API Documentation

Ludwig Johnson

Last edited 2020-03-23
Version 1.4
ID #A01

1 Introduction

The API is a REST API based on HTTP requests (although HTTPS is preferable where possible) and JSON responses. API keys can be generated from the web interface (including read/write permissions on a per key basis) and are needed for all requests. Times are stored and processed as UTC.

2 Endpoints

The API is on an HTTP endpoint with the request methods being GET and POST. The endpoint format is

`http://x.x.x.x/api/v1/`

All API calls below should be appended to the above.

3 Activity

An activity is what all physiological and related data will be tied to and stores when an activity was started and stopped. This will also be used to notify devices connected to a user when they should start/stop sending data. Only one activity per user is possible at one time.

3.1 POST /activity/control

API to start and stop an activity. When starting, the API responds with an activity ID above 0 to be used when sending data. If an activity is already active, the call returns the activity ID of that activity and allows for optional parameters to be added. When stopping, the API responds with an activity ID of 0. Optional parameters include type of activity, this can be used if a device supports selecting different types of activities. For example, coding, reviewing, research etc. The parameters **repo** and **commit** allows an IDE plugin to send a start call to the API with information about a repository and/or commit that is being worked on and at what time. Several repositories/commits can be associated with one activity, a timestamp (UTC) will be added for when the repository/commit was added to the database. If an error occurs, success will be set to false and any error codes stored in errors.
Permissions needed: write

Required parameters

```
device_id - Unique id for the device - generated by the user in web
            interface - linked to the user
USER-ID   - Unique id for the user - from web interface, linked to the
            API key
API-KEY   - API key generated in web interface - linked to the user
action    - To start or stop an active activity
```

Optional parameters

```
type      - Type of activity if device supports it
            Default: standard
repo      - Ability to link a git repository to an activity
            Default: n/a
commit    - Ability to link a git commit to an activity
            Default: n/a
```

Example

```
curl --location --request POST 'http://x.x.x.x/api/v1/activity/control' \
--header 'Content-Type: application/json' \
--header 'USER-ID: 1' \
--header 'API-KEY: 3cf54f9495f935ea18f5' \
--data-raw '{
    "device_id": "1",
    "action": "start",
    "type": "code_review",
    "repo": "torvalds/linux",
    "commit": "asd123"
}'
```

Response

```
{
    "success": true,
    "errors": [],
    "activity_id": 9345
}
```

3.2 GET /activity/status

API to check if an activity is running. In the response, activity ID is 0 if no activity is running. If there is an activity for the user, the API responds with a value above 0 which is the activity ID needed to send physiological data. If an error occurs, success will be set to false and any error codes stored in errors.
Permissions needed: read

Required parameters

```
USER-ID   - Unique id for the user - from web interface, linked to the
            API key
API-KEY   - API key generated in web interface - linked to the user
```

Example

```
curl --location --request GET 'http://x.x.x.x/api/v1/activity/status' \
--header 'Content-Type: application/json' \
--header 'USER-ID: 1' \
--header 'API-KEY: 3cf54f9495f935ea18f5' \
--data-raw ''
```

Response

```
{
    "success": true,
    "errors": [],
    "activity_id": 6542
}
```

4 Data

API for sending and receiving physiological and related data.

4.1 POST /data/in

API to send physiological and related data to database behind endpoint. This API call relies on an activity being active for the user already, otherwise an error will be returned. The data can be of different types such as heart rate, sweat percentage, compilation errors, etc. With every response, the continue variable is set to either true or false. This is will be set to false if another device has used the activity/control API to stop the activity. This means that continuous polling to see if an activity is active is not required when data transmission has started. Once this is set to false, the device should stop sending data. If an error occurs, success will be set to false and any error codes stored in errors.
Permissions needed: write

Required parameters

```
DEVICE-ID   - Unique id for the device - generated by the user in web
              interface - linked to the user
USER-ID     - Unique id for the user - from web interface, linked to
              the API key
API-KEY     - API key generated in web interface - linked to the user
ACTIVITY-ID - What activity id the data is associated to
TYPE        - What type of data is being sent
data        - JSON formatted data
```

Example

```
curl --location --request POST 'http://x.x.x.x/api/v1/data/in' \
--header 'Content-Type: application/json' \
--header 'USER-ID: 1' \
--header 'API-KEY: 3cf54f9495f935ea18f5' \
--header 'DEVICE-ID: 1' \
--header 'ACTIVITY-ID: 3' \
--header 'TYPE: heartbeat' \
--data-raw '{"bpm": 123, "message": "321"}'
```

Response

```
{
    "success": true,
    "errors": [],
    "continue": true
}
```

4.2 GET /data/out/users

API to get list of users.
Permissions needed: read

Required parameters

```
API-KEY     - Global API key generated in web interface by admin
USER-ID     - User-id associated with the global API key
```

Example

```
curl --location --request GET 'http://x.x.x.x/api/v1/data/out/users' \
--header 'API-KEY: 3cf54f9495f935ea18f5' \
--header 'USER-ID: 1'
```

Response

```
{
    "success": true,
    "errors": [],
    "users": [
        {
            "user_id": "1",
            "name": "Ludwig Johnson",
            "privilege": "admin"
        },
        {
            "user_id": "2",
            "name": "Alan Turing",
            "privilege": "user"
        }
    ]
}
```

4.3 GET /data/out/activity

API to get activities tied to a user, device and/or a range of other variables to filter the response.
Time format: yyyy-mm-ddTHH:MM:SS[.fff]
Ex: 2020-02-14T13:04:23 or 2020-02-14T13:04:23.231
Permissions needed: read

Required parameters

```
API-KEY - Global API key generated in web interface by admin
USER-ID     - User-id associated with the global API key
```

Optional parameters

```
id            - Filter response by a specific activity
user_id       - Filter response by a specific user
device_id     - Filter response by device starting activity
activity_type - Filter response by type of activity
repo          - Filter response by a git repository
commit        - Filter response by a git commit
active        - Filter response by if activity is active now
time_start    - Filter response with data only after timestamp
time_end      - Filter response with data only before timestamp
```

Example

```
curl --location --request GET 'http://x.x.x.x/api/v1/data/out/activity?id=1&
user_id=1&device_id=1&activity_type=code_review&repo=torvalds/linux&
commit=f35111a946548e3b34a55abbad3e9bacce6cb10f&active=true&
time_start=2020-02-20T15:00:00.000&time_end=2020-02-24T13:23:45' \
--header 'API-KEY: 3cf54f9495f935ea18f5' \
--header 'USER-ID: 1'
```

Response


```
{
  "success": true,
  "errors": [],
  "activities": [
    {
      "activity_id": 30,
      "user_id": 3,
      "device_id": 5,
      "type": "write code",
      "active": false,
      "time_start": "20201230T215030Z",
      "time_end": "20201230T224530Z"
    },
    {
      "activity_id": 31,
      "user_id": 3,
      "device_id": 5,
      "type": "code review",
      "active": true,
      "time_start": "20201230T225030Z",
```

```
      "time_end": "null"
    }
  ]
}
```

4.4 GET /data/out/device

API to get devices tied to a user, activity and/or device type to filter the response.
Permissions needed: read

Required parameters

```
API-KEY     - Global API key generated in web interface by admin
USER-ID     - User-id associated with the global API key
```

Optional parameters

```
id          - Filter response by a specific device
user_id     - Filter response by a specific user
name        - Filter response by a device name
device_type - Filter response by type of device
```

Example

```
curl --location --request GET 'http://x.x.x.x/api/v1/data/out/device?id=1&
name=My_watch&device_type=Garmin%206X&user_id=1' \
--header 'API-KEY: 3cf54f9495f935ea18f5' \
--header 'USER-ID: 1'
```

Response


```
{
  "success": true,
  "errors": [],
  "devices": [
    {
      "device_id": 231,
      "name": "My watch",
      "type": "Apple watch",
      "user_id": 2,
      "time_added": "2020-03-23T14:03:23.122Z"
    },
    {
      "device_id": "534",
      "name": "My new watch",
      "type": "Garmin watch",
      "user_id": 2,
      "time_added": "2020-03-23T14:03:23.122Z"
    }
  ]
}
```

4.5 GET /data/out

API to get data out.
Time format: yyyy-mm-ddTHH:MM:SS[.fff]
Ex: 2020-02-14T13:04:23 or 2020-02-14T13:04:23.231
Permissions needed: read

Required parameters

```
API-KEY     - Global API key generated in web interface by admin
USER-ID     - User-id associated with the global API key
```

Optional parameters

```
id            - Filter response by a specific data object
user_id       - Filter response by a specific user
device_id     - Filter response by a specific device
activity_id   - Filter response by a specific activity
type_of_data  - Filter response by type of data
activity_type - Filter response by type of activity
active        - Filter response by currently active/inactive activities
repo          - Filter response by a certain repository
commit        - Filter response by a certain commit
device_name   - Filter response by a device name
device_type   - Filter response by type of device
before_time   - Get data at or before this timestamp
after_time    - Get data at or after this timestamp
```

Example

```
curl --location --request GET 'http://x.x.x.x/api/v1/data/out?id=1&
user_id=1&device_id=1&activity_id=8&type_of_data=heartbeat&
activity_type=code_review&active=false&repo=torvalds/linux&
commit=f35111a946548e3b34a55abbad3e9bacce6cb10f&
device_name=My_watch&device_type=Garmin%206X&
before_time=2020-03-10T20:00:00&after_time=2020-03-05T20:32:12' \
--header 'API-KEY: 3cf54f9495f935ea18f5' \
--header 'USER-ID: 1'
```

Response

```
{
  "success": true,
  "errors": [],
  "data_objects": [
    {
      "data_id": 1,
      "device_id": 1,
      "activity_id": 1,
      "type": "heartbeat",
      "data": [
        {
          "bpm": 123,
          "message": "abc123"
        }
      ],
```

```
      "user_id": 1,
      "time_added": "2020-03-23T14:04:15.812Z"
    },
    {
      "data_id": 2,
      "device_id": 1,
      "activity_id": 1,
      "type": "heartbeat",
      "data": [
        {
          "bpm": 127,
          "message": "db"
        }
      ],
      "user_id": 1,
      "time_added": "2020-03-23T14:04:25.921Z"
    }
  ]
}
```

5 Error codes

100 = Invalid API key or user id
101 = Invalid API permissions
102 = Invalid API key (Global keys)
200 = No activity available
201 = Wrong activity ID
300 = General server error
301 = Invalid parameter used
302 = Malformed request
303 = Device does not belong to user
304 = Device ID does not exist
400 = JSON not valid