    ],
    "count": 200,
    "pages": 4
}
```


**Response JSON Object**

- **id** (*string*) – Sensors's ID.
- **imei** (*string*) – Sensor's IMEI.
- **name** (*string*) – Sensor's name.
- **application** (*string*) – Application ID sensor is connected to.
- **batteryCharge** (*number*) – Sensor's battery charge in percent.
- **isBatteryCharging** (*boolean*) – Indicates if battery is charging or not.
- **isOnline** (*boolean*) – Indicates if sensor is online and sending data or not.
- **is_public** (*boolean*) – Show if sensor is public or not.

**Query Parameters**

- **page** – used for pagination. Default is 1.
- **name** – filter by name. Search is case-insensitive and searches using *like*.
- **imei** – filter by imei. Will search by exact match.
- **id** – filter by id. Will search by exact match.
- **online_status** – pass "online" to search for online sensors or "offline" for offline sensors.
- **battery_charge_min** – filter sensors by battery charge
- **battery_charge_max** – filter sensors by battery charge
- **sort** – sorting parameter.

  You must provide string with sorting field name and sorting type like this:

  *name,asc*

  **Available fields:**

  – *name* - sort by sensor name

  **Available sort types:**

  – *asc* - Ascending sort

  – *desc* - Descending sort

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with sensors list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- code=1 - Sensor must be correct UUID format
- code=2 - *online_status* parameter must be *online* or *offline*
- code=3 - *battery_charge_min* must be an integer
- code=4 - *battery_charge_max* must be an integer
- code=5 - *battery_charge_max* should not be less than *battery_charge_min*
- code=6 - *page* must be an integer
- code=7 - *next* must be correct UUID format

---

**Note:** Available for:

- User token
- Application token

Application token will have access only to sensors assigned to this application.

---

### Get single sensor

**GET /api/v1/sensors/**(*id*)

**Request**:

```
GET /api/v1/sensors/7765e032-9a18-11e8-9eb6-529269fb1459 HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "id": "7765e032-9a18-11e8-9eb6-529269fb1459",
    "imei": "863911091619316",
    "name": "Esta",
    "application": "8078b94c-9a18-11e8-9eb6-529269fb1459",
    "batteryCharge": 6.7315,
    "isBatteryCharging": false,
    "isOnline": true,
    "is_public": false,
}
```

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Response JSON Object**

- **id** (*string*) – Sensors's ID.
- **imei** (*string*) – Sensor's IMEI.
- **name** (*string*) – Sensor's name.
- **application** (*string*) – Application ID sensor is connected to.
- **batteryCharge** (*number*) – Sensor's battery charge in percent.
- **isBatteryCharging** (*boolean*) – Indicates if battery is charging or not.
- **isOnline** (*boolean*) – Indicates if sensor is online and sending data or not.
- **is_public** (*boolean*) – Show if sensor is public or not.

**Status Codes**

- 200 OK – No errors, will return result with sensors list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 404 Not Found – Sensor not found

---

**Note:** Available for:

- User token
- Application token

Application token will have access only to sensors assigned to this application.

---

### Update sensor

**PATCH /api/v1/sensors/** (*id*)

**Request**:

```
PATCH /api/v1/sensors/acdfab8a-9a18-11e8-9eb6-529269fb1459 HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json

{
    "name": "Updated Sensor ID",
    "application": "b2a2fdb0-9a18-11e8-9eb6-529269fb1459"
}
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "success": true,
```

(continues on next page)

(continued from previous page)

```
    "code": 100,
    "message": "Sensor updated.",
}
```

**Request JSON Object**

- **name** (*string*) – Name for your sensor.
- **application** (*string*) – Application's ID to connect sensor too.
- **is_public** (*boolean*) – Set sensor public or private (true or false).

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with applications list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 404 Not Found – Sensor not found.
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- code=1 - field=name - Please, provide name field. This cannot be empty
- code=2 - field=application - *application* must be correct UUID format
- code=3 - field=is_public - *is_public* must be correct boolean format

---

**Note:** Available for:

- User token
- Application token

Application token will have access only to sensors assigned to this application.

---

### Get measurements for sensor

**GET /api/v1/sensors/**(*id*)**/measurements**

**Request**:

```
GET /api/v1/sensors/4634da2c-9a18-11e8-9eb6-529269fb1459/measurements HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
```

(continues on next page)

(continued from previous page)


```
    "result": [
        {
            "type": "KVF",
            "value": [
                3.896,
                4.037,
                5.42
            ],
            "timestamp": "1533627864"
        },
        {
            "type": "TTA",
            "value": [
                4.465,
                4.126,
                4.535
            ],
            "timestamp": "1533627865"
        },
    ],
    "next": "d438a4b2-9a17-11e8-9eb6-529269fb1459",
    "prev": null
}
```


**Query Parameters**

- **next** – Use *next* or *prev* fields from response to get next or previous page.
- **type** – filter by type.

**Response JSON Object**

- **result.type** (*string*) – Measurement type.
- **result.value** (*string*) – Array of values.
- **result.timestamp** (*string*) – Timestamp for measurement.
- **next** (*string*) – This param shows next measurement ID for next page. Use it with *next* query parameter.
- **prev** (*string*) – This param shows prev measurement ID for prev page. Use it with *next* query parameter.
- **timestamp_start** (*string*) – Filter by timestamp range.
- **timestamp_stop** (*string*) – Filter by timestamp range.

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with sensors list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- code=2 - field=sensor_id - This is not correct id format
- code=3 - field=next - This is not correct id format
- code=4 - field=timestamp_start - *timestamp_start* should be correct unix timestamp format
- code=5 - field=timestamp_stop - *timestamp_stop* should be correct unix timestamp format
- code=6 - field=timestamp_stop - *timestamp_stop* should be more or equal *timestamp_start*

---

**Note:** Available for:

- User token
- Application token

Application token will have access only to measurements of sensors assigned to this application.

---

### Get last measurement for sensor

**GET /api/v1/sensors/**(*id*)**/measurements/last**

**Request**:

```
GET /api/v1/sensors/0028fcf0-9c90-11e8-8ee7-d12e1783ec90/measurements/last HTTP/1.
↪1
Host: staging.sensorlab.io
Accept: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-Type: text/javascript

{
    "type": "UDX",
    "value": [
        6.625,
        8.893,
        9.414
    ],
    "timestamp": 1533627864
}
```

**Query Parameters**

- **type** – filter by type.

**Response JSON Object**

- **type** (*string*) – Measurement type.
- **value** (*string*) – Array of values.
- **timestamp** (*string*) – Timestamp for measurement.

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with sensors list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.

---

**Note:** Available for:

- User token
- Application token

Application token will have access only to measurements of sensors assigned to this application.

---

### 1.2.5 Measurements endpoints

#### Get list of measurements

**`GET /api/v1/measurements`**

**Request**:

```
GET /api/v1/measurements?sensor_id=5ab8b113fc10152c70cdeb65 HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "result": [
        {
            "type": "KVF",
            "value": [
                3.896,
                4.037,
                5.42
            ],
            "timestamp": "1533627864"
        },
        {
            "type": "TTA",
            "value": [
                4.465,
                4.126,
                4.535
            ],
            "timestamp": "1533627865"
        },
    ],
    "next": "d438a4b2-9a17-11e8-9eb6-529269fb1459",
    "prev": null
}
```

**Query Parameters**

- **sensor_id** – Sensor's ID.
- **next** – Use *next* or *prev* fields from response to get next or previous page.
- **type** – filter by type.

**Response JSON Object**

- **result.type** (*string*) – Measurement type.
- **result.value** (*array*) – Array of values.
- **result.timestamp** (*number*) – Timestamp for measurement.
- **next** (*string*) – This param shows next measurement ID for next page. Use it with *next* query parameter.
- **prev** (*string*) – This param shows prev measurement ID for prev page. Use it with *next* query parameter.

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with sensors list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- code=1 - field=sensor_id - Please, provide sensor field. This cannot be empty
- code=2 - field=sensor_id - This is not correct id format
- code=3 - field=next - This is not correct id format
- code=4 - field=timestamp_start - *timestamp_start* should be correct unix timestamp format
- code=5 - field=timestamp_stop - *timestamp_stop* should be correct unix timestamp format
- code=6 - field=timestamp_stop - *timestamp_stop* should be more or equal *timestamp_start*

---

**Note:** Available for:

- User token
- Application token

Application token will have access only to measurements of sensors assigned to this application.

---

### Get last measurement

**GET /api/v1/measurements/last**
**Request**:

```
GET /api/v1/measurements/last?sensor_id=11e34426-9a17-11e8-9eb6-529269fb1459 HTTP/
↪1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "type": "UDX",
    "value": [
        6.625,
        8.893,
        9.414
    ],
    "timestamp": 1533627864
}
```

**Query Parameters**

- **sensor_id** – Sensor's ID.
- **type** – filter by type.

**Response JSON Object**

- **type** (*string*) – Measurement type.
- **value** (*string*) – Array of values.
- **timestamp** (*string*) – Timestamp for measurement.

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with sensors list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- code=1 - field=sensor_id - Please, provide sensor field. This cannot be empty
- code=2 - field=sensor_id - This is not correct id format

---

**Note:** Available for:

- User token
- Application token

Application token will have access only to measurements of sensors assigned to this application.

---

### 1.2.6 Measurements for public sensor endpoints

#### Get measurements for sensor

**GET /api/v1/sensors/**(*id*)**/measurements**

**Request**:

```
GET /api/v1/sensors/4634da2c-9a18-11e8-9eb6-529269fb1459/measurements HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "result": [
        {
            "type": "KVF",
            "value": [
                3.896,
                4.037,
                5.42
            ],
            "timestamp": "1533627864"
        },
        {
            "type": "TTA",
            "value": [
                4.465,
                4.126,
                4.535
            ],
            "timestamp": "1533627865"
        },
    ],
    "next": "d438a4b2-9a17-11e8-9eb6-529269fb1459",
    "prev": null
}
```

**Query Parameters**

- **next** – Use *next* or *prev* fields from response to get next or previous page.
- **type** – filter by type.

**Response JSON Object**

- **result.type** (*string*) – Measurement type.
- **result.value** (*string*) – Array of values.
- **result.timestamp** (*string*) – Timestamp for measurement.
- **next** (*string*) – This param shows next measurement ID for next page. Use it with *next* query parameter.
- **prev** (*string*) – This param shows prev measurement ID for prev page. Use it with *next* query parameter.

- **timestamp_start** (*string*) – Filter by timestamp range.
- **timestamp_stop** (*string*) – Filter by timestamp range.

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with sensors list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 404 Not Found – Sensor is not found.
- 403 Forbidden – Sensor is not public.
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- code=2 - field=sensor_id - This is not correct id format
- code=3 - field=next - This is not correct id format
- code=4 - field=timestamp_start - *timestamp_start* should be correct unix timestamp format
- code=5 - field=timestamp_stop - *timestamp_stop* should be correct unix timestamp format
- code=6 - field=timestamp_stop - *timestamp_stop* should be more or equal *timestamp_start*

---

**Note:** Available for:

- User token
- Application token

Application token will have access only to measurements of sensors assigned to this application.

---

### Get last measurement for public sensor

**GET /api/v1/public/sensors/** (*id*) **/measurements/last**

**Request**:

```
GET /api/v1/public/sensors/0028fcf0-9c90-11e8-8ee7-d12e1783ec90/measurements/last␣
↪HTTP/1.1
Host: staging.sensorlab.io
Accept: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-Type: text/javascript

{
    "type": "UDX",
    "value": [
        6.625,
```

(continues on next page)

(continued from previous page)

```
        8.893,
        9.414
    ],
    "timestamp": 1533627864
}
```

**Query Parameters**

- **type** – filter by type.

**Response JSON Object**

- **type** (*string*) – Measurement type.
- **value** (*string*) – Array of values.
- **timestamp** (*string*) – Timestamp for measurement.

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with sensors list.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 404 Not Found – Sensor is not found.
- 403 Forbidden – Sensor is not public.

---

**Note:** Available for:

- User token
- Application token

Application token will have access only to measurements of sensors assigned to this application.

---

## 1.2.7 Sensor alerts endpoints

### Create sensor alert configuration

**POST /api/v1/sensor/** (*sensor*) **/alerts**

**Request**:

```
POST /api/v1/sensor/8dbc5460-9a18-11e8-9eb6-529269fb1459/alerts HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json

{
    "threshold_type": "min",
    "measurement_type": "HUM",
    "threshold_value": 65.25
}
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "success": true,
    "code": 100,
    "message": "Sensor alert created",
    "sensor_alert": {
        "id": "b5550190-c63a-11e8-98c9-cf732fa3c579",
        "threshold_type": "min",
        "measurement_type": "HUM",
        "threshold_value": 65.25
    }
}
```

**Example request with threhsold_type=loc**:

```
POST /api/v1/sensor/8dbc5460-9a18-11e8-9eb6-529269fb1459/alerts HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json

{
    "threshold_type": "loc",
    "measurement_type": "LOC",
    "threshold_value": {
        "lat": 36.1812440939046,
        "lng": -101.84828589116029,
        "radius": 1000
    }
}
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "success": true,
    "code": 100,
    "message": "Sensor alert created",
    "sensor_alert": {
        "id": "b5550190-c63a-11e8-98c9-cf732fa3c579",
        "threshold_type": "loc",
        "measurement_type": "LOC",
        "threshold_value": {
            "lat": 36.1812440939046,
            "lng": -101.84828589116029,
            "radius": 1000
        }
    }
}
```

**Request JSON Object**

- **threshold_type** (*string*) – Threshold type.

  Possible values:

- *min* - set minimum value and alert will be generated if value from sensor will be lower than this value
- *max* - set maximum value and alert will be generated if value from sensor will be higher than this value
- *loc* - threshold type for GPS coordinates. If location is out of radius from specified point - alert will be generated

- **measurement_type** (*string*) – Measurement type to check.
- **threshold_value** (*string*) – Threshold value to compare to.

  This parameter depends on the threshold_type.

  - for type *min* value should be a number.
  - for type *max* value should be a number.
  - for type *loc* value should be a JSON object with lat, lng and radius parameters. Example: {lat: 36.1812440939046, lng: -101.84828589116029, radius: 1000}

**Response JSON Object**

- **sensor_alert** (*object*) – New sensor alert object.
- **sensor_alert.id** (*string*) – Sensor alert ID.
- **sensor_alert.threshold_type** (*string*) – Threshold type.
- **sensor_alert.measurement_type** (*string*) – Measurement type.
- **sensor_alert.threshold_value** (*string*) – Threshold value.

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with sensor_alert.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- code=2 - field=sensor - *sensor* field should be a correct UUID format
- code=3 - field=threshold_type - *threshold_type* is required
- code=4 - field=threshold_type - *threshold_type* must be *min*, *max* or *loc*
- code=5 - field=measurement_type - *measurement_type* is required
- code=6 - field=threshold_value - *threshold_value* is required
- code=7 - field=threshold_value - *threshold_value* for *threshold_type* “min” must be a correct float
- code=8 - field=threshold_value - *threshold_value* for *threshold_type* “max” must be a correct float
- code=9 - field=threshold_value - *threshold_value* for *threshold_type* “loc” must be a correct JSON object with *lat*, *lng* and *radius* parameters
- code=10 - field=threshold_value - *threshold_value.lat* must be correct latitude

- code=11 - field=threshold_value - *threshold_value.lng* must be correct longitude
- code=12 - field=threshold_value - *threshold_value.radius* must be correct positive integer

---

**Note:** Available for:

- User token
- Application token

---

### List sensor alerts configurations

**GET /api/v1/sensor/** (*sensor*) **/alerts**

**Request**:

```
GET /api/v1/sensor/8dbc5460-9a18-11e8-9eb6-529269fb1459/alerts HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

[
    {
        "id": "6cc44ad0-c635-11e8-bf95-1dddf61566fa",
        "threshold_type": "min",
        "measurement_type": "TMP",
        "threshold_value": -15
    },
    {
        "id": "6cfcbff0-c635-11e8-bf95-1dddf61566fa",
        "threshold_type": "max",
        "measurement_type": "HUM",
        "threshold_value": 80
    },
    {
        "id": "b5550190-c63a-11e8-98c9-cf732fa3c579",
        "threshold_type": "loc",
        "measurement_type": "LOC",
        "threshold_value": {
            "lat": 36.1812440939046,
            "lng": -101.84828589116029,
            "radius": 1000
        }
    }
]
```

**Response JSON Object**

- **id** (*string*) – Sensor alert ID.
- **threshold_type** (*string*) – Sensor alert threshold type.
- **measurement_type** (*string*) – Sensor alert measurement type.
- **threshold_value** (*mixed*) – Sensor alert value.

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with sensor_alert.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.

**Possible validation errors and codes:**

- code=2 - field=sensor - *sensor* field should be a correct UUID format

---

**Note:** Available for:

- User token
- Application token

---

### Get sensor alert configuration

**GET /api/v1/sensor/** (*sensor*) **/alerts/**
*id* **Request**:

```
GET /api/v1/sensor/8dbc5460-9a18-11e8-9eb6-529269fb1459/alerts/6cc44ad0-c635-11e8-
↪bf95-1dddf61566fa HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "id": "6cc44ad0-c635-11e8-bf95-1dddf61566fa",
    "threshold_type": "min",
    "measurement_type": "TMP",
    "threshold_value": -15
}
```

**Response JSON Object**

- **id** (*string*) – Sensor alert ID.
- **threshold_type** (*string*) – Sensor alert threshold type.
- **measurement_type** (*string*) – Sensor alert measurement type.
- **threshold_value** (*mixed*) – Sensor alert value.

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with sensor_alert.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.

**Possible validation errors and codes:**

- code=2 - field=sensor - *sensor* field should be a correct UUID format
- code=4 - field=id - *id* field should be a correct UUID format

---

**Note:** Available for:

- User token
- Application token

---

### Delete sensor alert configuration

**DELETE /api/v1/sensor/**(*id*)**/alerts/**

*id* **Request**:

```
DELETE /api/v1/sensor/8dbc5460-9a18-11e8-9eb6-529269fb1459/alerts/6cc44ad0-c635-
↪11e8-bf95-1dddf61566fa HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "success": true,
    "code": 100,
    "message": "Sensor alert deleted."
}
```

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with sensor_alert.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.

**Possible validation errors and codes:**

- code=2 - field=sensor - *sensor* field should be a correct UUID format
- code=4 - field=id - *id* field should be a correct UUID format

---

**Note:** Available for:

- User token

- Application token

### Update sensor alert configuration

**PATCH /api/v1/sensor/** (*sensor*) **/alerts/**
*id* **Request**:

```
PATCH /api/v1/sensor/8dbc5460-9a18-11e8-9eb6-529269fb1459/alerts/6cc44ad0-c635-
↪11e8-bf95-1dddf61566fa HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json

{
    "threshold_type": "min",
    "measurement_type": "HUM",
    "threshold_value": 65.25
}
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "success": true,
    "code": 100,
    "message": "Sensor alert updated"
}
```

**Example request with threhsold_type=loc**:

```
PATCH /api/v1/sensor/8dbc5460-9a18-11e8-9eb6-529269fb1459/alerts/6cc44ad0-c635-
↪11e8-bf95-1dddf61566fa HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json

{
    "threshold_type": "loc",
    "measurement_type": "LOC",
    "threshold_value": {
        "lat": 36.1812440939046,
        "lng": -101.84828589116029,
        "radius": 1000
    }
}
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

{
    "success": true,
    "code": 100,
    "message": "Sensor alert updated"
}
```

**Request JSON Object**

- **threshold_type** (*string*) – Threshold type.

  Possible values:

  - *min* - set minimum value and alert will be generated if value from sensor will be lower than this value
  - *max* - set maximum value and alert will be generated if value from sensor will be higher than this value
  - *loc* - threshold type for GPS coordinates. If location is out of radius from specified point - alert will be generated

- **measurement_type** (*string*) – Measurement type to check.
- **threshold_value** (*string*) – Threshold value to compare to.

  This parameter depends on the threshold_type.

  - for type *min* value should be a number.
  - for type *max* value should be a number.
  - for type *loc* value should be a JSON object with lat, lng and radius parameters. Example: {lat: 36.1812440939046, lng: -101.84828589116029, radius: 1000}

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with sensor_alert.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.
- 422 Unprocessable Entity – Validation error.

**Possible validation errors and codes:**

- code=2 - field=sensor - *sensor* field should be a correct UUID format
- code=4 - field=id - *id* field should be a correct UUID format
- code=5 - field=threshold_type - *threshold_type* is required
- code=6 - field=threshold_type - *threshold_type* must be *min*, *max* or *loc*
- code=7 - field=measurement_type - *measurement_type* is required
- code=8 - field=threshold_value - *threshold_value* is required
- code=9 - field=threshold_value - *threshold_value* for *threshold_type* “min” must be a correct float
- code=10 - field=threshold_value - *threshold_value* for *threshold_type* “max” must be a correct float
- code=11 - field=threshold_value - *threshold_value* for *threshold_type* “loc” must be a correct JSON object with *lat*, *lng* and *radius* parameters
- code=12 - field=threshold_value - *threshold_value.lat* must be correct latitude
- code=13 - field=threshold_value - *threshold_value.lng* must be correct longitude
- code=14 - field=threshold_value - *threshold_value.radius* must be correct positive integer

**Note:** Available for:

- User token
- Application token

### Get last generated alerts for sensor

**GET /api/v1/sensor/** (*sensor*) **/alerts/last**

**Request**:

```
GET /api/v1/sensor/7e9fb090-c717-11e8-b4d3-c587309ce935/alerts/last HTTP/1.1
Host: staging.sensorlab.io
Content-type: application/json
```

**Success response**:

```
HTTP/1.1 200 OK
Content-type: application/json

[
    {
        "measurement": {
            "value": [
                48.213800963395784,
                15.13119759639926
            ],
            "id": "e224df91-c718-11e8-9ef1-39ba3cb4dad3",
            "timestamp": 1538577032000,
            "sensor": "7e9fb090-c717-11e8-b4d3-c587309ce935",
            "type": "LOC"
        },
        "threshold": {
            "id": "e1d38b44-c718-11e8-9ef1-39ba3cb4dad3",
            "measurement_type": "LOC",
            "threshold_type": "loc",
            "threshold_value": {
                "lat": 48.213641,
                "lng": 15.130691999999954,
                "radius": 25
            }
        }
    },
    {
        "measurement": {
            "value": [
                85.16
            ],
            "id": "e224df90-c718-11e8-9ef1-39ba3cb4dad3",
            "timestamp": 1538577032000,
            "sensor": "7e9fb090-c717-11e8-b4d3-c587309ce935",
            "type": "HUM"
        },
        "threshold": {
            "id": "e1d38b43-c718-11e8-9ef1-39ba3cb4dad3",
```

(continues on next page)

(continued from previous page)


```
            "measurement_type": "HUM",
            "threshold_type": "max",
            "threshold_value": 60
        }
    },
    {
        "measurement": {
            "value": [
                20.25
            ],
            "id": "e224b883-c718-11e8-9ef1-39ba3cb4dad3",
            "timestamp": 1538577032000,
            "sensor": "7e9fb090-c717-11e8-b4d3-c587309ce935",
            "type": "TMP"
        },
        "threshold": {
            "id": "e1d38b42-c718-11e8-9ef1-39ba3cb4dad3",
            "measurement_type": "TMP",
            "threshold_type": "min",
            "threshold_value": 25
        }
    }
]
```


**Response JSON Object**

- **measurement** (*object*) – Triggering measurement.
- **measurement.value** (*array*) – Measurement value.
- **measurement.id** (*string*) – Measurement ID.
- **measurement.timestamp** (*number*) – Timestamp for measurement.
- **measurement.sensor** (*string*) – Sensor ID.
- **measurement.type** (*string*) – Measurement type.
- **threshold** (*object*) – Triggered threshold/sensor alert configuration.
- **threshold.id** (*string*) – Triggered sensor alert ID.
- **threshold.measurement_type** (*string*) – Sensor alert's measurement type.
- **threshold.threshold_type** (*string*) – Sensor alert's threshold type.
- **threshold.threshold_value** (*mixed*) – Sensor alert's threshold value.

**Request Headers**

- Authorization – Bearer token from authentication.
- Content-Type – application/json

**Status Codes**

- 200 OK – No errors, will return result with sensor_alert.
- 401 Unauthorized – User is not authorized - token is incorrect or outdated.

**Possible validation errors and codes:**

- code=2 - field=sensor - *sensor* field should be a correct UUID format

**Note:** Available for:

- User token
- Application token

CHAPTER 2

# Sensorlab.io Websockets

## 2.1 Basic Information

You can receive measurements and alerts in real-time using websockets.

Sensorlab.io websockets is based on Socket.IO.

Socket.IO is mainly used with Javascript, but you can find different implementation for other languages.

Here some links on libraries:

- Javascript
- Java
- C++

There are 3 main namespaces:

- /measurements - namespace to receive measurements
- /alerts - namespace to receive alerts
- /public - namespace to receiver measurements from public sensors

/measurements and /alerts requires JWT for authentication, /public requires public api key from application.

## 2.2 Websockets namespaces

### 2.2.1 Measurements namespace

You can receive measurements using websockets.

In order to connect to websockets you should generate correct JWT.

```
let api = new SensorlabApi();
api.auth.application_token(public_api_key, private_api_key)
         .then((application_token) => {
            console.log(application_token);
         })
         .catch((response) => {
            console.log(response);
         });
```

When you have token you can connect to /measurements namespace:

```
let io = require('socket.io-client');

let socket = io('https://ws.test.sensorlab.io:8080/measurements?token='+token, {
    path: '/ws'
});
```

If your token is invalid, you will receive an error. You can catch it like this:

```
socket.on('error', function (reason) {
    reject(JSON.parse(reason));
    console.error('Unable to connect Socket.IO', reason);
});
```

You will receive error like this:

```
{code: 401, message: 'Authentication error. Please check your token.'}
```

When connected, you can join sensor rooms. There are 2 types of rooms:

- measurements for sensor
- measurements by type for sensor

Connect sensor room:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.emit('sensor', { sensor: sensor});
```

Connect sensor/type room:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.emit('sensor', { sensor: sensor, type: 'TMP'});
```

To leave room use this code:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.emit('sensor/disconnect', { sensor: sensor});
```

And to leave sensor/type room:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.emit('sensor/disconnect', { sensor: sensor, type: 'TMP'});
```

You can also leave all rooms at once:

```
this.socket.emit('sensor/disconnect/all');
```

If user/application with this token doesn't have rights to the sensor, a "sensor/access_denied" message will be emitted.

You can catch this message like this:

```
socket.on('sensor/access_denied', function(params) {
    console.log(params.sensor, params.message);
});
```

If there are no problems server will start emitting measurements and you can receive them with your client:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.on('measurements/' + sensor, (measurements) => {
    measurements.forEach((measurement) => {
        console.log(measurement.timestamp);
        console.log(measurement.type);
        console.log(measurement.value);
    });
});
```

To receive measurements from sensor/type room use this code:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.on('measurements/' + sensor + '/TMP', (measurements) => {
    measurements.forEach((measurement) => {
        console.log(measurement.timestamp);
        console.log(measurement.type);
        console.log(measurement.value);
    });
});
```

### 2.2.2 Alerts namespace

You can receive alerts using websockets.

In order to connect to websockets you should generate correct JWT.

```
let api = new SensorlabApi();
api.auth.application_token(public_api_key, private_api_key)
         .then((application_token) => {
            console.log(application_token);
         })
         .catch((response) => {
            console.log(response);
         });
```

When you have token you can connect to /alerts namespace:

```
let io = require('socket.io-client');

let socket = io('https://ws.test.sensorlab.io:8080/alerts?token='+token, {
    path: '/ws'
});
```

If your token is invalid, you will receive an error. You can catch it like this:

```
socket.on('error', function (reason) {
    reject(JSON.parse(reason));
```

(continues on next page)

(continued from previous page)

```
    console.error('Unable to connect Socket.IO', reason);
});
```

You will receive error like this:

```
{code: 401, message: 'Authentication error. Please check your token.'}
```

When connected, you can join sensor rooms.

Connect sensor room:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.emit('sensor', { sensor: sensor });
```

To leave room use this code:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.emit('sensor/disconnect', { sensor: sensor});
```

You can also leave all rooms at once:

```
this.socket.emit('sensor/disconnect/all');
```

If user/application with this token doesn't have rights to the sensor, a "sensor/access_denied" message will be emitted.

You can catch this message like this:

```
socket.on('sensor/access_denied', function(params) {
    console.log(params.sensor, params.message);
});
```

If there are no problems server will start emitting alert and you can receive them with your client:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.on('alerts/' + sensor, (alerts) => {
    alerts.forEach((alert) => {
        console.log(alert.measurement.timestamp);
        console.log(alert.measurement.type);
        console.log(alert.measurement.value);

        console.log(alert.threshold.measurement_type);
        console.log(alert.threshold.threshold_type);
        console.log(alert.threshold.threshold_value);
    });
});
```

### 2.2.3 Public measurements namespace

You can receive measurements from public sensors using websockets.

In order to connect to websockets you should get public api key for your application.

When you have token you can connect to /measurements namespace:

```
let io = require('socket.io-client');

let socket = io('https://ws.test.sensorlab.io:8080/public?token='+token, {
    path: '/ws'
});
```

If your public api key is invalid, you will receive an error. You can catch it like this:

```
socket.on('error', function (reason) {
    reject(JSON.parse(reason));
    console.error('Unable to connect Socket.IO', reason);
});
```

You will receive error like this:

```
{code: 401, message: 'Authentication error. Please check your token.'}
```

When connected, you can join sensor rooms. There are 2 types of rooms:

- measurements for sensor
- measurements by type for sensor

Connect sensor room:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.emit('sensor', { sensor: sensor});
```

Connect sensor/type room:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.emit('sensor', { sensor: sensor, type: 'TMP'});
```

To leave room use this code:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.emit('sensor/disconnect', { sensor: sensor});
```

And to leave sensor/type room:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.emit('sensor/disconnect', { sensor: sensor, type: 'TMP'});
```

You can also leave all rooms at once:

```
this.socket.emit('sensor/disconnect/all');
```

If user/application with this token doesn't have rights to the sensor or sensor is not public, a "sensor/access_denied" message will be emitted.

You can catch this message like this:

```
socket.on('sensor/access_denied', function(params) {
    console.log(params.sensor, params.message);
});
```

If there are no problems server will start emitting measurements and you can receive them with your client:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.on('measurements/' + sensor, (measurements) => {
    measurements.forEach((measurement) => {
        console.log(measurement.timestamp);
        console.log(measurement.type);
        console.log(measurement.value);
    });
});
```

To receive measurements from sensor/type room use this code:

```
let sensor = '7e9fb090-c717-11e8-b4d3-c587309ce935';
socket.on('measurements/' + sensor + '/TMP', (measurements) => {
    measurements.forEach((measurement) => {
        console.log(measurement.timestamp);
        console.log(measurement.type);
        console.log(measurement.value);
    });
});
```

# HTTP Routing Table

## /api