Safectory Track – API Quickstart Guide

safectory GmbH

1 General Remarks 2

2 Login: HTTP POST Request 2

3 Approach 1: WebSocket 2
- 3.1 WebSocket Endpoint . . . 2
- 3.2 Connection Requirements . . . 2
- 3.3 Ping/Pong Mechanism . . . 4
- 3.4 Web Session Timeout . . . 5
- 3.5 Analyze WebSocket Traffic . . . 5
- 3.6 Example: Retrieve Beacon RSSI (`rssi`) from WebSocket . . . 5

4 Approach 2: RESTful Service 8
- 4.1 API Endpoints in General . . . 8
- 4.2 Retrieving Position Information . . . 8
- 4.3 Position Filters . . . 9
  - 4.3.1 PositionId Filter . . . 9
  - 4.3.2 DeviceId Filter . . . 10
- 4.4 Devices . . . 10
- 4.5 Beacons . . . 11
- 4.6 Reports . . . 12
- 4.7 Updating Entities . . . 13

1 General Remarks

To access data from the Track backend (https://track.safectory.com), you must have a Track account. If you do not have an account, please contact us at support@safectory.com.

The JSON examples in this document are illustrative and may not include all possible fields. Future Track releases may introduce new fields or deprecate existing ones, but we strive to maintain backward compatibility whenever possible.

2 Login: HTTP POST Request

URL: `https://track.safectory.com/api/session`

Example:

```
curl -s -c ./track_cookies https://track.safectory.com/api/session --data
↪  'email=account@yourdomain.de&password=9ejKJJAYKw3VCWev'
```

Content of the cookie without comments:

```
track.safectory.com     FALSE /api    FALSE   0       JSESSIONID 1234567890abcdefghij
```

3 Approach 1: WebSocket

The WebSocket API provides real-time updates in JSON format.

- **Initial Data Push:** Upon opening a connection, all known devices, beacons, and their latest positions will be sent once.
- **Real-Time Updates:** After the initial push, only changes will be sent asynchronously.

3.1 WebSocket Endpoint

`wss://track.safectory.com/api/socket`

3.2 Connection Requirements

To establish a WebSocket connection, the client must include a valid `Sec-WebSocket-Key` header. This key should be a randomly generated 16-byte (base64-encoded) value, as specified in RFC 6455.

You can generate a valid key using the following command:

```
head -c16 /dev/urandom | base64
```

Example `Sec-WebSocket-Key`:

```
Glqex2Dld1EaSqg/QK5BJ+9zdt9WbI5I6XndnZL24+4=
```

Example cURL call:

```
#!/bin/bash

HOST="track.safectory.com"
URL="https://$HOST";
USER="<User>";
PASS="<Password>";

rm -f ./track_cookie

curl -s -c ./track_cookie "$URL"/api/session --data 'email='"$USER"'&password='"$PASS"
```

```
curl -i -b ./track_cookie --no-buffer --header 'Connection: Upgrade' --header 'Upgrade: websocket'
 ↪  --header 'Host: '"$HOST" --header 'Origin: '"$URL" --header 'Sec-WebSocket-Version: 13'
 ↪  --header 'Sec-WebSocket-Key: '$(head -c32 /dev/urandom |base64) "$URL"/api/socket --output -
```

Output (excerpt):

```
HTTP/1.1 101 Switching Protocols
Server: nginx
Date: Wed, 26 Feb 2020 16:05:25 GMT
Connection: upgrade
Sec-WebSocket-Accept: qGEgH3En71di5rrssAZTmtRTyFk=
Upgrade: WebSocket

~
{
    "positions": [{
        "id": 67532839,
        "name": null,
        "attributes": {
            "floor": "2",
            "batteryLevel": 100.0,
            "appid": "com.safectory.trackclient",
            "distance": 0.0,
            "totalDistance": 0.0,
            "motion": false
    },
        "protocol": "osmand",
        "serverTime": "2020-01-14T10:18:44.514+0000",
        "deviceTime": "2020-01-14T10:18:44.000+0000",
        "fixTime": "2020-01-14T10:18:44.000+0000",
        "outdated": false,
        "valid": true,
        "latitude": 49.897595543798104,
        "longitude": 10.878755114972591,
        "altitude": 0.0,
        "speed": 0.0,
        "course": 0.0,
        "address": null,
        "accuracy": 0.0,
        "networkId": 0,
        "deviceId": 60031591,
        "beaconId": 3921,
        "beaconName": "Operation Management",
        "beaconMajor": 55,
        "beaconMinor": 123,
        "beaconUuid": "9a06de64642b4441b377b63c2d5cd123"
    }]
}~{
        "devices": [{
            "id": 44969282,
            "name": "LynKey",
            "attributes": {
            "uuid": "9a06de64642b4441b377b63c2d5cd828",
            "Weather": "Cloudy"
            },
            "uniqueId": "13614234773123",
            "status": "online",
            "lastUpdate": "2020-02-26T16:05:26.469+0000",
            "geofences": [],
```

```
            "activeGeofences": [],
            "notifications": [],
            "positionId": 75864142,
            "phone": "",
            "model": "",
            "contact": "",
            "category": null,
            "disabled": false,
            "allgeofencesenabled": false
        }]
}~{
    "positions": [{
        "id": 75864142,
        "name": null,
        "attributes": {
            "floor": "2",
            "batteryLevel": 100.0,
            "activationLevel": 0.0,
            "scanRssi": -55.0,
            "appid": "com.safectory.blegateway.push.12345_bz12345",
            "distance": 0.0,
            "totalDistance": 0.0,
            "motion": false
        },
        "protocol": "osmand",
        "serverTime": "2020-02-26T16:05:26.464+0000",
        "deviceTime": "2020-02-26T16:05:26.176+0000",
        "fixTime": "2020-02-26T16:05:26.176+0000",
        "outdated": false,
        "valid": true,
        "latitude": 49.897595543798104,
        "longitude": 10.878755114972591,
        "altitude": 0.0,
        "speed": 0.0,
        "course": 0.0,
        "address": null,
        "accuracy": 0.0,
        "networkId": 0,
        "deviceId": 44969282,
        "beaconId": 3921,
        "beaconName": "Operation Management",
        "beaconMajor": 55,
        "beaconMinor": 7,
        "beaconUuid": "9a06de64642b4441b377b63c2d5cd123"
    }]
}
```


3.3 Ping/Pong Mechanism

To maintain an active WebSocket session, the backend sends **ping** messages to the client every **30 seconds** (default).

- Each ping message contains a unique UUID.
- The client **must** respond with a corresponding **pong** message, using the same UUID.
- The UUID **changes** with every ping/pong exchange.

Example ping message (received from backend):

```
{
  "status": [
    {
      "message": "a0b34365-908f-4868-a52a-84a3ed83bd59",
      "type": "PING",
      "localeKey": null,
      "localeValues": null
    }
  ]
}
```

Expected pong response (sent by client):

```
pong a0b34365-908f-4868-a52a-84a3ed83bd59
```

3.4 Web Session Timeout

Initializing a WebSocket requires one HTTPS request to authenticate and authorize with the backend. Such HTTPS requests do have a (web) session timeout, which means that the session is automatically terminated if there was no user activity within a certain time frame (usually a whole day = 24 h = 86400 sec). When using the frontend via browser, expiration of that timeout is indicated by the following key–value pair (the value shows which web session timeout the respective Track instance is configured with):

```
{"webSessionTimeout":[86400]}
```

This timeout is only shown once on the WebSocket and has no further consequences on its connectivity. That is, as long as the ping requests are acknowledged with pong replies (see previous subsection 1.3 Ping/Pong Mechanism), the WebSocket is kept open.

3.5 Analyze WebSocket Traffic

You can analyze WebSocket communication directly in your browser using Developer Tools. The following steps have been confirmed to work in **Firefox** and **Chrome**:

1. **Open the Web Application:**
   - Navigate to the URL of your installation.
2. **Log In:**
   - Enter your **username** and **password** to access the system.
3. **Open Developer Tools (Firefox) / DevTools (Chrome):**
   - **Shortcut:** Press `F12`.
   - **Alternative:** Right-click anywhere inside the Track web application (TrackUI) and select **“Inspect”**.
4. **Monitor WebSocket Requests:**
   - Navigate to the **Network** tab.
   - Apply the **“WS”** filter to display WebSocket connections.
5. **Reload the Page:**
   - Use `CTRL + R` or `F5` to refresh the page.
6. **Inspect WebSocket Communication:**
   - Locate the WebSocket request (named `socket`).
   - Click on it and navigate to the **“Response”** (Firefox) / **Message** (Chrome) section to view real-time messages.

3.6 Example: Retrieve Beacon RSSI (`rssi`) from WebSocket

The WebSocket API pushes a **devices** object containing information about tracked devices. Additionally, a **positions** object is generated based on database table information. The `positions` object contains detailed device and beacon data, including the beacon `rssi`.

Overview to extract the beacon `rssi` value:

1. Get `devices` data from WebSocket.
2. Get `positions` data from WebSocket.
3. Match `positions.id` with `positionId` in the `devices` object.
4. Extract `rssi` from the matched position.

Step 1: Create `read_websocket.sh`

The following Bash script logs into the WebSocket API and captures the data stream.

Save the following script as `read_websocket.sh`:

```
#!/bin/bash
#
# Script for opening a WebSocket to the Track backend for continuously retrieving data
#
source ./userdata  # Load credentials (= variables EMAIL and PASSWORD)
DOM="track.safectory.com"
URL_PATH="https://track.safectory.com/api"
COOKIE="./track_cookie"

# Login and create a session cookie
if curl -k -s -c "$COOKIE" "$URL_PATH/session" --data "email=$EMAIL&password=$PASSWORD" | grep -q
 ↪  name; then
    echo "Logged In..."
else
    echo "LOGIN failed"
    exit 1
fi

WEBSOCK_KEY="$(head -c16 /dev/urandom | base64)"

curl -k -i -b "$COOKIE" --no-buffer \
     --header "Connection: Upgrade" \
     --header "Upgrade: websocket" \
     --header "Host: $DOM" \
     --header "Origin: $URL_PATH" \
     --header "Sec-WebSocket-Version: 13" \
     --header "Sec-WebSocket-Key: $WEBSOCK_KEY" \
     "$URL_PATH/socket" --output -
```

The content of the `userdata` file may simply look as follows:

```
EMAIL=api-user@safectory.com
PASSWORD=<REDACTED>
```

Step 2: Run the Script

Execute the script and log the output:

```
bash read_websocket.sh > /tmp/log_from_websocket.log
```

Step 3: Find Device

To find data for a specific device, search for its name (or **ID** if known) in the log file:

```
grep -a "quickstart-guide-device-1" /tmp/log_from_websocket.log
```

This will show JSON data similar to:

```
{
  "devices": [
        {
```

```
        "id": 794513675,
        "name": "quickstart-guide-device-1",
        "attributes": {
          "uuid": "d87b32bdcc0e437a81b3383833bc5ff8",
          "major": "4242",
          "minor": "1"
        },
        "uniqueId": "661168222238",
        "status": "unknown",
        "lastUpdate": "2025-02-21T10:05:04.906+00:00",
        "geofences": [],
        "activeGeofences": null,
        "notifications": [],
        "positionId": 794513912,
        "phone": "",
        "model": "",
        "contact": "",
        "category": "null",
        "disabled": false,
        "allgeofencesenabled": false,
        "lastSuccessfulAsnJob": null,
        "lastOneTimeJob": null,
        "lastIncompleteAsnJob": null,
        "firstDataTime": "2025-02-21T10:05:04.882+00:00",
        "lastAttempt": null,
        "lastAsnFail": null,
        "lastAsnCookie": null,
        "asnConfigs": [],
        "firmwareVersion": "4.6.1712.0-38546f5-sBISD-hMZ-gGUCH-lFEW-wEB-bOD-oES",
        "batteryLevel": 100,
        "leafGroupName": null
        }
    ]
}
```


→ For the next step, we are interested in: `"positionId": 794513912`

Step 4: Retrieve Position Data

Find the matching position using `positionId`:

```
grep -a 794513912 /tmp/log_from_websocket.log
```

This will show JSON data similar to:


```
{
    "positions": [
        {
        "id": 794513912,
        "name": null,
        "attributes": {
        "batteryLevel": 100,
        "scanRssi": -42
        },
        "protocol": "osmand",
        "serverTime": "2025-02-21T10:05:04.882+00:00",
        "deviceTime": "2025-02-21T10:05:04.882+00:00",
        "fixTime": "2025-02-21T10:02:20.172+00:00",
        "outdated": true,
        "valid": true,
```

```
        "latitude": 0,
        "longitude": 0,
        "eventId": null,
        "deviceId": 794513675,
        "scanRssi": -42,
        "rssi": -64,
        "beaconId": 794513671,
        "beaconName": "quickstart-guide-beacon-2",
        "beaconMajor": 4242,
        "beaconMinor": 2,
        "beaconUuid": "d87b32bdcc0e437a81b3383833bc5ff8",
        "sourceDeviceId": null,
        "deviceName": "quickstart-guide-device-1"
        }
    ]
}
```


→ The `rssi` **field** in the **positions** object represents the **beacon RSSI** value: `"rssi": 204`

4 Approach 2: RESTful Service

For a detailed endpoint description, please refer to https://track.safectory.com/api/swagger.

4.1 API Endpoints in General

Each endpoint supports standard CRUD operations:

- **GET** → Retrieve a list of objects
- **POST** (body: object) → Create a new object *(with object data expected in the request body)*
- **PUT** `/{id}` (body: object) → Update an existing object *(requires sending the full object state)*

4.2 Retrieving Position Information

Endpoint: `/api/positions`

URL: `https://track.safectory.com/api/positions`

Description:

Returns all position data accessible to the user, including beacon details (major/minor) and GPS coordinates.

Example request:

```
curl -s -b ./track_cookie "https://track.safectory.com/api/positions" | jq
```

Example response:


```
[
  {
    "id": 794513912,
    "name": null,
    "attributes": {
      "batteryLevel": 100,
      "scanRssi": -42
    },
    "protocol": "osmand",
    "serverTime": "2025-02-21T10:05:04.882+00:00",
    "deviceTime": "2025-02-21T10:05:04.882+00:00",
    "fixTime": "2025-02-21T10:02:20.172+00:00",
    "outdated": true,
```

```
    "valid": true,
    "latitude": 0,
    "longitude": 0,
    "eventId": null,
    "deviceId": 794513675,
    "scanRssi": -42,
    "rssi": -64,
    "beaconId": 794513671,
    "beaconName": "quickstart-guide-beacon-2",
    "beaconMajor": 4242,
    "beaconMinor": 2,
    "beaconUuid": "d87b32bdcc0e437a81b3383833bc5ff8",
    "sourceDeviceId": null,
    "deviceName": "quickstart-guide-device-1"
  }
]
```


4.3 Position Filters

4.3.1 PositionId Filter

URL: `https://track.safectory.com/api/positions?id=<positionId>`

Description:

Retrieve position details using a specific position ID.

Example request:

```
curl -s -b ./track_cookie 'https://track.safectory.com/api/positions?id=794513672' | jq
```

Example response:


```
[
  {
    "id": 794513672,
    "name": null,
    "attributes": {
      "appid": "beacon_change-by-api-quickstart-guide-user-api-quickstart-guide-group"
    },
    "protocol": null,
    "serverTime": null,
    "deviceTime": "2025-02-21T10:02:20.168+00:00",
    "fixTime": "2025-02-21T10:02:20.168+00:00",
    "outdated": false,
    "valid": true,
    "latitude": 0,
    "longitude": 0,
    "eventId": null,
    "deviceId": 794513670,
    "scanRssi": null,
    "rssi": null,
    "beaconId": 794513671,
    "beaconName": "quickstart-guide-beacon-2",
    "beaconMajor": 4242,
    "beaconMinor": 2,
    "beaconUuid": "d87b32bdcc0e437a81b3383833bc5ff8",
    "sourceDeviceId": null,
    "deviceName": "quickstart-guide-device-1"
  }
]
```

4.3.2 DeviceId Filter

URL: `https://track.safectory.com/api/positions?deviceId=<deviceId>`

Description:

Retrieve the latest position of a specific device by its device ID.

Example request:

```
curl -s -b ./track_cookie 'https://track.safectory.com/api/positions?deviceId=794513675'
```

Example response:


```
[
  {
    "id": 794513912,
    "name": null,
    "attributes": {
      "batteryLevel": 100,
      "scanRssi": -42
    },
    "protocol": "osmand",
    "serverTime": "2025-02-21T10:05:04.882+00:00",
    "deviceTime": "2025-02-21T10:05:04.882+00:00",
    "fixTime": "2025-02-21T10:02:20.172+00:00",
    "outdated": true,
    "valid": true,
    "latitude": 0,
    "longitude": 0,
    "eventId": null,
    "deviceId": 794513675,
    "scanRssi": -42,
    "rssi": -64,
    "beaconId": 794513671,
    "beaconName": "quickstart-guide-beacon-2",
    "beaconMajor": 4242,
    "beaconMinor": 2,
    "beaconUuid": "d87b32bdcc0e437a81b3383833bc5ff8",
    "sourceDeviceId": null,
    "deviceName": "quickstart-guide-device-1"
  }
]
```


4.4 Devices

URL: `https://track.safectory.com/api/devices`

Description:

Retrieve all devices a user has access to and filter by device name.

Example request:

```
curl -s -b ./track_cookie https://track.safectory.com/api/devices | jq -r '.. | objects |
↪  select(.name and (.name | contains("quickstart-guide-device-1"))) | select(.id)'
```

Example response:


```
{
  "id": 794513675,
```

```
  "name": "quickstart-guide-device-1",
  "attributes": {
    "uuid": "d87b32bdcc0e437a81b3383833bc5ff8",
    "major": "4242",
    "minor": "1"
  },
  "uniqueId": "661168222238",
  "status": "unknown",
  "lastUpdate": "2025-02-21T10:05:04.906+00:00",
  "geofences": [],
  "activeGeofences": [],
  "notifications": [],
  "positionId": 794513912,
  "phone": "",
  "model": "",
  "contact": "",
  "category": "null",
  "disabled": false,
  "allgeofencesenabled": false,
  "lastSuccessfulAsnJob": null,
  "lastOneTimeJob": null,
  "lastIncompleteAsnJob": null,
  "firstDataTime": "2025-02-21T10:05:04.882+00:00",
  "lastAttempt": null,
  "lastAsnFail": null,
  "lastAsnCookie": null,
  "asnConfigs": [],
  "firmwareVersion": "4.6.1712.0-38546f5-sBISD-hMZ-gGUCH-lFEW-wEB-bOD-oES",
  "batteryLevel": 100,
  "leafGroupName": "api-quickstart-guide-group"
}
```


4.5 Beacons

URL: `https://track.safectory.com/api/beacons`

Description:

Retrieve all beacons a user has access to and filter by major and minor.

Example request

```
curl -sS -b ./track_cookie https://track.safectory.com/api/beacons | jq -r '.. | objects |
↪  select(.major == 4242 and .minor == 1)'
```

Example response:


```
{
  "id": 794513676,
  "name": "quickstart-guide-beacon-1",
  "attributes": {},
  "deviceId": 794513675,
  "uuid": "d87b32bdcc0e437a81b3383833bc5ff8",
  "major": 4242,
  "roleGroupId": 794314697,
  "minor": 1,
  "latitude": 0,
  "longitude": 0,
  "lastSeen": null,
  "batteryLevel": null,
```

```
  "relative": false
}
```

4.6 Reports

The following example script fetches report data for a specific device with name `quickstart-guide-device-1`:

```
#!/bin/bash
#
# Script to fetch a route for a device and format output via jq
#
source ./userdata # Load credentials (= variables EMAIL and PASSWORD)
URL="https://track.safectory.com/api"
COOKIE="./track_cookie"

# Log in and create session cookie
if curl -s -c "$COOKIE" "$URL/session" --data "email=$EMAIL&password=$PASSWORD" | grep -q name;
 ↪  then
  echo "Logged In..."
else
  echo "LOGIN failed"
  exit 1
fi

# Fetch device ID
DEVICEID=$(curl -sb "$COOKIE" "$URL/devices" | jq -r '.. | objects | select(.name and (.name |
 ↪  contains("quickstart-guide-device-1"))) | .id')

# Specify time span and filters
TO=2025-02-21T20%3A00%3A00.000Z
FROM=2025-02-21T10%3A00%3A00.000Z
FILTERS=filterSameLocation=true&filterFlapping=true

# Fetch route
curl -sb "$COOKIE" -H 'Accept: application/json'
 ↪  "$URL/reports/route?deviceId=$DEVICEID&from=$FROM&to=$TO&$FILTERS" | jq

rm "$COOKIE" # Cleanup
```

Example response:


```
[
  {
    "id": 794524664,
    "name": null,
    "attributes": {
      "batteryLevel": 100,
      "scanRssi": -42
    },
    "protocol": "osmand",
    "serverTime": "2025-02-21T12:04:24.093+00:00",
    "deviceTime": "2025-02-21T12:04:24.093+00:00",
    "fixTime": "2025-02-21T12:04:19.946+00:00",
    "outdated": true,
    "valid": true,
    "latitude": 0,
    "longitude": 0,
    "eventId": null,
    "deviceId": 794524647,
```

```
    "scanRssi": -42,
    "rssi": -64,
    "beaconId": 794524643,
    "beaconName": "quickstart-guide-beacon-2",
    "beaconMajor": 4242,
    "beaconMinor": 2,
    "beaconUuid": "d87b32bdcc0e437a81b3383833bc5ff8",
    "sourceDeviceId": null,
    "deviceName": "quickstart-guide-device-1"
  }
]
```


4.7 Updating Entities

Description:

In this example, we will update a beacon's latitude and longitude (values of all other fields remain unchanged).

URL: `https://track.safectory.com/api/beacons`

Get beacon ID:

```
curl -sS -b ./track_cookie "https://track.safectory.com/api/beacons" | jq -r '.. | objects |
 ↪  select(.major == 4242 and .minor == 2) | .id'
```

→ Result: `794524643`

Update beacon data:

```
curl -b ./track_cookie -X PUT \
  -H 'Accept: */*' -H 'Content-Type: application/json' \
  --data '{"id":794524643,"name":"quickstart-guide-beacon-2","attributes":{},
   ↪  "deviceId":794524642,"uuid":"d87b32bdcc0e437a81b3383833bc5ff8","major":4242,
   ↪  "roleGroupId":794314697,"minor":2,"lastSeen":"2025-02-21T12:04:24.093+00:00",
   ↪  "batteryLevel":100.0,"relative":false,"latitude":49.897644460986925,
   ↪  "longitude":10.878723934292793}' \
  'https://track.safectory.com/api/beacons/794524643'
```

Example response:


```
{
  "id": 794524643,
  "name": "quickstart-guide-beacon-2",
  "attributes": {},
  "deviceId": 794524642,
  "uuid": "d87b32bdcc0e437a81b3383833bc5ff8",
  "major": 4242,
  "roleGroupId": 794314697,
  "minor": 2,
  "latitude": 49.897644460986925,
  "longitude": 10.878723934292793,
  "lastSeen": "2025-02-21T12:04:24.093+00:00",
  "batteryLevel": 100,
  "relative": false
}
```