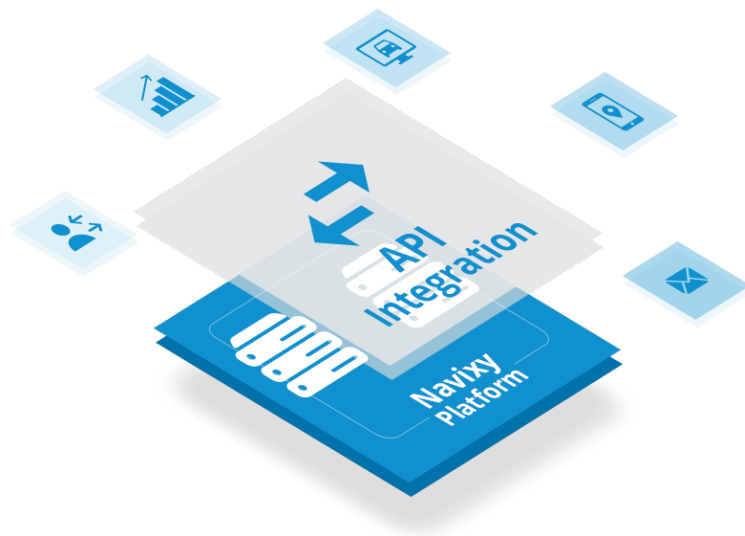


# Welcome to Navixy Developer Documentation



[Navixy](#) is a GPS tracking platform developed by [SquareGPS](#) company. Here you can find information about integration of 3rd party solutions with the Navixy platform, API and technical documentation for developers and partners.

All calls and methods will allow you to develop an application that pulls all the necessary information from the platform. By combining and processing the information you receive you will be able to cover the needs of your customers and partners by providing them with a customized app. More personalized solution can attract more customers and increase their loyalty.

Last update: November 1, 2021



# Getting started

## How to read this documentation

The documentation presented in several sections that are responsible for its own part of the Navixy platform:

- **General** - introductory section. Explains how to work with the documentation and how you can help us improve it. It also has information regarding translation of the platform into different languages.
- **Backend API** - describes all calls for working with information presented to users or sub-users in the UI. Tracking, reports, tasks, and more.
- **Panel-API** - describes all calls for working with information presented to administrators in the admin panel. Information about devices, tariff plans, users, and more.
- **Frontend** - provides information on customizing the welcome page and additional Weblocator and Delivery plugins.

You can switch between sections using menu on the top of the page. On the right side of the menu, you can find a button for downloading a PDF version of documentation and a link to our **github** page.

All files of the section are presented in menu on the left. Once you click on one of them - it will display file contents. On the right side of page you can find file's internal menu. Use it for quick navigation between parts of the file.

The documentation has three types of files: documents, guides and API calls.

Documents and guides are divided into semantic parts, the first of which is an introduction that briefly describes what the document is about.

API calls have the following structure:

- **Introduction** - API call description and general information about its purpose.
- **The structure of the object** - describes an object that is used in the calls. (optional)
- **API-actions** - the API base call and actions. All API-actions also divided into several points:
  - **Description of the API-actions** - describes purpose of the call.
  - **Requirements** - what rights are required to use the API-action. (optional)

- **Parameters table** - contains list of parameters for selected API call, their description and data type.
- **Examples** - example of a correct API call with all parameters listed. Examples can be useful for troubleshooting. You can also copy them and simply substitute the data with your own. Each example has a copy button in the upper right corner. If the parameters don't contain special characters, they are presented in two variants: POST and GET. If the parameters include special characters, only POST examples given.
- **Response** - an example of successful response from the server with description of every field.
- **Errors** - specific errors for this API-action. General error list applies to all calls.

## Limits

To maintain the stability of the system for all users, the platform has a limit of 50 requests/second per user and per IP address (if your app works with multiple users).

## Get involved

You can really help to improve [this documentation](#) or [localizations](#) of Navixy Platform.

If the translation of the user interface into your language is missing or contains errors, you can make or fix the localization on the [CrowdIn platform](#) yourself. Read [here](#) how to do it.

Current documentation may also contain errors or white spots. All of it is available in the public domain on [GitHub](#), and you can independently contribute in its correction or addition. Read [here](#) how to do it.

## Useful things

It is convenient to [use postman](#) for testing work with API.

Last update: February 20, 2023





# Get involved

If you notice an inaccuracy, mistake, typo or want to supplement the information in this documentation, then you can help us to improve it. All of this documentation is available in the public domain on [GitHub](#).

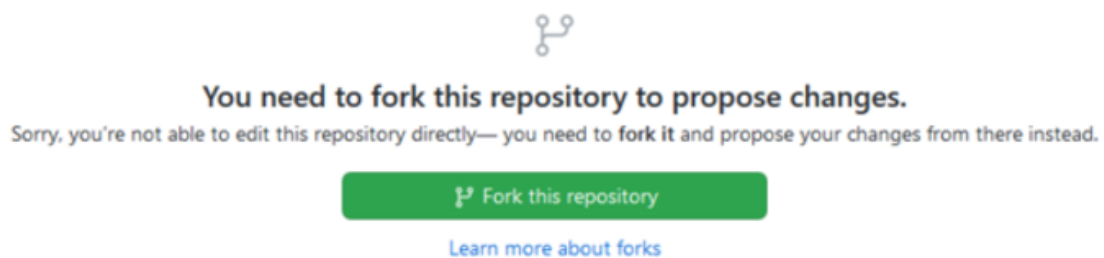
There are several ways:

1. [Creating an issue](#) with a detailed description of the problem.
2. [Editing a single page in a browser](#).
3. [Manually creating a fork](#) and doing multiple commits before creating a pull request.
4. [Installing and editing](#) documentation locally on your PC.


In each of these cases, a GitHub account is required. If you don't want to register on GitHub, you can just [contact us](#) with any convenient way.

## Easy way

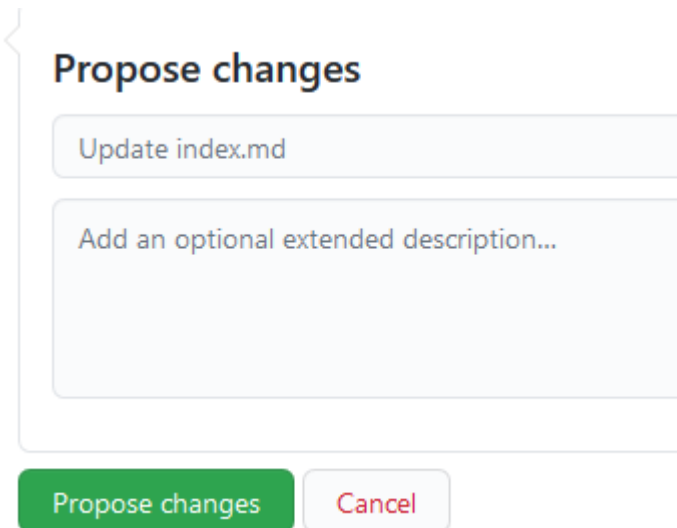
On each page in the upper right corner of the text top there is a link with a picture of a pencil :material-pencil:. After clicking on this link, you will be asked to create a fork of the repository (if you have not done this before).



Creating a fork done with one green button. After that, the edit form with page source code will open.

 **For correct edit of page, please read the [introduction into Mkdocs](#).**

After editing the page, you must fill out a description of what you have done.



**Propose changes**

Update index.md

Add an optional extended description...

Propose changes Cancel

Submitting a change will write it to a new branch in your fork, so you can send a pull request. We will review your pull request and accept it in the main branch.

Thus, this method is only suitable for simple edits on one page. There is [another way](#) to create pull requests to fix multiple pages at once.

## Second way

This method allows you to make several edits on different pages before proposing them in a pull request.

1. Create a fork [of the repository](#) if it has not been created yet. (Just click the "Fork" button in the upper right corner.)
2. Go to the created fork and find the file you are interested in.
3. Open the file and click the edit button.
4. Make edits and commit with a clear description of the changes.
5. Edit other files of interest to you in the same way.
6. Go to the start page of the fork and click on the "Pull request" button.

After review and pull request will be merged, and you can drop a fork.

## Hard way

This method involves installing the Git, IDE, Python and [Material for MkDocs](#) on your PC.

1. Install [Python 3](#).



2. Install [Git client](#).
3. Install an IDE, for example [IntelliJ IDEA](#) (Community edition would be enough).
4. Create a fork [of the repository](#) and cloning it to local project. In IDEA: `File -> New -> Project from version control`;
5. Install [mkdocs-material](#) and other dependencies. In console:

```
cd /path/to/project
mkdir venv
python -m venv ./venv
pip3 install -r requirements.txt
```

6. Start the documentation server locally. In console:

```
cd /path/to/project
source venv/bin/activate
# Windows: \venv\Scripts\activate.bat
mkdocs serve --dirtyreload
```

7. To check that the server has started, open in a browser: <http://localhost:8000>
8. Create a local git branch in project.
9. Make changes in documentation and test it in browser. Read the [introduction](#).
10. Commit and push changes. Please, use English in commit message.
11. Create a Pull Request (PR) on GitHub from your fork. Please, use English in PR description.
12. After the PR has been reviewed and merged to upstream you can remove branch and rebase a fork to the upstream.

## Introduction into Mkdocs

This documentation built on [mkdocs engine](#) and [mkdocs-material theme](#). Firstly, read [how to layout and write your Markdown source files](#) for an overview of how to write docs.

## Menu

The menu formed using the plugin [awesome-pages](#) automatically. To set the desired page order in the menu, use the file `.pages.yml` in directory. For example:

```
title: Backend API
nav:
```

```
- getting-started.md
- how-to
- resources
- websocket
```

`title` sets the name for menu section. `nav:` sets the sub-items order.

## Meta information

Each page must have meta information section at the beginning. Required fields: `title` and `description`. For example:

```
---
title: Get involved
description: Get involved into improving documentation and
translations of the Navixy Platform
---
```

Title will be displayed in menu and in browser title.

## Headers

The information on each page should be structured. On pages of the same type, the structure should be uniform.

## Example

API resource page structure:

```
# Resource name

Resource description.

## Object name

Object and its description

## API actions

Path: `/path/to/resource/`.

### method1

Method description.

#### Parameters
```

name	description	type	restrictions
param1	description.	int	`[1..100]`, not null
param2	description.	boolean	not null

#### Examples

```
=== "cURL"

```shell
curl -X POST 'https://api.navixy.com/v2/resource/sub_resource/
action' \
  -H 'Content-Type: application/json' \
  -d '{"param1": "value1", "param2": "value2", "hash":
"a6aa75587e5c59c32d347da438505fc3"}'
```

=== "HTTP GET"

...

https://api.navixy.com/v2/resource/sub_resource/action?
param1=value1&param2&hash=a6aa75587e5c59c32d347da438505fc3
...

#### Response

```json
{ "success": true }
```

#### Errors

Special error codes.

#### method2

...
```

#### Please note

If the response or structure has comments it is necessary to write these comments separately in the form of a list below.

For real example see [/user](#) and [source](#).

Last update: November 20, 2023



# Contacts

If you have questions, write to us in any way convenient for you.

You can call us or send email: [navixy.com/contact](mailto:navixy.com/contact).

Follow us in the social networks:

- [GitHub](#)
- [LinkedIn](#)
- [Twitter](#)
- [Facebook](#)
- [Instagram](#)
- [Youtube](#)

Last update: September 6, 2022



# Zapier

Automation is simply setting something up to run automatically. Automation is all around you, even if you don't realize it. Take your smartphone, for example. You receive alerts whenever you receive a text message, voice mail, or email.

The heart of any automation boils down to a simple command: WHEN and DO. "When this happens, do that." Even the most complex automation can be broken down into this simple command.

Zapier is a tool that helps you automate repetitive tasks between two or more apps—no code necessary. When an event happens in one app, Zapier can tell another app to perform (or do) a particular action.

A Zap is an automated workflow that tells your apps to follow this simple command: "When this happens, do that." Every Zap has a trigger and one or more actions. A trigger is an event that starts a Zap, and action is what your Zap does for you. When a Zap runs, each action it completes counts as one task.

Zapier is not for free but offers a free trial. Check their [pricing](#) before you start.

Example use cases you can achieve:

- Send a GPRS command / activate output on schedule;
- Send a Slack message on a tracker event;
- Deactivate tracker on an event.

## Create your first Zap

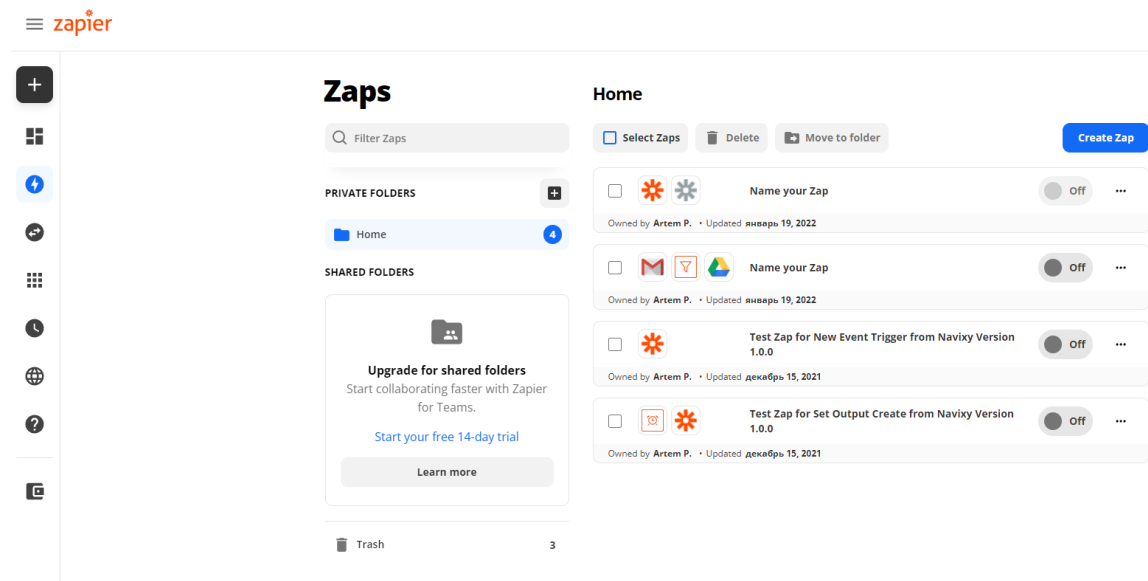
First, make sure you have [signed up for a Zapier account](#).

Before you create a Zap, it's helpful to think about what you're trying to accomplish. For example, let's say you have shared equipment that shouldn't work outside a particular geofence. Every time you receive a notification about the geofence leaving you can open an account and switch output responsible for its work. But it is much more appreciated being done automatically, isn't it?

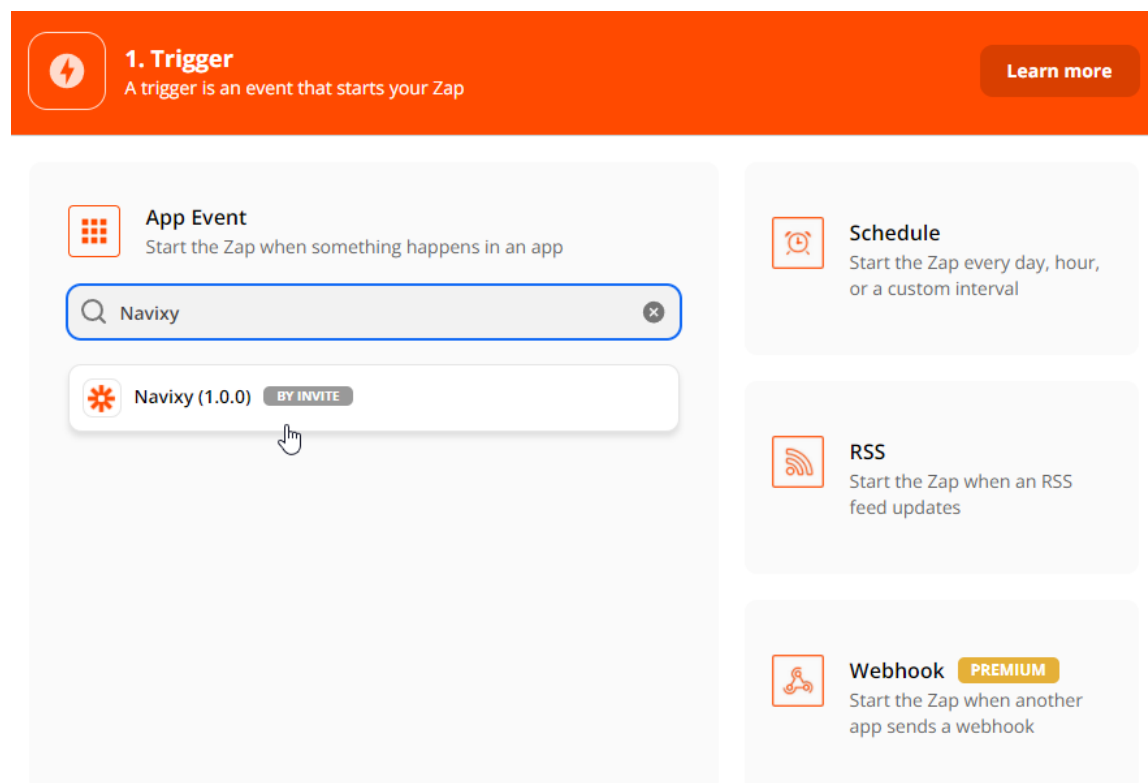
You can [create a rule in the UI](#) or make a new one [with APIs](#).

Open the [invitation link](#) to get access to the Navixy triggers and actions. Without this link actions will be not in search results. Click on the "Accept invite & Build a Zap" button to proceed.

Here you will see a dashboard with all your Zaps. You can create, update, switch on/off your Zaps.



Let's create a Zap. In a new window we should choose our app. Specify Navixy into search line and choose it from results.

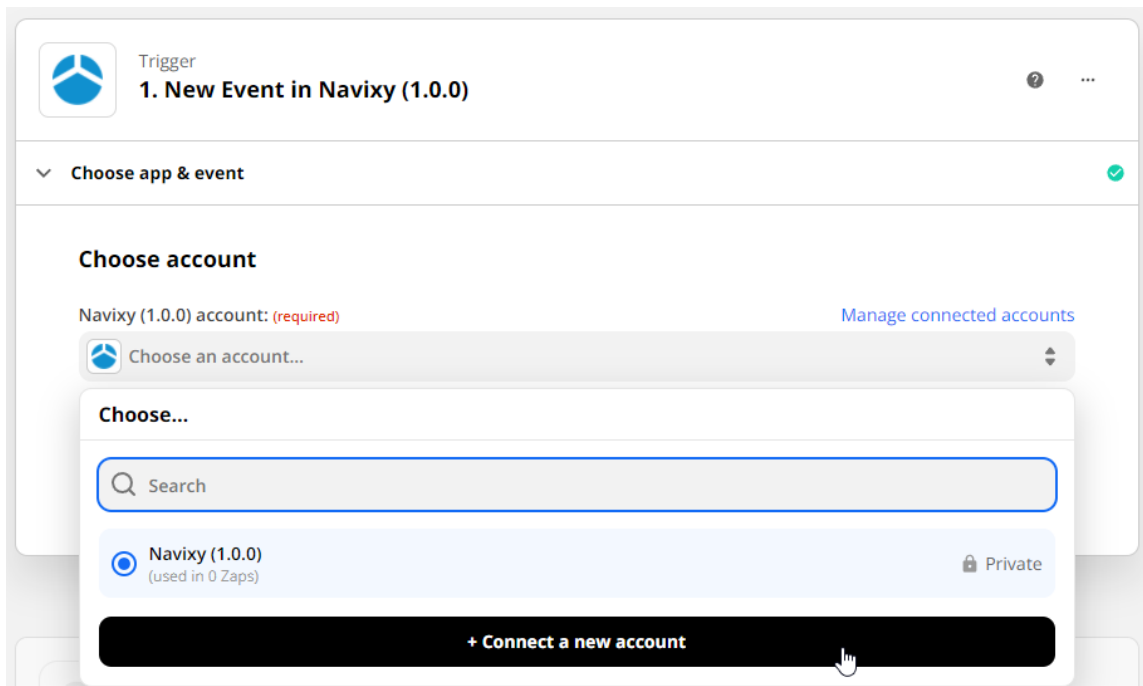


After it, you will see the Zap's body that contains triggers and actions. Start with a trigger and choose your first in a dropdown. We need to get all new geofence exit



events for all user's trackers on the platform and search for new of them constantly. So let's choose a New Tracker Event.

The program will request you connect an account.



Insert a User Session Key. Go to Navixy Admin Panel -> Users -> User -> click Get session key in the right menu. Read about limitations of [User Session Keys](#). Also, you can [create an API key](#) for one user and make as many Zaps as you want with it. At the same time API keys will not expire.

There you should choose the correct server where your account is located. If your user account ID starts with 1000xxxx - it is the US. Otherwise, choose the EU server.

## Set trigger

The next step is to configure a trigger. To create a simple Zap we recommend you use the New Tracker Event.

▼ Choose app & event



▼ Choose account



### Set up trigger

Tracker IDs 1 2 3

(required)

669673

Specify tracker IDs (not IMEI) to search for events, separate with comma. E.g. 123456, 654321. To find the id: Admin Panel > Trakers -> ID column or Platform -> Tracking -> Show device Info -> Copy id from the URL

Event types

(required)

outzone

Event type to monitor in quotes. E.g. "online", "offline". Take values from the list: inzone outzone sos battery\_off lowpower poweroff poweron sensor\_inrange sensor\_outrange input\_change output\_change task\_completed task\_delayed task\_failed task\_arrived task\_in\_progress checkpoint\_completed checkpoint\_delayed checkpoint\_failed checkpoint\_arrived checkpoint\_in\_progress route\_completed route\_faulty route\_failed task\_form\_submitted gps\_lost gps\_recover Contact support for the full list

Time zone 1 2 3

(required)

3

Enter the time zone of your Navixy account (e.g. -3)

Query Interval 1 2 3

(required)

15

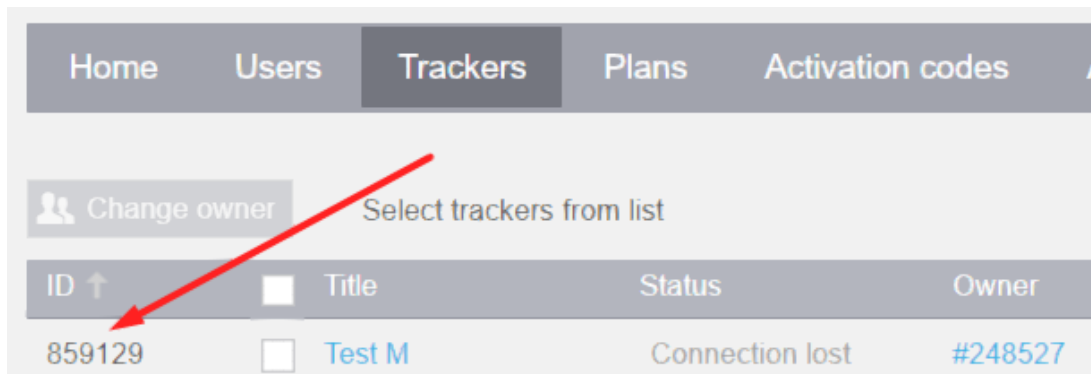
Interval in minutes. Look for events inside of this interval.


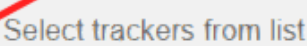
↻ Refresh fields

Continue

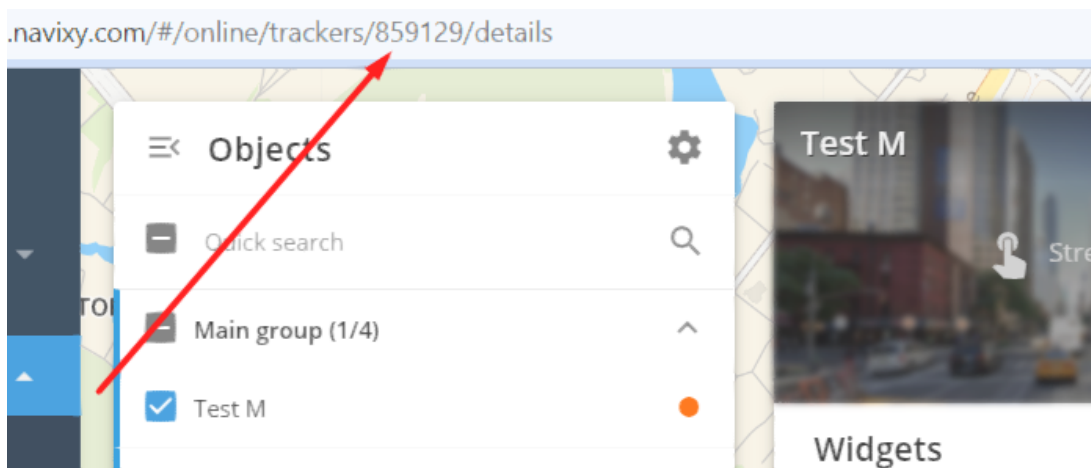
## Specify tracker IDs

You can find it in the admin panel. Admin Panel > Trakers -> ID column



| Home  | Users                    | Trackers | Plans           | Activation codes | A |
|---|--------------------------|----------|-----------------|------------------|---|
|   |                          |          |                 |                  |   |
| ID ↑  | <input type="checkbox"/> | Title    | Status          | Owner            |   |
| 859129  | <input type="checkbox"/> | Test M   | Connection lost | #248527          |   |

Or in the user interface. Platform -> Tracking -> Show device Info -> Copy ID from the URL



## Set event types

You can choose one type or several. Below is the list of all types:

- inzone
- outzone
- offline
- online
- sos
- battery\_off
- lowpower
- poweroff
- poweron
- sensor\_inrange
- sensor\_outrange
- input\_change
- output\_change
- task\_completed
- task\_delayed
- task\_failed
- task\_arrived
- task\_in\_progress
- checkpoint\_completed
- checkpoint\_delayed
- checkpoint\_failed
- checkpoint\_arrived
- checkpoint\_in\_progress
- route\_completed
- route\_faulty
- route\_failed
- task\_form\_submitted
- gps\_lost
- gps\_recover
- idle\_start
- idle\_end

- service\_task\_soon
- service\_task\_expired
- detach
- attach
- bracelet\_close
- bracelet\_open
- obd\_plug\_in
- obd\_unplug
- strap\_bolt\_cut
- strap\_bolt\_ins
- light\_sensor\_bright
- light\_sensor\_dark
- vibration\_start
- vibration\_end
- lock\_opened
- lock\_closed
- case\_opened
- case\_closed
- g\_sensor
- force\_location\_request
- alarmcontrol
- crash\_alarm
- door\_alarm
- hood\_alarm
- ignition
- parking
- security\_control
- gsm\_damp
- info
- odometer\_set
- tracker\_rename
- harsh\_driving
- auto\_geofence\_in
- auto\_geofence\_out

- inroute
- outroute
- speedup
- track\_end
- track\_start
- work\_status\_change
- call\_button\_pressed
- driver\_changed
- driver\_identified
- driver\_not\_identified
- driver\_absence
- driver\_enter
- driver\_distraction\_started
- driver\_distraction\_finished
- external\_device\_connected
- external\_device\_disconnected
- fueling
- drain
- forward\_collision\_warning
- headway\_warning
- lane\_departure
- peds\_in\_danger\_zone
- tsr\_warning
- peds\_collision\_warning
- checkin\_creation
- tacho
- antenna\_disconnect
- check\_engine\_light
- location\_response
- backup\_battery\_low
- fatigue\_driving
- fatigue\_driving\_finished
- proximity\_violation\_start
- proximity\_violation\_end

- no\_movement
- gps\_damp
- cruise\_control\_on
- cruise\_control\_off
- over\_speed\_reported
- distance\_breached
- distance\_restored
- excessive\_driving\_start
- excessive\_driving\_end
- excessive\_parking
- excessive\_parking\_finished
- state\_field\_control

#### Specify account's timezone

Account time zones should be specified like the next example:

- timezone is UTC+3 then specify 3;
- timezone is UTC-3 then specify -3.

#### Set a time interval for searching

Here the platform expects to get query intervals in minutes. Your Zap will look for events inside them.

For example, 15 means - check events for the last 15 minutes before this Zap requested info. It should be the same as the frequency of your Zap requests.

## Set action

Now it is time to create an action. When conditions are specified Zapier wants to know - what action should be used.

There are two output actions and one for sending the GPRS command.



### Special control for some models


There are different safety output change scenarios on some models that prevent output switching when there is a fast speed. For example, SECO on Teltonika devices. Other devices have specific commands that can switch output even on speed. You can contact the manufacturer to get the correct configurations, or you can use another action: "Send GPRS command" (if it is a device with a special command for output switch on speed).

We will describe here all actions to show how you can configure them. Choose only one that is convenient for your device and use case.




## Set output

Is an action for devices that can change one output at a time. For example, enable immobilizer on the first output.



Action

2. Navixy (1.0.0)

 Navixy (1.0.0)

Change

Action Event

(required)

Choose an event

CREATE

Set Output

Sets an output for device. Some trackers support Set Output command, other Set OutputS command.


Set Outputs

Sets outputs for device. Some trackers support Set Output command, other Set OutputS command.

Send GPRS Command

Send GPRS command to device

Here we should choose an account again. It is already in the dropdown. The next what we should do is to set up action settings.

 Action

2. Set Output in Navixy (1.0.0)



▼ Choose app & event



▼ Choose account

### Set up action


Tracker 1 2 3 (required)  
664348  
Specify tracker ID (not IMEI) of the device for output switching. To find the id: Admin Panel > Trakers -> ID column or Platform -> Tracking -> Show device Info -> Copy id from the URL

Output 1 2 3 (required)  
1  
Output number to change

Switch output to   (required)  

 Enable 

  
enable or disable output state

 Refresh fields

Continue

Zapier asks for the next information to proceed:


- Tracker ID - described in triggers;
- Output - output number to change;
- Switch output to - enable or disable output state.



## Set outputs


Is an action for devices that can change outputs with one general command. If we need to change one particular output state on such devices we should specify a new state for one and previous states for others.


The screenshot shows a configuration window for the '4. Set Outputs in Navixy (1.0.0)' action. At the top, there is a gear icon and the text 'Action 4. Set Outputs in Navixy (1.0.0)'. Below this, the 'Choose app & event' section contains a dropdown menu showing 'Navixy (1.0.0)' with a 'Change' button to its right. Underneath, the 'Action Event' field is labeled '(required)' and contains the text 'Set Outputs'. A dropdown menu is open below this field, listing three options: 'Set Output' (described as 'Sets an output for device. Some trackers support Set Output command, other Set OutputS command.'), 'Set Outputs' (described as 'Sets outputs for device. Some trackers support Set Output command, other Set OutputS command.'), and 'Send GPRS Command' (described as 'Send GPRS command to device'). The 'Set Outputs' option is currently selected and highlighted in blue.

Here we should choose an account again. It is already in the dropdown. The next what we should do is to set up action settings.

 Action

2. Set Outputs in Navixy (1.0.0)  

▼ Choose app & event 

▼ Choose account 

### Set up action

tracker ID 1 2 3 (required)


676714

Specify tracker ID (not IMEI) of the device for output switching. To find the id: Admin Panel > Trakers -> ID column or Platform -> Tracking -> Show device Info -> Copy id from the URL

output states (required)

true,false,false

Write needed output states. E.g. true,false,false

 Refresh fields

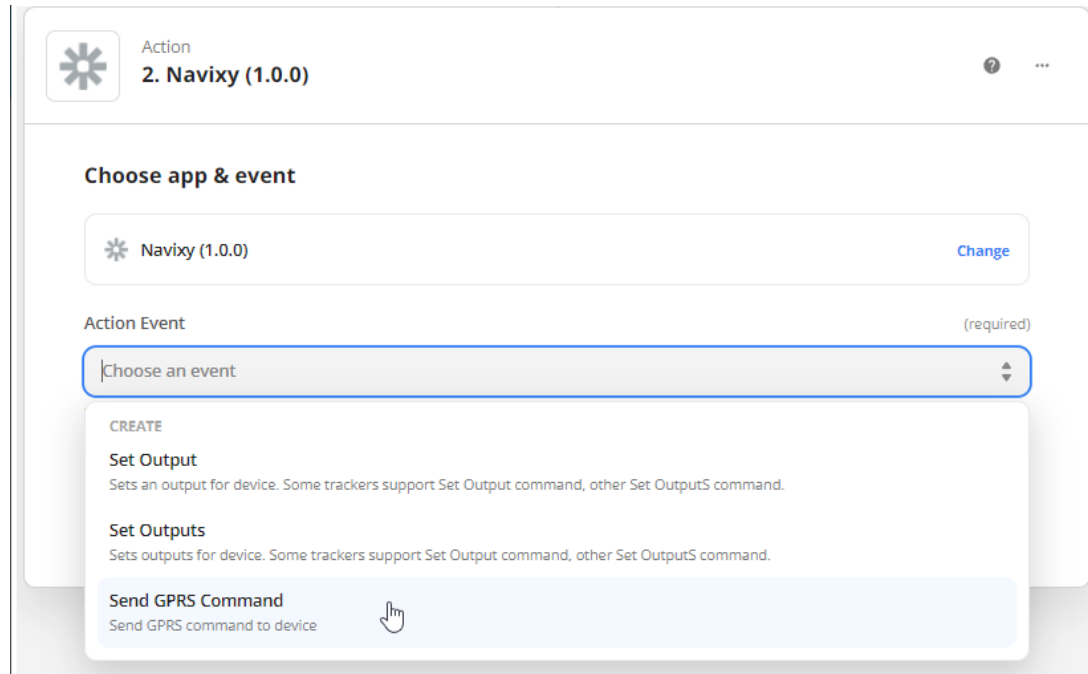
Continue

Zapier asks for the next information from us:

- Tracker ID - described in triggers;
- Output states - desired states of all digital outputs, e.g. true,true,false means output 1 is on, output 2 is on, output 3 is off.

## Send GPRS command

Is an action to send any command to a device. In our example, we use the command to switch digital output 1 on the Teltonika FMB920 device.



**2. Navixy (1.0.0)**

**Choose app & event**

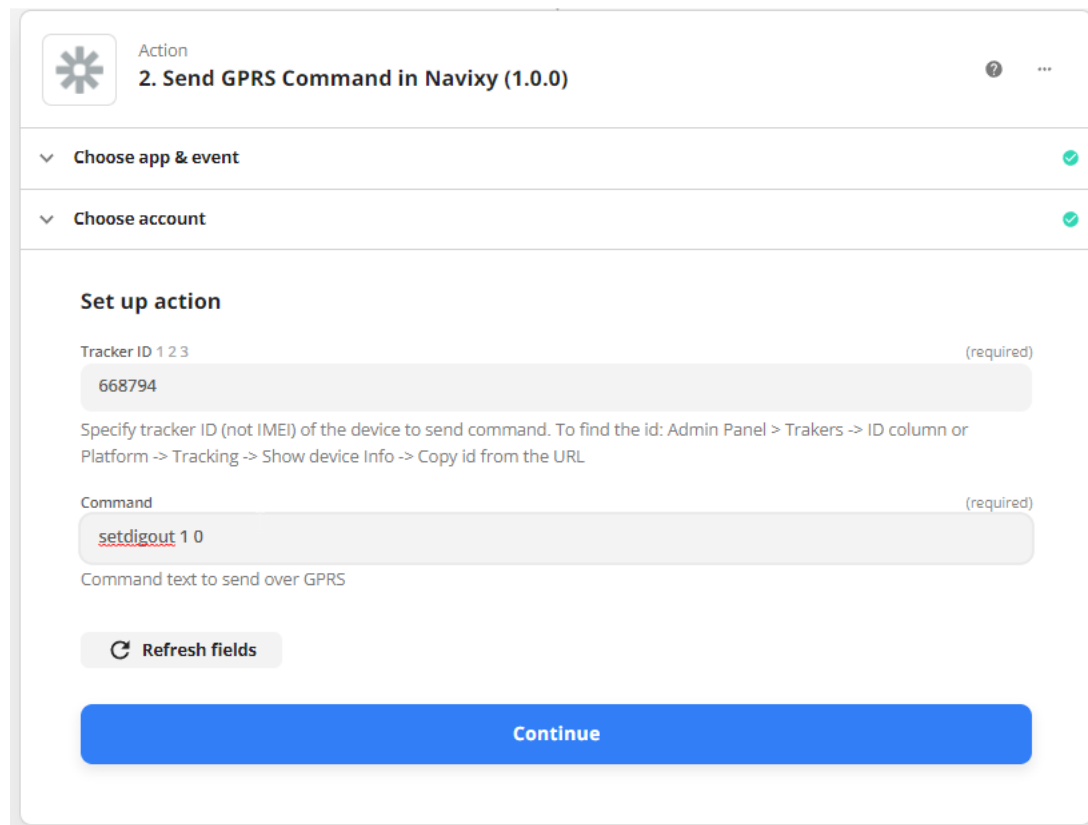
Navixy (1.0.0) [Change](#)

Action Event (required)

Choose an event

- CREATE
- Set Output**  
Sets an output for device. Some trackers support Set Output command, other Set OutputS command.
- Set Outputs**  
Sets outputs for device. Some trackers support Set Output command, other Set OutputS command.
- Send GPRS Command**  
Send GPRS command to device

Here we should choose an account again. It is already in the dropdown. The next what we should do is to set up action settings.



**2. Send GPRS Command in Navixy (1.0.0)**

✓ **Choose app & event**

✓ **Choose account**

**Set up action**

Tracker ID 1 2 3 (required)  
668794

Specify tracker ID (not IMEI) of the device to send command. To find the id: Admin Panel > Trakers -> ID column or Platform -> Tracking -> Show device Info -> Copy id from the URL

Command (required)  
setdigout 1 0

Command text to send over GPRS

[Refresh fields](#)

**Continue**

Zapier requests to specify the next info:

- Tracker ID - described in triggers;
- Command - a model's protocol-related command to send over GPRS.

## Results

The Zap is ready to work. Turn it on to start and try to out of the zone with your equipment. It will not work outside. Your device should be online to receive a command.

### **Output switching**

Switching off the engine on a moving vehicle is dangerous. Navixy carries no responsibility for such action.

Last update: June 28, 2023



# Languages

Navixy already supports many languages and provides an easy way to [add a new language](#):

```
Arabic
Croatian
Dutch
English
French
Georgian
German
Greek
Indonesian
Korean
Mongolian
Polish
Portuguese
Portuguese (Brazil)
Romanian
Russian
Sinhala
Spanish
Tamil
Thai
Turkish
Ukrainian
and others...
```

In all Navixy products both left-to-right and right-to-left languages are supported. The following Navixy projects are maintained currently:

- Desktop web interface
- Mobile web interface
- Java backend and API
- Tracker mobile app for iOS /Android
- Viewer mobile app for iOS / Android

[Become a contributor](#) and help us to translate Navixy products to a new language or improve the existing language packs.

Last update: November 1, 2021







# Translate Navixy

Localizing Navixy products to the language of your choice is simple and handy. Add a new language or update an existing translation in a few easy steps. Then you can translate Navixy to your language.

All translations are done through the [crowdin](#) online translation service, developed specifically for team-based translation projects.

## Getting started

First of all you should contact your manager, to obtain Crowdin account associated with Navixy Crowdin project.

There are two ways to localize navixy platform:

- Crowdin In-context translation
- Translate via Crowdin UI

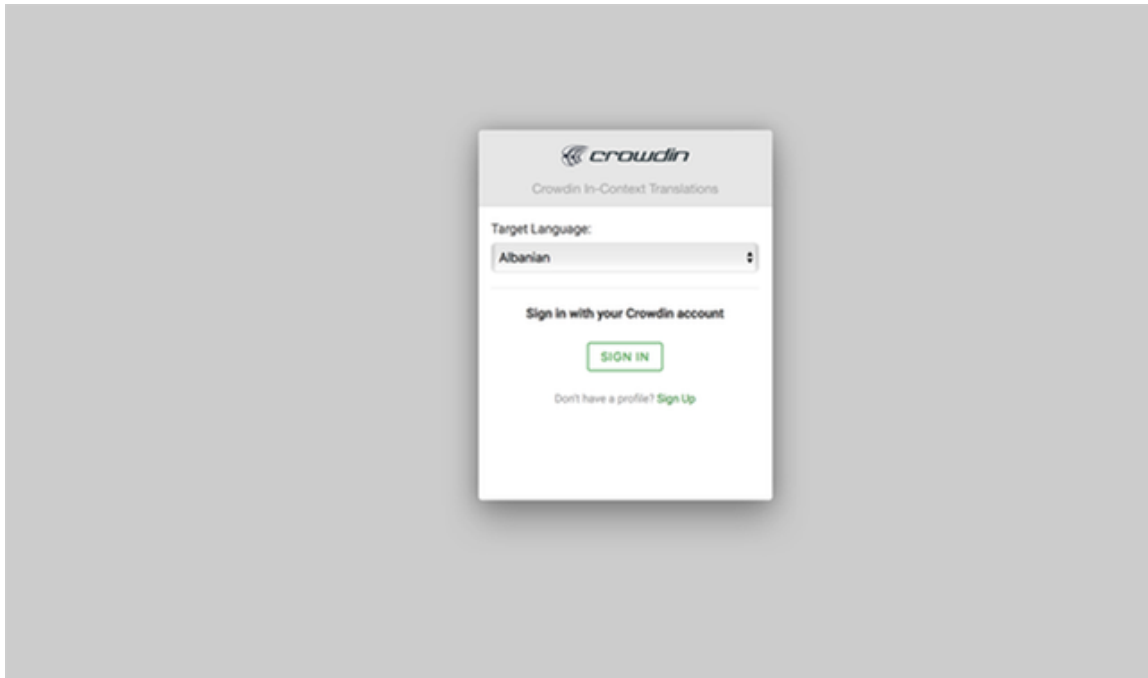
## Crowdin In-context translation (only Web UI)

Crowdin In-context translation is the most handy way to translate Navixy Web UI.

To launch Crowdin In-context service you should use special link:

```
https://demo.navixy.com/?locale=ach#/login
```

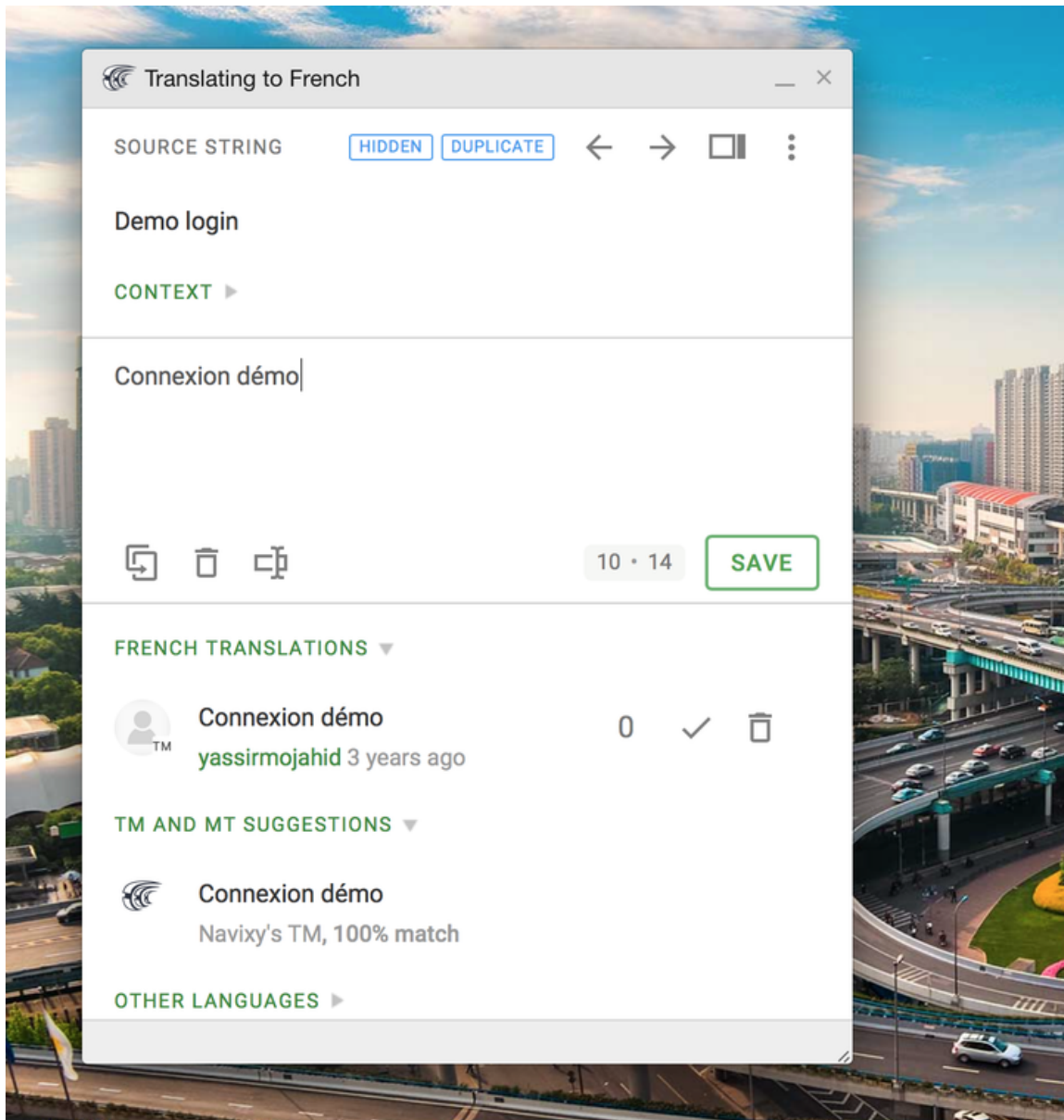
You should see crowdin authorization dialog.



After authorization standard Navixy UI will appear in a special translation mode. Click on a little icon near each text item



opens translation dialog



## Translate via Crowdin UI

Crowdin UI is a most powerful way to work with translations in Navixy, and the only way if you want to translate not Backend and Mobile apps.



## Translations delivery

Usually it takes about a week to deploy translations to production environment.

In case of standalone installations and mobile apps this time is linked to standalone/mobile app release schedule.

If you translate Navixy to the new language, after translation you should notify your manager that your translation is complete. Your manager will ask development team to add new language to the platform. In the other case translations of the existing language will be delivered to production automatically.

Last update: December 26, 2022







# Postman

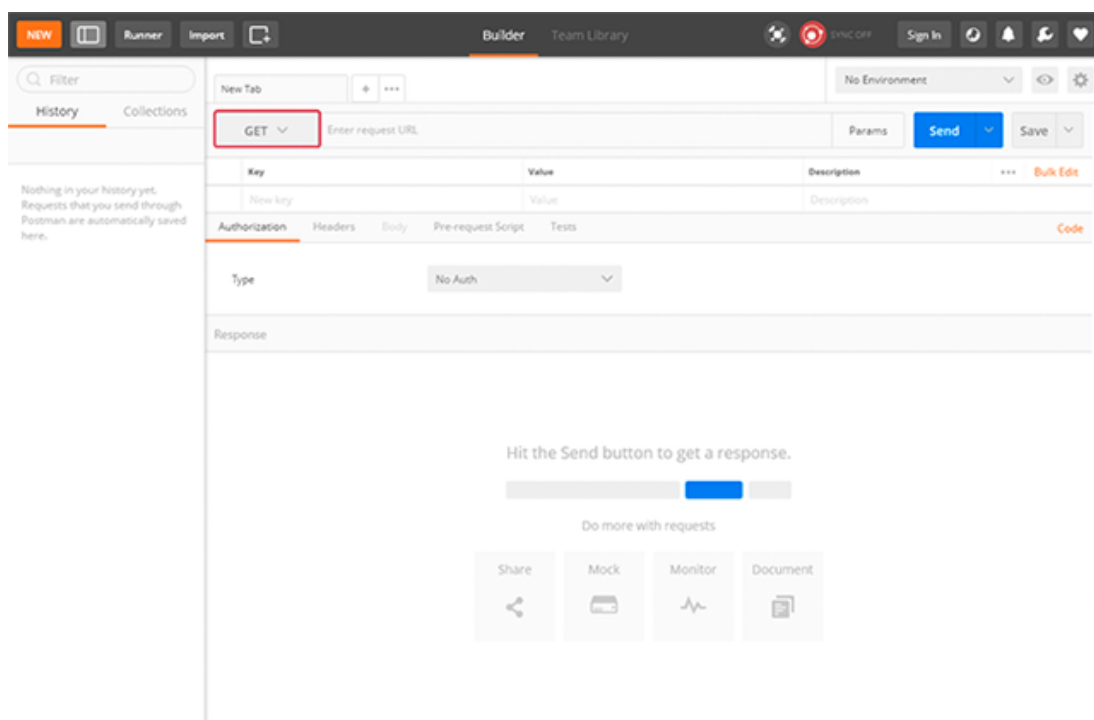
There are many tools that could be used to work with API requests. From simple input to browser's address line or cmd tool to more complex software. One of our personal favorites is Postman application. [Postman](#) is a collaboration platform for API development. It can be used for a variety of purposes ranging from simple request testing to creating and maintaining your own APIs for your own software.

For our purposes we will only review their API client.

## Your first request

Postman API client allows you to easily send various API requests and helps you fill out parameters without worrying that you will miss a quote or bracket. This can be especially handy when working with large requests.

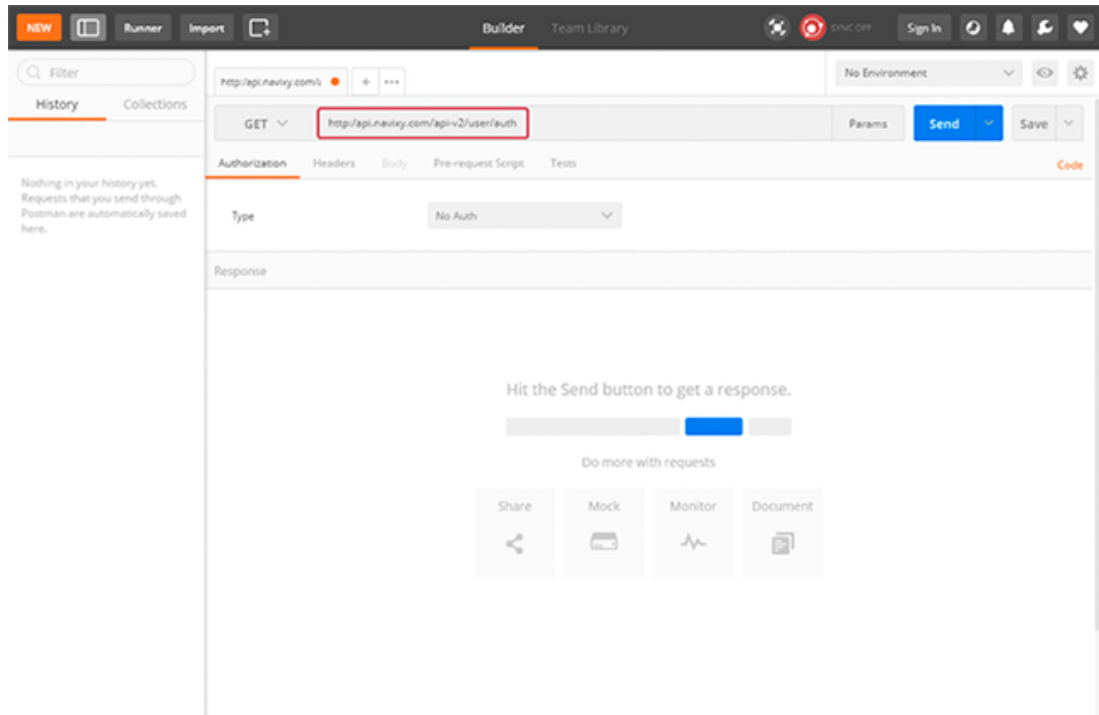
1. Select a request method:



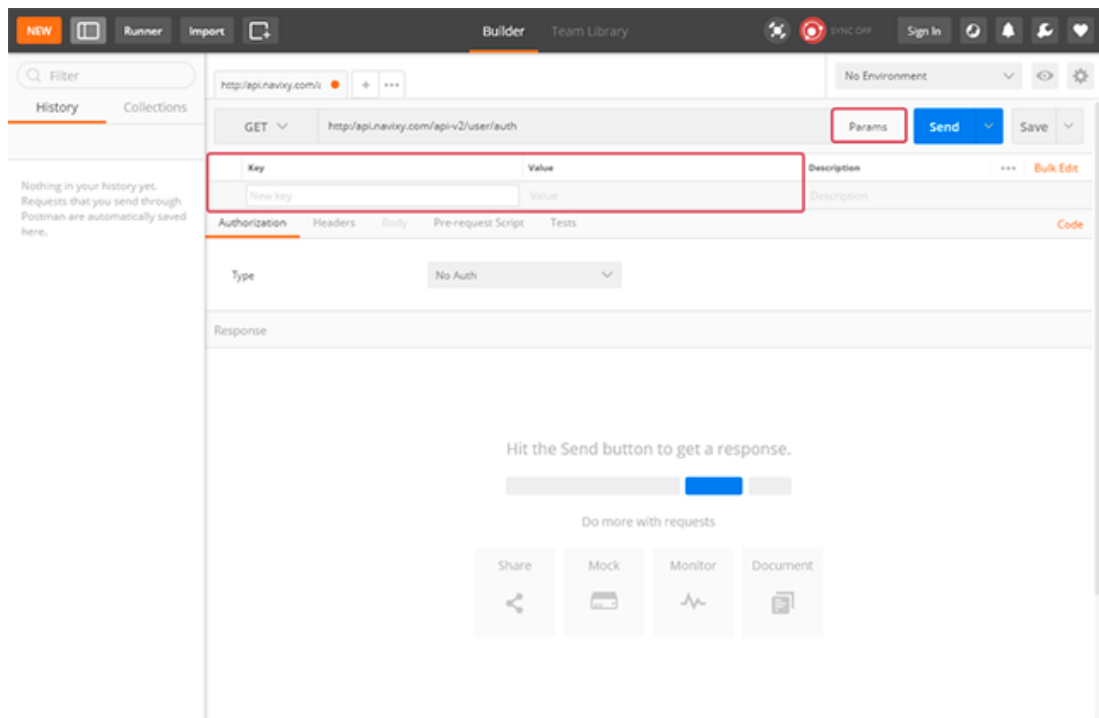
Each API request uses an HTTP method. The most common methods for Navixy API are GET and POST. GET methods retrieve data from an API. POST sends new data to an API.

2. Enter base request URL with the resource and sub-resource. In our example we will use [user/auth](#) and [tracker/list](#) requests. Base request URLs are:

- For EU server - <https://api.eu.navixy.com/v2/>
- For US server - <https://api.us.navixy.com/v2/>

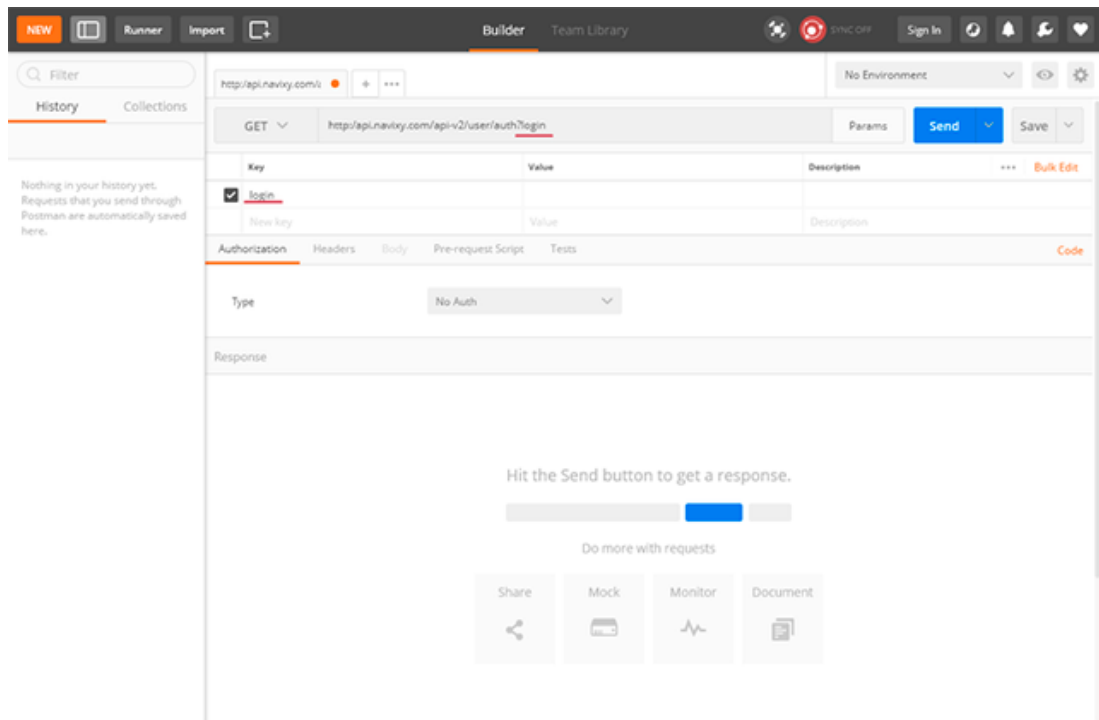


3. Click on the Params button, and you will see a table for key and value input:

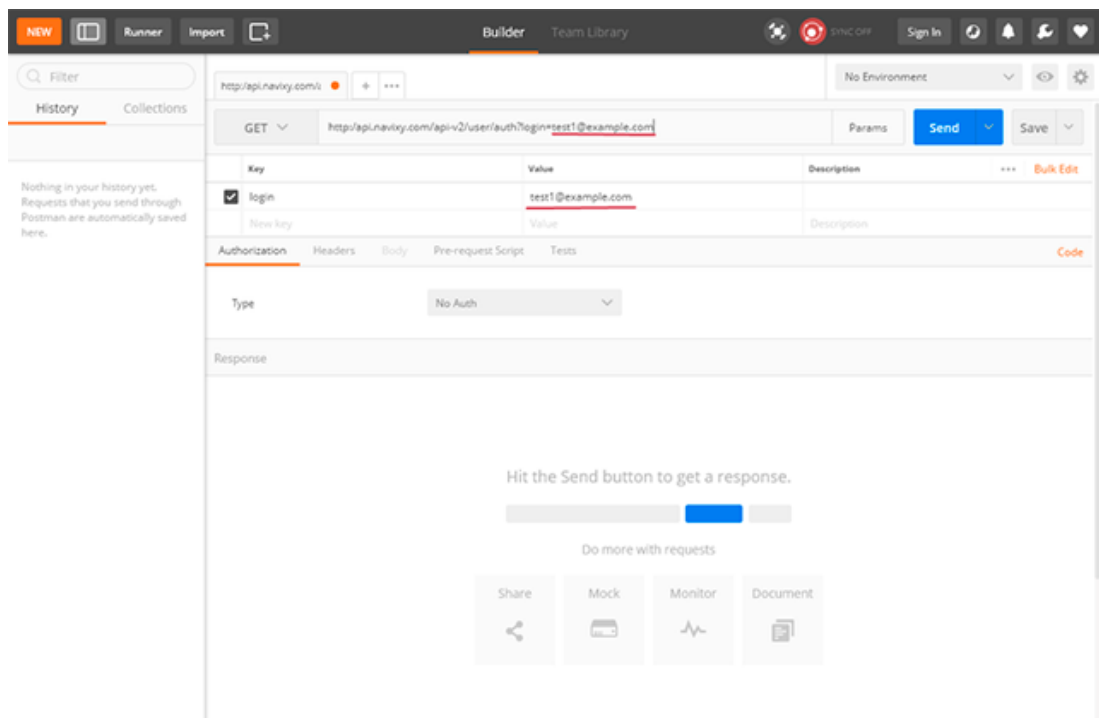


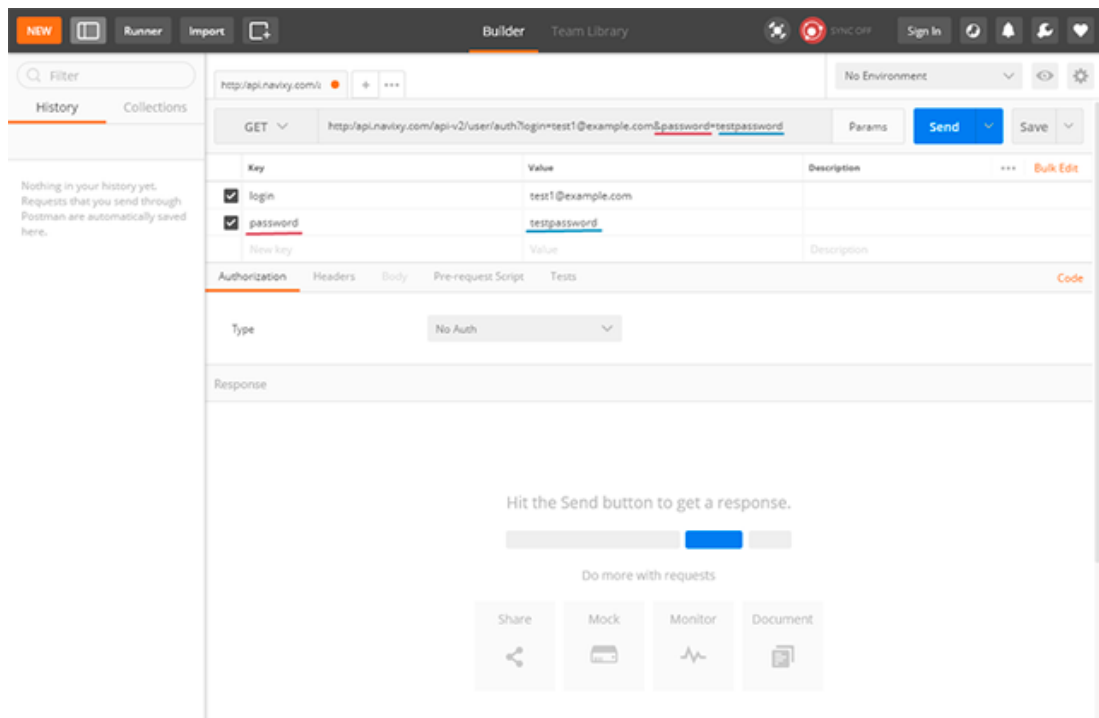
We will only ever need to fill 2 fields - Key (parameter name from documentation) and value. For user/auth request, we have 2 keys that should be transmitted - login and password.

You can see that once we fill out the parameter name - it is automatically added to the request line.

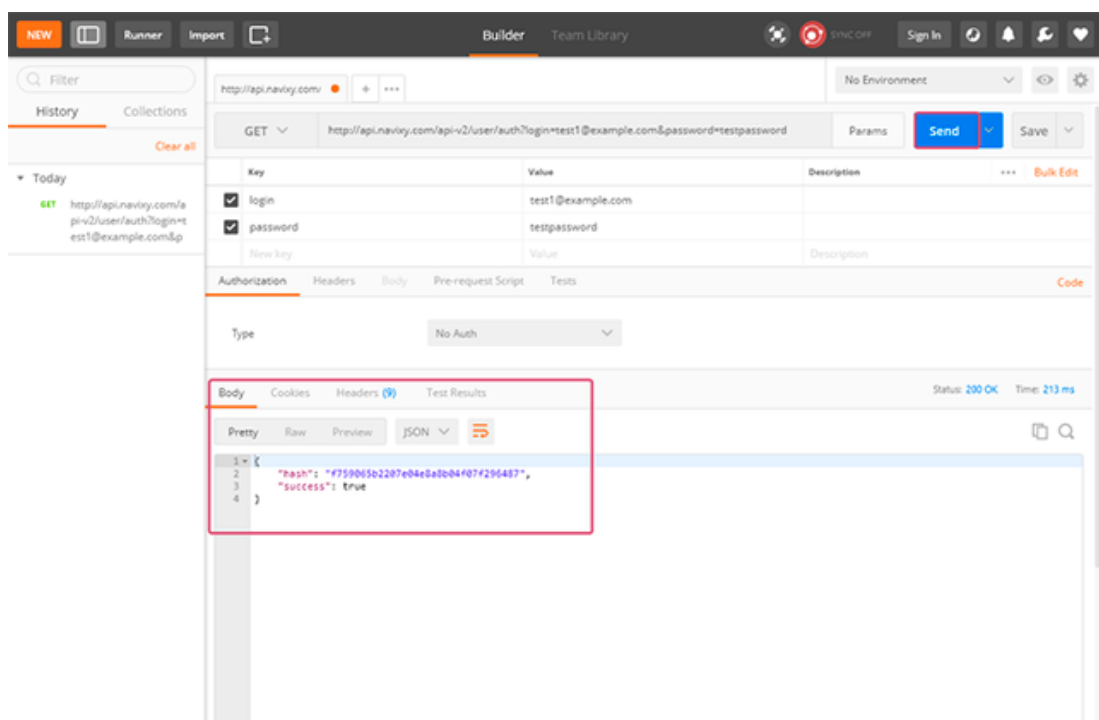


Similarly, with values and additional parameters:



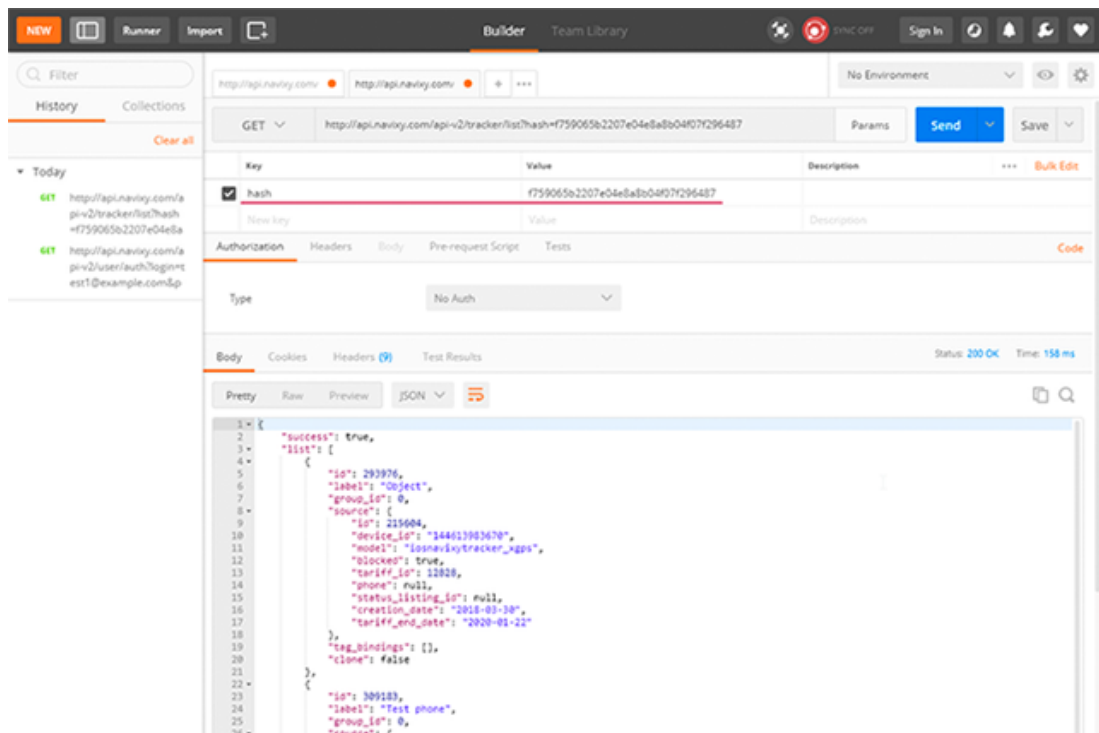


4. Press send, and you will see the reply, already split and highlighted for easier reading



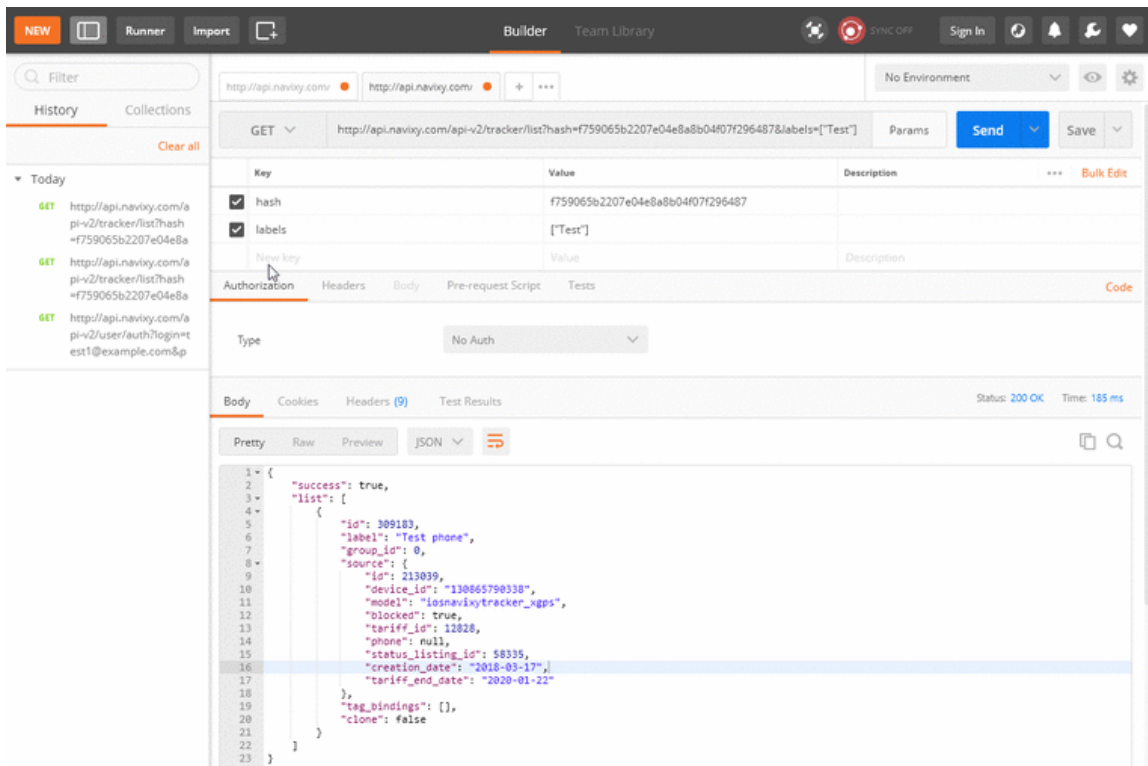
In this case, we have received a hash that should be copied and user for future requests.

Example: [tracker/list](#) request



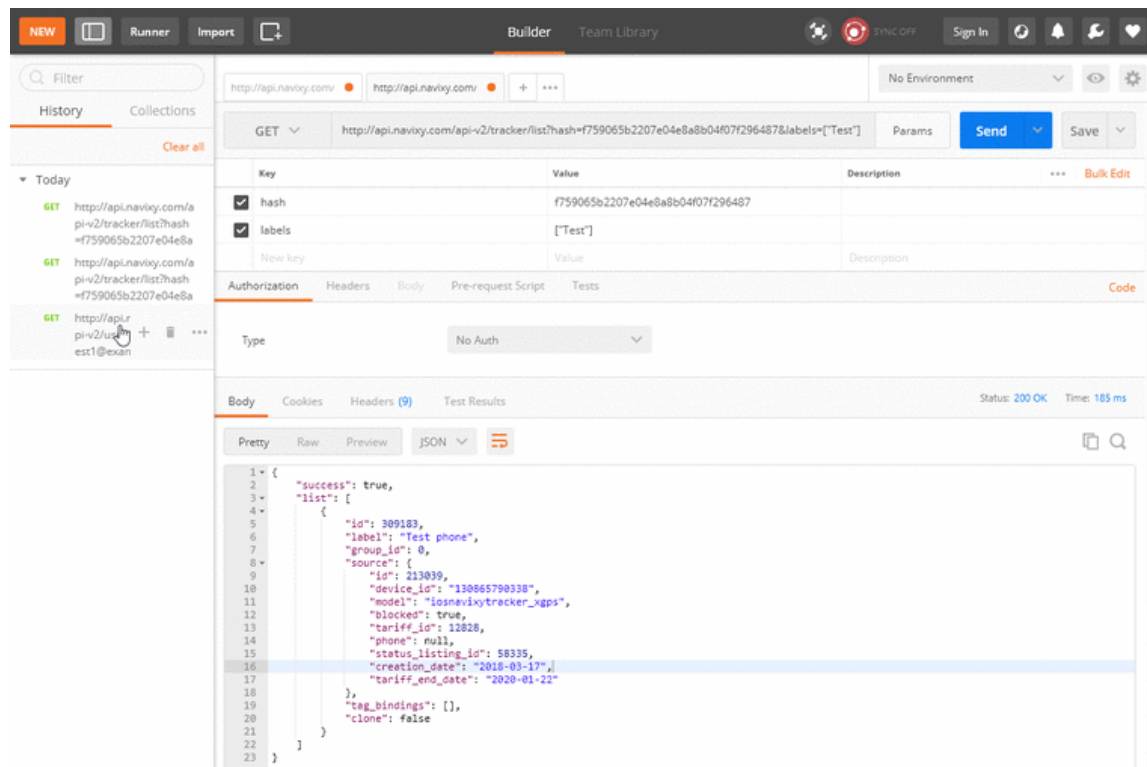
## Working with parameters

If your request has multiple parameters listed - you can easily enable and disable, preventing errors:



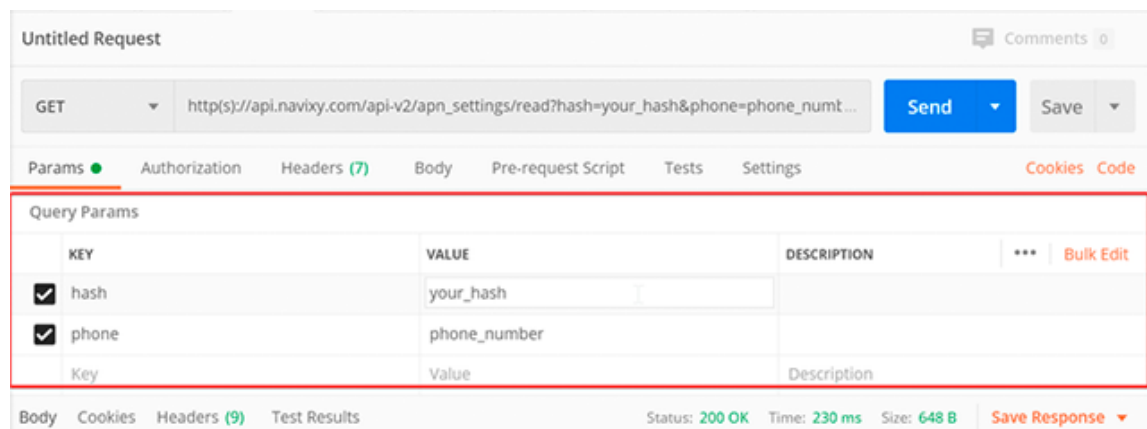
## History of requests

On the left side of postman application a history of your requests is stored. If you made errors or oo many changes and just want to go back to the old version or re-execute the request made in the past - a simple double-click will open a request in a new tab:



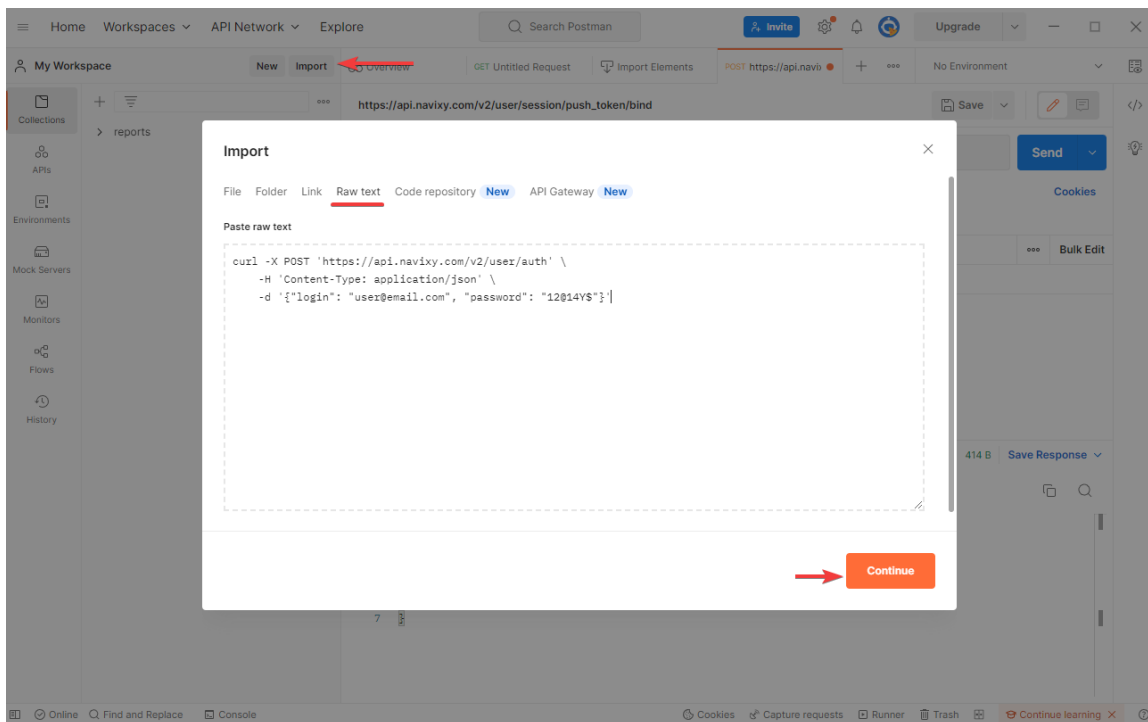
## Examples in documentation

You could see that our API documentation has both structure of the request and examples. You can copy them and paste in postman. In this case all parameters will be automatically separated to strings for more convenient edit.



## cURL examples in Postman

You can copy the cURL examples from our documentation and import them into Postman for later use. Copy an example, open an import tab and choose the Raw text. Then paste our example here and save the file.



## How to install

To get the latest version of the [Postman app](#), visit the download page and click "Download" for your platform.

Last update: December 20, 2022







# Navixy Backend API

## General

Each API resource semantically corresponds to some entity, for example: geofences, rules, objects, etc. The API calls for CRUD and other operations with these entities have similar names regardless the resource used: list, read, create, delete.

## Standard workflow (example)

Let us describe the standard workflow for API developer using very simple and most common example – requesting the track points data:

1. Determine [URL to API calls](#).
2. [Obtain hash of an API key](#)
3. Get objects lists with `tracker/list`.
4. Get track lists with `track/list`.
5. Get the track itself: `track/read`.



**You can get an API key via user's web interface. This is the recommended way instead of getting user session hash.**

In other words, to start working with API, the developers should have API call description (as provided herein), and know user login and password.

## API base URL

Depending on the physical location of the platform it will be:

- `https://api.eu.navixy.com/v2` for European Navixy ServerMate platform.
- `https://api.us.navixy.com/v2` for American Navixy ServerMate platform.
- `https://api.your_domain` for the self-hosted (On-Premise) installations.

For example, to make `user/auth` API call on the European Navixy ServerMate, you should use the URL:

```
https://api.eu.navixy.com/v2/user/auth
```

For white label solutions based on Navixy ServerMate platforms you can use your domain without `navixy`. In this case, use the same domain as for web interface and add `/api-v2` at the end:

```
https://your_domain/api-v2
```

The form is same for European and American Navixy ServerMate platforms without `eu` and `us` there.

## API calls format

Notation used in this doc:

```
/resource/sub_resource/action(parameter1,parameter2,[parameter3])
```

Which means that you should use the following URL:

```
[api_base_url]/resource/sub_resource/action
```

with named parameters:

- parameter1
- parameter2
- parameter3 is optional

Parameters can be passed in the:

- HTTP POST `application/json` with JSON content, **recommended**
- HTTP POST `application/x-www-form-urlencoded` with parameters in the request body
- HTTP GET - **not recommended**, should be used only for idempotent requests with small parameters size

#### HTTP POST `application/json`


```
$ curl -X POST '[api_base_url]/resource/sub_resource/action' \
-H 'Content-Type: application/json' \
-d '{"param1": "value1", "hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

#### HTTP POST `application/x-www-form-urlencoded`

```
$ curl -X POST '[api_base_url]/resource/sub_resource/action' \
-d 'param1=value1' \
-d 'hash=a6aa75587e5c59c32d347da438505fc3'
```

#### HTTP GET

```
$ curl '[api_base_url]/resource/sub_resource/action?
param1=value1&hash=a6aa75587e5c59c32d347da438505fc3'
```

 **Hash of an API key** is required for most API calls to identify user.

Typical actions:

- `list` – list all resource entities with IDs and minimum additional info
- `read` – read one entity by ID
- `update` – update one entity by ID
- `delete` – delete one entity by ID

## Request and response format

To make API call, for example, `resource/action` send `POST` request to

```
[api_base_url]/resource/action/
```

The response will be given with `application/json` content type, even errors (see [error handling](#)). Response fields and object structure is specific to API call.

#### Ensuring compatibility

Our API evolves over time, and new methods and JSON object fields are being added. We are doing our best to ensure our API remains backwards compatible with legacy API clients. However, you **must** ensure that any JSON object fields which are not supported by your app are **ignored**, and that in event if new JSON fields are returned, your application will not break. Also, sometimes, to reduce response size, JSON fields which are NULL are omitted. Your JSON parser should handle missing JSON fields as if they were NULL.

### For example

To read [user's tracker list](#) use `[api_base_url]/tracker/list/?hash=a6aa75587e5c59c32d347da438505fc3` and get response:

```
{
  "success": true,
  "list": [
    {
      "id": 560,
      "label": "GV55",
      "group_id": 12,
      "avatar_file_name": "super-avatar.jpg",
      "source": {
        "id": 2915,
        "model": "gv55lite",
        "blocked": false,
        "tariff_id": 2,
        "phone": "111",
        "status_listing_id": 333,
        "creation_date": "2014-02-02",
        "device_id": "8888888888888888"
      },
      "tag_bindings": [
        {
          "tag_id": 1,
          "ordinal": 1
        }
      ],
      "clone": true
    },
    {
      "id": 2799,
      "label": "2799",
      "group_id": 0,
      "source": {
        "id": 2692,
        "model": "m7",
        "blocked": true,
        "tariff_id": 5,
        "phone": null,
        "status_listing_id": null,
        "creation_date": "2006-02-10",
        "device_id": "3333333333333333"
      },
      "tag_bindings": [
        {
          "tag_id": 9,
          "ordinal": 3
        }
      ],
      "clone": false
    }
  ]
}
```

Or error if hash is wrong:

```
{
  "success": false,
  "status": {
    "code": 4,
    "description": "User or API key not found or session ended"
  }
}
```

## HTTP codes

If `success` is `true`, HTTP code is always `200 OK` (unless otherwise stated). If there is an error, HTTP code is `400 BAD REQUEST` (may vary depending on error type) (see [error](#)).

## Authorization and access levels

Unless otherwise noted, every API call requires a valid API Key hash (A string containing 32 hexadecimal characters) that can be passed (in order of lookup priority):

1. As `hash` parameter of the request body (root-level property for `application/json`).
2. As `hash` parameter of the HTTP query string.
3. As value of the HTTP header `Authorization` in the following form:

```
Authorization: NVX SessionHashValue
```

Following is pseudo-grammar that illustrates the construction of the `Authorization` request header:

```
ApiKeyValue = 32 hexademical characters;
Authorization = "NVX" + " " + ApiKeyValue ;
```

Read [how to get an API key](#).

## Data types

- `bool`, boolean - logical type: `true` of `false`.
- `byte` - signed 8 bits integer in range `[-128 .. 128]`.

- `short` - signed 16 bits integer in range `[-32,768 .. 32,767]` .
- `int` , integer - signed 32 bits integer in range `[-2,147,483,648 .. 2,147,483,647]` .
- `long` - signed 64 bits integer in range `[-9,223,372,036,854,775,808 .. 9,223,372,036,854,775,807]` .
- `float` - signed 32 bits float number `[3.40282347 x 10^38, 1.40239846 x 10^-45]` .
- `double` - signed 64 bits float number `[1.7976931348623157 x 10^308, 4.9406564584124654 x 10^-324]` .
- `string` - string literals.
- `enum` - string literals from predefined set.
- `date/time` - is a string containing date/time in [defined formats](#).
- `local_time` - is a string containing local time in `HH:mm:ss` format.
- `location` - is json object contains geographical coordinates, e.g.

```
{"lat": 34.178868, "lng": -118.599672}
```

- `locale` - string in format `language[_country]` , where `language` is [ISO 639 alpha-2](#) language code, and `country` is [ISO 3166 alpha-2](#) country code, e.g. `en_US` or `ru` . User interface support only language codes: `ru`, `en`, `es`, `ar`, `de`, `pt`, `ro` and `uk` .

## Date/time formats

Date/time type can be represented with formats:

- `yyyy-MM-dd HH:mm:ss` format (in user's timezone), default
- [ISO 8601](#) `yyyy-MM-dd'T'HH:mm:ssZZ`

To use ISO 8601 date/time format you should pass `true` to (in order of lookup priority):

1. `iso_datetime` parameter of the request body (root-level property for `application/json`).
2. `iso_datetime` parameter of the HTTP query string.
3. HTTP header `NVX-ISO-DateTime`



### JSON request body parameter

```
$ curl -X POST '[api_base_url]/resource/sub_resource/action' \
-H 'Content-Type: application/json' \
-d '{"iso_datetime": true, "hash":
"a6aa75587e5c59c32d347da438505fc3"}'
```

### form request parameter

```
$ curl -X POST '[api_base_url]/resource/sub_resource/action' \
-d 'iso_datetime=true' \
-d 'hash=a6aa75587e5c59c32d347da438505fc3'
```

### HTTP Header

```
$ curl -X POST '[api_base_url]/resource/sub_resource/action' \
-H 'Content-Type: application/json' \
-H 'NVX-ISO-DateTime: true' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

response example with fixed offset date/time :

```
{
  "success": true,
  "user_time": "2014-07-09T07:50:58+05:00",
  "list": [
    {
      "type": "odometer",
      "value": 100500.1,
      "update_time": "2014-03-06T13:57:00+05:00"
    }
  ]
}
```

response example with UTC date/time :

```
{
  "success": true,
  "user_time": "2014-07-09T02:50:58Z",
  "list": [
    {
      "type": "odometer",
      "value": 100500.1,
      "update_time": "2014-03-06T17:57:00Z"
    }
  ]
}
```

## Error handling

If an error occurs, API returns special error response. You can also detect error by checking HTTP response code. If it is not 200 OK, you should parse and handle

response body as an error response. In the event of error occurs, the response will be in the following format:

```
{
  "success": false,
  "status": {
    "code": 1,
    "description": "Database error"
  }
}
```

where `code` is one on the [error codes](#).

### Error codes

Default HTTP code is 400. Common error codes (should be handled for all API calls) are 1-100 and resource or action specific errors are 101-300.

| code | description                                | HTTP code |
|------|--|-----------|
| 1    | Database error                             | 500       |
| 2    | Service Auth error                         | 403       |
| 3    | Wrong hash                                 |           |
| 4    | User or API key not found or session ended |           |
| 5    | Wrong request format                       |           |
| 6    | Unexpected error                           | 500       |
| 7    | Invalid parameters                         |           |
| 8    | Queue service error, try again later       | 503       |
| 9    | Too large request                          | 412       |
| 11   | Access denied                              | 403       |
| 12   | Dealer not found                           |           |

| code | description                             | HTTP code |
|------|---|-----------|
| 13   | Operation not permitted                 | 403       |
| 14   | Database unavailable                    | 503       |
| 15   | Too many requests (rate limit exceeded) | 429       |
| 101  | In demo mode this function is disabled  | 403       |
| 102  | Wrong login or password                 |           |
| 103  | User not activated                      |           |
| 111  | Wrong handler                           |           |
| 112  | Wrong method                            |           |
| 201  | Not found in database                   |           |
| 202  | Too many points in zone                 |           |
| 203  | Delete entity associated with           |           |
| 204  | Entity not found                        | 404       |
| 206  | Login already in use                    |           |
| 207  | Invalid captcha                         |           |
| 208  | Device blocked                          | 403       |
| 209  | Failed sending email                    |           |
| 210  | Geocoding failed                        |           |
| 211  | Requested time span is too big          |           |
| 212  | Requested limit is too big              |           |

| code | description   | HTTP code |
|------|---|-----------|
| 213  | Cannot perform action: the device is offline                      |           |
| 214  | Requested operation or parameters are not supported by the device |           |
| 215  | External service error  |           |
| 217  | List contains nonexistent entities                                |           |
| 218  | Malformed external service parameters                             |           |
| 219  | Not allowed for clones of the device                              | 403       |
| 220  | Unknown device model  |           |
| 221  | Device limit exceeded   | 403       |
| 222  | Plugin not found  |           |
| 223  | Phone number already in use                                       |           |
| 224  | Device ID already in use  |           |
| 225  | Not allowed for this legal type                                   | 403       |
| 226  | Wrong ICCID   |           |
| 227  | Wrong activation code   |           |
| 228  | Not supported by sensor   |           |
| 229  | Requested data is not ready yet                                   | 404       |
| 230  | Not supported for this entity type                                |           |
| 231  | Entity type mismatch  | 409       |
| 232  | Input already in use  |           |

| code | description   | HTTP code |
|------|---|-----------|
| 233  | No data file  |           |
| 234  | Invalid data format                                       |           |
| 235  | Missing calibration data                                  |           |
| 236  | Feature unavailable due to tariff restrictions            | 402       |
| 237  | Invalid tariff  |           |
| 238  | Changing tariff is not allowed                            | 403       |
| 239  | New tariff does not exist                                 | 404       |
| 240  | Not allowed to change tariff too frequently               | 403       |
| 241  | Cannot change phone to bundled sim. Contact tech support. |           |
| 242  | There were errors during content validation               |           |
| 243  | Device already connected.                                 |           |
| 244  | Duplicate entity label.                                   |           |
| 245  | New password must be different                            |           |
| 246  | Invalid user ID   |           |
| 247  | Entity already exists                                     | 409       |
| 248  | Wrong password  |           |
| 249  | Operation available for clones only                       | 403       |
| 250  | Not allowed for deleted devices                           | 403       |
| 251  | Insufficient funds  | 403       |

| code | description  | HTTP code |
|------|--|-----------|
| 252  | Device already corrupted   |           |
| 253  | Device has clones  |           |
| 254  | Cannot save file   | 500       |
| 255  | Invalid task state   |           |
| 256  | Location already actual  |           |
| 257  | Registration forbidden   | 403       |
| 258  | Bundle not found   | 404       |
| 259  | Payments count not comply with summary                               |           |
| 260  | Payments sum not comply with summary                                 |           |
| 261  | Entity has external links  | 403       |
| 262  | Entries list is missing some entries or contains nonexistent entries |           |
| 263  | No change needed, old and new values are the same                    |           |
| 264  | Timeout not reached  | 403       |
| 265  | Already done   | 403       |
| 266  | Cannot perform action for the device in current status               | 403       |
| 267  | Too many entities  |           |
| 268  | Over quota   | 402       |
| 269  | Invalid file state   |           |
| 270  | Too many sensors of same type already exist                          |           |

| code | description        | HTTP<br>code |
|------|--------------------|--------------|
| 271  | File over max size | 413          |

Last update: March 26, 2024







# How to

Working with API might seem hard at first, but the goal of our documentation is to assist you in this process and make it more approachable.

Our "How to" section has step-by-step examples of working with the Navixy API.

From initial step of obtaining an API key hash to more complicated operations like retrieving a list of devices, tracks or creating reports. Using API and scripting you will be able to develop applications that not only satisfy your customer's needs but also help you make your business more profitable.

- [How to get hash of an API key.](#)
- [How to register a device.](#)
- [How to get tracker list.](#)
- [How to get track points.](#)
- [How to get sensors and counters data.](#)
- [How to use virtual sensors and get information from them.](#)
- [How to create geofences.](#)
- [How to create points of interest \(POIs\) and use custom fields with them.](#)
- [How to use rules.](#)
- [How to work with notifications.](#)
- [How to get push notifications.](#)
- [How to create/assign the tasks and optimize route.](#)
- [How to create forms for tasks.](#)
- [How to work with statuses.](#)
- [How to send commands to device via GPRS.](#)
- [How to obtain report's information.](#)
- [How to use tags.](#)
- [How to use service works.](#)
- [How to use driver journals.](#)
- [How to use BLE beacons along with trackers and Navixy APIs.](#)

Last update: August 1, 2023





## Obtaining hash of an API Key

"Hash", "Session key" or "API Key" is a randomly generated string that is used to verify and authenticate actions. The hash of API key must be passed in most API calls.



**You can get an API key in the user's web interface. This is the recommended way instead of user session hash.**

To get an API key, you can create it in the user's web interface. If it needs to be done automatically, you must first get a user's session hash, because creating a new API key using the another API key is not available.

You can get the user's session hash by [user/auth](#) call with credentials of a user:

```
https://api.navixy.com/v2/user/auth?  
login=user_login&password=user_password
```

The response will be like this:

```
{  
  "success": true,  
  "hash": "882fb333405d006df0d5a3f410115e92"  
}
```

Where resulting user's session hash is `882fb333405d006df0d5a3f410115e92` (just an example, you will get a different hex string).

After that, you need to get a [list of API keys](#) or create a new one using the [/api/key/create](#) call with obtained user's session hash:

```
https://api.navixy.com/v2/api/key/list?  
hash=882fb333405d006df0d5a3f410115e92&title=Integration+Key
```

The response will be like this:

```
{  
  "success": true,  
  "value": {  
    "hash": "c915157ac483e7319b0b257408bc04e1",  
    "create_date": "2021-10-29 12:00:36",  
    "title": "Integration Key"  
  }  
}
```

You must pass API Key hash value with most API calls along other parameters required to make the call. Otherwise, you will get an error response:

```
{
  "success": false,
  "status": {
    "code": 3,
    "description": "Wrong hash"
  }
}
```

Whenever you see such response, it means that you did not pass hash value properly.

Last update: August 1, 2023







# How to register a device

Instruction about device registration on the platform step by step.

It is possible to activate any GPS tracking device listed in the supported models list. Every model will be shown with all integrated input types, available rule types and other necessary information. We need to make several steps to get the device registered on the platform.

Step 1. Check that the platform support registering device model with [list\\_models](#) API call.

Step 2. Check all plugins available for the user with [plugin/list](#) request.

We are interested in the next plugin IDs that are used for registration:

- 44 - device registration with optional activation code.
- 37 - device registration with mandatory activation code.
- 35 - mobile app registration with optional activation code.
- 68 - mobile app registration with mandatory activation code.

Full information about activation codes and for what purposes they needed is [here](#).

Step 3. Register the device using the [tracker/register](#) action.

## Tracker registration

There is information about tracker registration with plugins 44 and 37.

### Common parameters

- phone - device's phone number with country code and without + sign.
- apn\_name - this is the apn that depends on your device's SIM GSM carrier. Max length 40.
- apn\_user - it depends on your device's SIM too. Max length 40, can be empty.
- apn\_password - this parameter depends on the GSM carrier as two previous parameters. Max length 40, can be empty.
- device\_id - device's ID. What ID type is used in your device can be found with [list\\_models](#) action and [ID type field](#)

- model - name of the model in the platform's code. It can be found in the [list\\_models](#) request too.
- label - label for the device.
- group\_id - tracker group ID, 0 if tracker does not belong to any group. The specified group must exist. See [group/list](#).
- plugin\_id - what parameter ID to use. It must be listed in available [plugins list for the user](#).
- activation\_code - optional string with activation code. Not necessary for plugin 44 and mandatory for plugin 37.

## Using plugin ID 44

For example, we have a Teltonika FMB 140 device with IMEI 986575154632586. SIM's phone is 999999999969 and APN settings are internet, user, and passwd. It is supported on the platform and user has the plugin 44. Activation codes are optional for this plugin. We don't need to register it to the special group, so the group\_id will be 0. The label should be as my car's plate number T571T0 for convenience.

The API call will be the next:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/register' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "label": "T571T0", "group_id": 0, "plugin_id": 44, "model": "telfmb140", "phone": "999999999969", "device_id": "986575154632586", "apn_name": "internet", "apn_user": "user", "apn_password": "passwd"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/register?hash=a6aa75587e5c59c32d347da438505fc3&label=T571T0&group_id=0&plugin_i
```

After sending the platform will respond with the next information:

```
{
  "success":true,
  "value":{
    "id":833389,
    "label":"T571T0",
    "group_id":0,
    "source":{
      "id":526383,
      "device_id":"986575154632586",
      "model":"telfmb140",
      "blocked":false,
```

```

        "tariff_id":12163,
        "phone":"999999999969",
        "status_listing_id":null,
        "creation_date":"2021-06-03",
        "tariff_end_date":"2021-06-17"
    },
    "clone":false
}

```

- Tracker object fields described [here](#).

## Using plugin ID 37

In this example we need to specify an activation code during the registration. All other information will be the same as for the plugin 44. In this case, we have empty apn\_user and apn\_password to show the usage.

The API call will be the next:

### cURL

```

curl -X POST 'https://api.navixy.com/v2/tracker/register' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "label":
"T571T0", "group_id": 0, "plugin_id": 37, "activation_code":
"6045325592", "model": "telfmb140", "phone": "999999999969",
"device_id": "986575154632586", "apn_name": "internet"}'

```

### HTTP GET

```

https://api.navixy.com/v2/tracker/register?
hash=a6aa75587e5c59c32d347da438505fc3&label=T571T0&group_id=0&plugin_i

```

The platform will confirm with the same information as for plugin 44.

## Mobile app registration

### Common parameters

- notification\_email - optional parameter. Notification with invitation to install the app will be sent to the specified in parameter email.
- notification\_phone - optional parameter. Invitation to install the app will be sent to the specified phone. Phone should be specified in international format without + sign.
- model - enum with model always the same = `mobile_unknown_xgps`.

- label - string with name of your device.
- group\_id - tracker group ID, 0 if tracker does not belong to any group. The specified group must exist. See [group/list](#).
- plugin\_id - what parameter ID to use. It must be listed in available [plugins list for the user](#).
- activation\_code - optional string with activation code. Not necessary for plugin 35 and mandatory for plugin 68.

## Using parameter 35

For example, we need to activate the app for our employee Andrew. So we can name the device with his name for convenience. Also, we will send an invitation by SMS using his phone number.

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/register' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "label": "Andrew", "group_id": 0, "plugin_id": 35, "model": "mobile_unknown_xgps", "notification_phone": "999877459965"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/register?
hash=a6aa75587e5c59c32d347da438505fc3&label=Andrew&group_id=0&plugin_i
```

The platform will notify us about success and with information about this device. The platform will automatically assign `device_id` to the app.

```
{
  "success": true,
  "value": {
    "id": 833997,
    "label": "Andrew",
    "group_id": 0,
    "source": {
      "id": 526785,
      "device_id": "186196632419",
      "model": "mobile_unknown_xgps",
      "blocked": false,
      "tariff_id": 12163,
      "phone": null,
      "status_listing_id": null,
      "creation_date": "2021-06-04",
      "tariff_end_date": "2021-06-18"
    },
    "clone": false
  }
}
```

```
}  
}
```

## Using plugin ID 68

If our user has mandatory activation codes (plugin 68) we should use this parameter when registering a new device.

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/register' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "label":  
"Andrew", "group_id": 0, "plugin_id": 68, "activation_code":  
"6045325592", "model": "mobile_unknown_xgps",  
"notification_phone": "999877459965"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/register?  
hash=a6aa75587e5c59c32d347da438505fc3&label=Andrew&group_id=0&plugin_i
```

The platform will respond with the same information as for plugin 35.

## The device doesn't register

There could be several reasons - why the device doesn't register. If we omit the problems with the SMS gateway, and it works perfectly - all issues listed [here](#). When we eliminated all possible issues and checked that everything works well we can send [tracker/register\\_retry](#) request to not create the same unit for the user. Moreover, it is not possible to register two devices with the same ID on the platform.

Last update: July 7, 2023





# How to get tracker list

Now we [have a hash of an API key](#) — let's start with essential basics.

Navixy has tracking device as a main unit, so most requests would require you to specify one or several tracker IDs. You can receive a list of all trackers in user's account with [tracker/list](#) API request:

## cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

## HTTP GET

```
https://api.navixy.com/v2/tracker/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

It will return to you

```
{
  "success": true,
  "list": [
    {
      "id": 123456,
      "label": "tracker label",
      "clone": false,
      "group_id": 167,
      "avatar_file_name": "file name",
      "source": {
        "id": 234567,
        "device_id": 9999999988888,
        "model": "telfmb920",
        "blocked": false,
        "tariff_id": 345678,
        "status_listing_id": null,
        "creation_date": "2011-09-21",
        "tariff_end_date": "2016-03-24",
        "phone": "+71234567890"
      },
      "tag_bindings": [{
        "tag_id": 456789,
        "ordinal": 4
      }]
    }
  ]
}
```

- `id` - int. Tracker ID aka `object_id`.
- `label` - string. Tracker label.



- `clone` - boolean. True if this tracker is clone.
- `group_id` - int. Tracker group ID, 0 when no group.
- `avatar_file_name` - string. Optional. Passed only if present.
- `source` - object.
  - `id` - int. Source ID.
  - `device_id` - string. Device ID aka source\_imei.
  - `model` - string. Tracker model name from "models" table.
  - `blocked` - boolean. True if tracker blocked due to tariff end.
  - `tariff_id` - int. An ID of tracker tariff from "main\_tariffs" table.
  - `status_listing_id` - int. An ID of the status listing associated with this tracker, or null.
  - `creation_date` - date/time. Date when the tracker registered.
  - `tariff_end_date` - date/time. Date of next tariff prolongation, or null.
  - `phone` - string. Phone of the device. Can be null or empty if device has no GSM module or uses bundled SIM which number hidden from the user.
- `tag_binding` - object. List of attached tags. Appears only for "tracker/list" call.
  - `tag_id` - int. An ID of tag. Must be unique for a tracker.
  - `ordinal` - int. Number that can be used as ordinal or kind of tag. Must be unique for a tracker. Max value is 5.

If account has a large amount of trackers, and you only need certain ones, you can add an optional filter parameter to the request that will only return matching records.

This parameter has the following constraints: \* labels array size: minimum 1, maximum 1024. \* no null items. \* no duplicate items. \* item length: minimum 1, maximum 60.

To get a list of trackers with labels matching the filter use this API call:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "labels":
["aa", "b"]}'
```

Last update: December 26, 2022





# How to get track points for trips

Sometimes necessary to get all points of a trip with more info about the device's moves. How to get them?

Firstly you need to [obtain hash of an API key](#).

Once you get the hash, you need to [get your tracker\\_id](#). The platform must know points for what device must be in reply.

Now you can get all points for the interesting period using [/track/read API call](#). Parameters that necessary for this call:

- `tracker_id` - we got them in [tracker/list](#) call. Use only one `tracker_id` per call. It should be an integer.
- `from` - a string containing start [date/time](#).
- `to` - a string containing end [date/time](#).

Full parameters description see at [/track/read API call](#).

The platform will reply:

```
{
  "success": true,
  "limit_exceeded": true,
  "list": [
    {
      "lat": 53.445181,
      "lng": -2.276432,
      "alt": 10,
      "satellites": 8,
      "get_time": "2011-06-18 03:39:44",
      "address": "4B Albany Road, Manchester, Great Britain",
      "heading": 298,
      "speed": 70,
      "precision": 100,
      "gsm_lbs": true,
      "parking": true
    }
  ]
}
```

- `limit_exceeded` - boolean. `true` if the requested time period exceeds limit specified in a tracker's tariff.
- `lat` - float. Latitude.
- `lng` - float. Longitude.

- `alt` - int. Altitude in meters.
- `satellites` - int. Number of satellites used in fix for this point.
- `get_time` - date/time. GPS timestamp of the point, in user's timezone.
- `address` - string. Point address. Will be "" if no address recorded.
- `heading` - int. Bearing in degrees (0..360).
- `speed` - int. Speed in km/h.
- `precision` - optional int. Precision in meters.
- `gsm_lbs` - optional boolean. `true` if location detected by GSM LBS.
- `parking` - optional boolean. `true` if point does not belong to track.

You can also [download](#) a KML file. You could use this file with map services. It is useful if you need to see all points on the map:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/track/download' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
123456, "from": "2020-09-23 03:24:00", "to": "2020-09-23
06:24:00", "format": "kml", "split": false}'
```

All parameters are the same with track/read plus two new optional parameters:

- `format` – string. File format, "kml" or "kmz". Default is "kml".
- `split` – boolean. If `true`, split tracks by folders with start/end placemarks and track line. Default `false`.

Last update: July 19, 2022





# How to get information from sensors and counters of tracker

Devices can be used not only to track GPS location. They can provide information about mileage, engine hours, measured from sensors like fuel level and temperature. All API calls to interact with devices can be found in [tracking/tracker](#) branch.

## Counters

Odometer allows to control a vehicle's mileage in real-time. The mileage readings can be based on the data received from a GPS tracking device or CAN bus.

Engine hours is a tool that allows owners of vehicles and special machinery to monitor engine running time and schedule maintenance works based on this data.

## Counter creation

To get information from counters they should be created. To create a counter use the call [value/set](#).

For example, we need to create odometer and engine hours counters. In this case we should use the next commands:

Odometer:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/counter/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 311852, "type": "odometer", "value": 98342.1}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/counter/read?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=311852&type=odometer&
```

Engine hours:



## cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/counter/read' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":  
311852, "type": "engine_hours", "value": 2368.2}'
```

## HTTP GET

```
https://api.navixy.com/v2/tracker/counter/read?  
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=311852&type=engine_ho
```

The platform will notify you about success in reply.

## Getting values from counter

Now we can get information from these counters when we need with the [get\\_counters](#) API call. With it the last update time and values of all counters in one call will be received. If necessary to get information from only specific counter type and one device then [value/get](#) call will be suitable. The same information can be obtained for the list of devices . In this case use [value/list](#) call.

## Counter values for a history period

Sometimes necessary to get data for the specific period with timestamps. For example, it may be necessary for insurances or governments. In this case should be used [data/list](#) call. It will return JSON with the next information:

## Response

```
{
  "success": true,
  "list": [
    {
      "value": 581321.0,
      "update_time": "2021-05-30 12:16:01"
    },
    {
      "value": 581322.0,
      "update_time": "2021-05-30 12:36:01"
    },
    {
      "value": 581323.0,
      "update_time": "2021-05-30 12:56:01"
    },
    {
      "value": 581324.0,
      "update_time": "2021-05-30 13:16:01"
    },
    {
      "value": 581325.0,
      "update_time": "2021-05-30 13:36:01"
    }
  ]
}
```

## Counted mileage and engine hours for a period

Sometimes we need counted values for a period, for example, how many kilometers the device travelled for 10 days or how many hours the engine was on. In this case we should use [stats/mileage](#) or [stats/engine\\_hours](#) calls.

## Sensors

The platform has two subtypes of sensors:

- [metering sensors](#) - Discrete sensors responsible for inputs states on the platform.
- [discrete sensors](#) - Measurement sensors will show information from variable types of sensors.

The list of all supported sensors can be found [here](#)

## Sensor creation

The ability to connect sensors, as well as their number may vary depending on the device model. Some sensors automatically creates by the platform. The list of these sensors depends on device model and information received from them. Some sensors should be created manually.

Full sensor creation has several steps:

1. Data sending from the sensor should be configured on the device's side, and it should be received by the platform. How to know - which one is received by the platform? The best way is connecting to [AirConsole](#).
2. We know - which sensor sends data and can choose one to [create](#)
3. If this is an analog sensor, or it is a sensor that sends data in uncalibrated values (for example, fuel sensor that sends percents instead of liters) - it should be [calibrated](#).

## Getting values from sensors

All sensors can be found in different widgets. Discrete widgets in the inputs' widget. Measurement sensors can be found in the sensors readings, OBD & CAN and Fuel level widgets. For every widget we have its own API call to get data:

- To get input states use [get\\_inputs](#) request.
- To get data from CAN and OBD sensors [get\\_diagnostics](#) API action.
- Data from the fuel sensors can be obtained using [get\\_fuel](#) call.
- Readings from metering not CAN, OBD and fuel sensors may be received with [get\\_readings](#) call.

## Getting values from all sensors and states

Also, you are able to get the data from all sensors of the device, its states and counters. Use [tracker/readings](#) request. It will reply with the next information:

```
{
  "success": true,
  "inputs": [
    {
      "label": "Board voltage",
      "units": "V",
      "name": "board_voltage",
      "type": "power",
```

```

        "value": 26.13,
        "units_type": "custom",
        "converted_units_type": null,
        "converted_value": null,
        "update_time": "2021-06-01 15:23:03"
    },
    {
        "label": "Analog sensor #1",
        "units": "",
        "name": "analog_1",
        "type": "fuel",
        "min_value": 0.0,
        "max_value": 450.0,
        "value": 269.82,
        "units_type": "litre",
        "converted_units_type": null,
        "converted_value": null,
        "update_time": "2021-06-01 15:23:03"
    }
],
"states": [
    {
        "field": "battery_level",
        "value": 4.01,
        "update_time": "2021-06-01 15:23:03"
    },
    {
        "field": "input_status",
        "value": 0,
        "update_time": "2021-06-01 15:23:03"
    },
    {
        "field": "movement_state",
        "value": "parked",
        "update_time": "2021-06-01 15:23:03"
    },
    {
        "field": "actual_track",
        "value": 34112,
        "update_time": "2021-06-01 12:58:03"
    },
    {
        "field": "output_status",
        "value": 3,
        "update_time": "2021-06-01 15:23:03"
    },
    {
        "field": "tcp_status",
        "value": 2,
        "update_time": "2021-06-01 15:23:05"
    }
]

```

```
}  
]
```

- input status and output status fields will show you binary information in decimal form. For example, output\_status field shows 3 - it is 11 in binary. The example device has two outputs. That's why 11 means output 1 = On and output 2 = ON.

## Getting data from multiple devices in one request

There is an API call that may optimize work with multiple devices at once. Just in case it is necessary to request the current data from a lot of devices in one account, use [tracker/readings/batch\\_list](#) API call. It allows getting the same information as in the previously described method for multiple trackers.

## Getting historical data from sensors

It may be necessary to get historical data from measurement sensors. In this case, you can use [tracker/sensor/data/read](#) API call which allows you to get all provided values from a sensor in a period of 30 days. Choose the necessary sensor and specify its ID. The list of a device's sensors with IDs you can get with [sensor/list](#) request.

Last update: August 1, 2023





# Virtual sensors usage

In the world of IoT and telematics, understanding and interpreting data is crucial. This is where virtual sensors come into play. These powerful tools help users comprehend data from various device sensors in a more meaningful way, often translating it into text form for easier understanding. In this article, we'll delve into what virtual sensors are, how they work, and how to configure them.

Find this instruction with operations in UI only in our [Expert center](#).

## What are virtual sensors

Virtual sensors interpret and translate raw data from the physical sensors of a device, making it comprehensible and actionable. For example, they can get data from state fields or even from a certain bit in a particular field.

One of the standout features of virtual sensors is their ability to monitor ignition, even on devices where no ignition input is provided or that input is already occupied by other needs. This allows users to read ignition from any desired source, be it motion, RPM, or voltage.

The information from virtual sensors can be viewed in several ways:

- Current readings in widgets.
- Historical readings in reports.
- Alerts when certain values are received in rules.

The virtual sensor object consists of the following parameters:

```
{
  "type": "virtual",
  "id": 1700049,
  "sensor_type": "virtual_ignition",
  "name": "Virtual Ignition",
  "input_name": "board_voltage",
  "parameters": {
    "calc_method": "in_range",
    "range_from": 13.4,
    "value_titles": [{
      "value": "0",
      "title": "Off"
    }, {
      "value": "1",
      "title": "On"
    }
  ]
}
```



```

    }
}
}

```

- `type` - string. This should be set as `virtual` for virtual sensors.
- `id` - int. This is the sensor's ID.
- `sensor_type` - enum. Must be "virtual\_ignition" for virtual ignition sensor or "state" for others.
- `name` - string. Your name of a sensor. May contain up to 100 characters.
- `input_name` - string. A source input field (identifier). It indicates from which sensor the information is received by the platform.
- `parameters` - an object with additional parameters.
- `calc_method` - enum. This defines the method of sensor value calculation. It must be one of the following: `in_range`, `identity`, `bit_index`.
- `range_from` - double. Lower boundary of the range and is only used with the "in\_range" calculation method.
- `range_to` - double. Upper boundary of the range and is only used with the "in\_range" calculation method.
- `bit_index` - int, [1..N]. A bit index in the input field source value and is only used with the "bit\_index" calculation method.
- `value_titles` - a mapping for assigning special titles for sensor values, if required.
- `value` - string. Sensor value - raw value that comes from a device. Max size 64 chars.
- `title` - string. Your title for the sensor value. Max size 64 chars.

!!! note "There can only be one virtual sensor of type `virtual_ignition` for each tracker. For the "in\_range" calculation method, one or both fields "range\_from" and "range\_to" must be specified. The "bit\_index" field is mandatory for the "bit\_index" calculation method. All values within "value\_titles" must be unique."

1## Where virtual sensors types are useful

## Value in range

A 'Value in Range' sensor is an effective tool for maintaining essential parameters, such as virtual ignition, temperature, humidity, and fuel level, within a specified range. It functions on a simple principle:

- if a sensor value falls within defined boundaries, it equates to 1 (your A value).

- if it's outside these boundaries, it corresponds to 0 (your B value).

### **To get virtual ignition**

If your device is without an ignition input or all physical inputs are used on it, a virtual ignition can be utilized to detect the ignition state. This process works by detecting a significant increase in the car's onboard voltage when the engine is turned on. This change in voltage can then be used as a signal to determine whether the engine is running or not. Typically, if the board voltage exceeds 13.2 V, it's a clear indication that the engine is operational.

### **To get sensor values into understandable format**

This example parallels the one we just discussed about setting up a virtual ignition, but in this case, instead of monitoring a vehicle's ignition, we're keeping tabs on temperature.

Let's say you have an analog sensor that gathers temperature data. For instance, it might output 1020 for -10°C, and 1900 for 0°C. It's important to note that the information from these analog sensors comes uncalibrated, which means you'll need to specify it in this raw form when setting up your virtual sensor.

So, by following this method, you can translate complex sensor values into a simplified, more understandable format.

### **Source value**

With virtual sensors, you have the flexibility to assign your own definitions to any values received. This feature is particularly handy when dealing with predefined sets of values or strings, allowing you to work with static values without the need to specify different ranges. Plus, it's adaptable to any data you require.

For instance:

- 0/1
- true/false
- on/off
- open/close
- armed/disarmed
- state 1/state 2/state 3
- key 1/key 2/key 3, etc.

The mode operates in the following way:

- When value 1 is received, that's designated as your value A.
- When value 2 arrives, that becomes your value B.
- When value 3 is transmitted, that's identified as your value C, and so forth.

The best way to get historical readings on this calculation method sensors is to use it with state field value alert with report on all events.

Let's clarify this functionality with a practical example.

#### **When you need to define every sensor value to understand the readings**

Certain sensors might supply different numerical values to a platform. For example, let's say we have a truck equipped with a PTO drive engagement sensor that only outputs the following values:

- 0 - No PTO drive is engaged
- 1 - At least one PTO drive is engaged
- 2 - Error
- 3 - Not available

With virtual sensor you can get useful information instead of codes like 0, 1 or others.

#### **Hardware key readings for drivers, equipment and trailers**

Certain devices have the ability to recognize drivers and their iButtons, RFID keys, or equipment linked via Bluetooth sensors. The platform can identify the closest equipment or driver to the device, and a Virtual Sensor can display these names.

The simplest method of identification is through the use of tags. Each unit that's connected to heavy equipment has its own sensor with an attached tag. This tag serves as a hardware key that's recognizable by the platform. When the unit is connected to the machine, this key is transmitted to the platform.

Just like the way we named values for PTO, the associated name of this key can be displayed in an easily understandable format. This ensures that you always know which unit is communicating with your machine.

#### **Event code readings**

Navixy platform has the capability to provide you with the most recent event code received from your device. In this scenario, you can select the event code input and

define the appropriate event codes to be displayed in widgets. For instance, if you're using a Jimi JC400, you can access Driver Monitoring System (DMS) events.

## Bit index

Certain devices might transmit complex data in their packets, sometimes consolidating several parameters into a single value. The Virtual Sensors feature gives you the ability to interact with segments of telematics values, helping you decode the data sent by the GPS hardware.

For instance, let's say a value of 011 is transmitted. We must first interpret this information in little endian (from the right to the left) according to the protocol:

- 1 - Indicates the status of the driver's seat belt: 0 signifies it's fastened, 1 means it's unfastened. (Bit 1)
- 1 - Displays the status of the driver's door: 0 means it's closed, 1 indicates it's open. (Bit 2)
- 0 - Represents the condition of the hood: 0 means it's closed, 1 indicates it's open. (Bit 3)

Each position in the parameter reflects the status of different vehicle systems. To configure and display these, you'll need to create a separate sensor for each parameter.

## Create

You can create such sensor with [tracker/sensor/create](#) request. For example, we need to track the virtual ignition on a device by the board voltage. In this case, we should use call:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/sensor/create' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
123456, "sensor": {"type": "virtual", "sensor_type":
"virtual_ignition", "name": "Virtual Ignition", "input_name":
"board_voltage", "parameters": {"calc_method": "in_range",
"range_from": 13.4, "value_titles": [{"value": "0", "title":
"Off"}, {"value": "1", "title": "On"}]}}'
```

**Don't use `sensor_id` parameter in the sensor object since there is no sensor with any ID before it is created.**

The platform will notify you about the result with the assigned ID to this newly created sensor.

## Reconfigure

In case you want to change something in the virtual sensor settings, you can use [tracker/sensor/update](#) API call. For instance, we should change the range from which the virtual ignition must be calculated.

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/sensor/update' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "sensor": {"type": "virtual", "sensor_id": 965837, "sensor_type": "virtual_ignition", "name": "Virtual Ignition", "input_name": "board_voltage", "parameters": {"calc_method": "in_range", "range_from": 13.7, "value_titles": [{"value": "0", "title": "Off"}, {"value": "1", "title": "On"}]}}'
```

## Get values per period

When you want to show sensor readings or probably generate your own report it will be useful to get information in form value - time. In this case, the API call [tracker/sensor/data/read](#) will help.

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/sensor/data/read' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "sensor_id": 965837, "from": "2023-07-24 00:00:00", "to": "2023-07-24 23:59:00", "raw_data": false}'
```



**Use raw\_data: true in case, you need to get the raw sensor values per period.**

## Get the current values

It is possible to read all current values from all virtual and measurement sensors and counters from multiple devices at once with [tracker/readings/batch\\_list](#) API call. It will

provide you with all current information per one call so your app may request a lot of other requests without getting the limit 50 calls per second.

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/readings/
batch_list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3",
"trackers": [10181215,10038816]}'
```

## Get readings with reports

It is possible to get information about sensor readings within reports. Let's describe every report type and provide some examples.

### Equipment working time report

The equipment working time report reveals the operational times of any unit linked to discrete or virtual inputs with a calculation method value in the range or bit index. It allows you to learn about the operating time of the equipment both while moving and stationary, obtain daily activity data, and pinpoint when and where the equipment was activated.

Report may be generated with [plugin 12](#). Use the next example for reference:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/
generate' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "title":
"Equipment working time", "trackers": [642546], "from":
"2023-07-27 00:00:00", "to": "2023-07-27 23:59:59", "time_filter":
{"from": "00:00:00", "to": "23:59:59", "weekdays":
[1,2,3,4,5,6,7]}, "plugin": {"hide_empty_tabs":true,"plugin_id":
12,"show_seconds":false,"min_working_period_duration":
60,"show_idle_percent":true,"filter":false,"sensors":
[{"tracker_id":642546,"sensor_id":1931610}]}'
```

### Engine hours report

The engine hours report provides the working duration for ignition-based sensors. It offers valuable insights into the operation time of your ignition-based equipment,

whether it's on the move or idle. The report also delivers daily activity data, enabling you to identify precisely when and where the ignition was on.

Generate report using [plugin 7](#). The next example shows correct API request for that:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/
generate' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "title":
"Engine hours report", "trackers": [642546], "from": "2023-07-27
00:00:00", "to": "2023-07-27 23:59:59", "time_filter": {"from":
"00:00:00", "to": "23:59:59", "weekdays": [1,2,3,4,5,6,7]},
"plugin": {"hide_empty_tabs":true,"plugin_id":
7,"show_seconds":false,"show_detailed":true,"include_summary_sheet":tr
```

## Measuring sensors report

The measuring sensors report displays data from any configured measurement sensors or virtual sensors with a calculation method source value for a selected period. It enables you to view both graphical and statistical information from your device's sensors.

This report can be generated with [plugin 9](#). For instance:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/
generate' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "title":
"Measuring sensors report", "trackers": [1685505], "from":
"2023-07-27 00:00:00", "to": "2023-07-27 23:59:59", "time_filter":
{"from": "00:00:00", "to": "23:59:59", "weekdays":
[1,2,3,4,5,6,7]}, "plugin": {"hide_empty_tabs":true,"plugin_id":
9,"details_interval_minutes":
5,"graph_type":"time","smoothing":true,"show_address":false,"filter":t
[{"tracker_id":1685505,"sensor_id":613753}]}'
```

## Vehicle readings report

The vehicle readings report showcases data gathered from your vehicle's instruments via the CAN/OBD or virtual sensors for any chosen time frame. This includes information such as mileage, engine RPMs, speed, fuel consumption, coolant temperature, and more.

Report is available with [plugin 22](#):

## cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/
generate' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "title":
"Vehicle readings report", "trackers": [642546], "from":
"2023-07-27 00:00:00", "to": "2023-07-27 23:59:59", "time_filter":
{"from": "00:00:00", "to": "23:59:59", "weekdays":
[1,2,3,4,5,6,7]}, "plugin": {"hide_empty_tabs":true,"plugin_id":
22,"details_interval_minutes":
30,"graph_type":"time","smoothing":false,"sensors":[{"tracker_id":
642546,"sensor_id":1866139}]}'
```

## Report on all events

Reports on all events is handy for obtaining information about specific states received by the sensor with the aid of rules.

Generate this report type with [plugin 11](#).

## cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/
generate' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "title":
"Event report", "trackers": [642546], "from": "2023-07-27
00:00:00", "to": "2023-07-27 23:59:59", "time_filter": {"from":
"00:00:00", "to": "23:59:59", "weekdays": [1,2,3,4,5,6,7]},
"plugin": {"hide_empty_tabs":true,"plugin_id":
11,"show_seconds":false,"group_by_type":false,"event_types":
["state_field_control","sensor_inrange","sensor_outrange"]}'
```

For obtaining information from reports, follow to our [instructions](#).

## Rules for virtual sensors

Since we have the possibility to generate events report, let's describe rules. There are a couple of rules you can implement to receive alerts based on virtual sensor values:

### Parameter in range

Parameter in range rule is associated with the use of measurement sensors. Its function is to generate a notification whenever the sensor data received by the platform falls within or outside a specified range.



## cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/rule/create' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "rule":
{"name": "Parameter in range", "description": "Rule for getting
alert on specific range", "alerts": {"sms_phones": [], "emails":
[], "phones": [], "push_enabled": true }, "extended_params":
{"sensor_id": 1991090, "threshold": 0, "min": 2, "max":
3 }, "primary_text": "Sensor value out range", "secondary_text":
"Sensor value in range", "suspended": false, "trackers":
[642546], "type": "sensor_range", "param": null, "zone_ids":
[], "schedule": [{"type": "weekly", "from": {"weekday": 1, "time":
"00:00:00"}, "to": {"weekday": 7, "time": "23:59:59"} }]}'
```

## State field value

State field value is used to monitor any virtual sensor states that you define in the state column when configuring the virtual sensor. As soon as that state is received, you'll be notified.

## cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/rule/create' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "rule":
{"name": "State field value", "description": "Rule for getting
specific states", "alerts": {"sms_phones": [], "emails":
[], "phones": [], "push_enabled": true }, "extended_params":
{"state_field_index": null, "state_field_max_index":
null, "virtual_sensor_id": 2136502, "trigger_value":
"1", "allow_repeat": false, "repeat_delay_seconds":
null, "state_field_index_max": null }, "primary_text": "Eye sensor:
movement detected", "secondary_text": "", "suspended":
false, "trackers": [642546], "type": "state_field_control", "param":
null, "zone_ids": [], "schedule": [{"type": "weekly", "from":
{"weekday": 1, "time": "00:00:00"}, "to": {"weekday": 7, "time":
"23:59:59"} }]}'
```

To obtain notifications on these rules refer to [the instruction](#).

Last update: August 8, 2023





# How to work with geofences

Geofence is a virtual perimeter for a real geographic area. The system can control whether object crossed geofence border (either "in" or "out"). All these events are logged, so user can obtain geofence reports and receive alerts.

Moreover, you can assign various rules for events to particular geofences. For example, if you need to get speeding alerts only within a certain area (e.g. in city) or route.

## Geofence creation

To create a geofence we should use the [zone/create](#) API call. We have several types of them. So the call and process can be different between them.

### Circle geofence

It is the easiest geofence to create. There we should use only one point as a center and radius. The platform will automatically calculate borders for it.

For example, we want to create a geofence with a radius 50 meters that will cover a business park to track employees. We need this geofence to create a rule that will provide an alert when they will come to work and another one - when they go from it.

API request:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/zone/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "zone":
{"label": "Circle geofence", "type": "circle", "center": {"lat":
61.49504550221769, "lng": 23.775476217269897}, "radius": 50,
"tags": [179227], "color": "03A9F4", "address": "Address"}}'
```

The platform will respond with status and created geofence ID. We can use this ID to [create a rule](#).

### Polygon geofence

The second type we will create - the polygon geofence. It is more difficult than the circle geofence because we should specify special points for it where the geofence border

will change direction. Maximum count of points is 100. This limitation is necessary because a geofence is not just a visual display of some area. The platform calculates the data for reports and alerts on the fly. When the number of points in a geofence is more than 100, computational costs begin to grow exponentially.

For example, we want to track maximum speed of our vehicles in Rome. To do that, we will need to create a geofence that covers the city.



**Geofence accuracy can be fairly low as long as it's border crosses all the main roads.**

API request:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/zone/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "zone":
{"label": "Speed limit in Rome", "type": "polygon", "color":
"27A9E3", "address": "Address"}, "points": [{"lat":
41.80970819375622, "lng": 12.576599121093752, "node": true},
{"lat": 41.79128073728445, "lng": 12.522354125976564, "node":
true}, {"lat": 41.80970819375622, "lng": 12.38983154296875,
"node": true}, {"lat": 41.86649282301996, "lng":
12.369232177734375, "node": true}, {"lat": 41.90943147946872,
"lng": 12.38090515136719, "node": true}, {"lat":
41.956426414614235, "lng": 12.379531860351562, "node": true},
{"lat": 41.98501507352485, "lng": 12.435150146484375, "node":
true}, {"lat": 41.98807738309159, "lng": 12.50724792480469,
"node": true}, {"lat": 41.97531678812783, "lng":
12.54913330078125, "node": true}, {"lat": 41.95795827518022,
"lng": 12.580718994140627, "node": true}, {"lat":
41.92322706102551, "lng": 12.61161804199219, "node": true},
{"lat": 41.902277040963696, "lng": 12.619171142578127, "node":
true}, {"lat": 41.86904950322354, "lng": 12.607498168945312,
"node": true}]}'
```

Don't forget that the rule isn't created yet. The platform will respond with geofence ID. Use [the next instruction](#) to create rules.

## Sausage geofence

We use sausage geofences for roads. They should be more accurate, and their calculated area is not so hard as for polygons. That's why the maximum number of points is 1024.

For example, we need to create special geofences for street cleaning cars, and we want to see - is this car cleaned the street, or it turned from it in the middle.

## cURL

```
curl -X POST 'https://api.navixy.com/v2/zone/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "zone":
{"label": "Clean street 1", "type": "sausage", "radius": 20,
"color": "27A9E3", "address": "Address"}, "points": [{"lat":
21.5337018035, "lng": -104.8700889945, "node": true}, {"lat":
21.5336107362, "lng": -104.8691622913, "node": true}, {"lat":
21.5336444186, "lng": -104.8674470186, "node": true}, {"lat":
21.5336494086, "lng": -104.8656499386, "node": true}, {"lat":
21.5341084873, "lng": -104.8656606674, "node": true}, {"lat":
21.5341434171, "lng": -104.8661112785, "node": true}, {"lat":
21.534742213, "lng": -104.8656713963, "node": true}, {"lat":
21.5350266402, "lng": -104.8659932613, "node": true}, {"lat":
21.5336593886, "lng": -104.8669320345, "node": true}, {"lat":
21.5336469136, "lng": -104.8691529036, "node": true}, {"lat":
21.5337367335, "lng": -104.8700594902, "node": true}, {"lat":
21.5338427707, "lng": -104.8705852032, "node": true}, {"lat":
21.5341184672, "lng": -104.8718833923, "node": true}, {"lat":
21.5344577853, "lng": -104.873329103, "node": true}, {"lat":
21.5346199591, "lng": -104.8735275865, "node": true}, {"lat":
21.532277154, "lng": -104.8760032654, "node": true}, {"lat":
21.5312941127, "lng": -104.8770868778, "node": true}, {"lat":
21.5301214405, "lng": -104.8784118891, "node": true}, {"lat":
21.5291383846, "lng": -104.8793131113, "node": true}, {"lat":
21.5287790935, "lng": -104.8795759678, "node": true}, {"lat":
21.5284647131, "lng": -104.8797154427, "node": true}, {"lat":
21.5280804693, "lng": -104.8797905445, "node": true}, {"lat":
21.5276413324, "lng": -104.879822731, "node": true}, {"lat":
21.5273668712, "lng": -104.8799729347, "node": true}]}'
```

The platform will provide the status, and geofence ID.

The sausage geofence also, could be used to create a special route for cars with valuable cargo, such as cash collectors. Or for patrol cars. In this case, use the rule "deviation from the route".

## Getting geofence name by a tracker's location

It may be necessary to get the geofence name or ID where a device is located. In this case, use [zone/search\\_location](#). For example, we want to get a geofence, where our device is located, or we want to count how many devices are in some zone.

To get this information we should request a device's [state and location](#) first. With received lat and lng parameters we can check geofences.

Last update: January 15, 2024







# How to create and work with points of interest.

Points of interest (POI) or places can be used for different purposes. They can help you organize your list of frequently visited clients, simplify your work with tasks, and can be used to analyze your business with reports.

Custom fields have also been designed for them, which can be used to add additional necessary information about locations and customers. They can be used for creation of your own CRM or ERP system, as well as for easy integration with third party systems. It is possible to add a phone number, e-mail, and other relevant customer data. To get to the next level, it's possible to assign specific employees to a customer.

Here we will describe - how places with custom fields can be created and used.

## Creation of fields and POIs

Before we start using fields and POIs we should create them. Our purpose is to create a new customer with necessary information and assign an employee to him. This employee will be able to see the all information in his mobile app. The place object described [here](#).

Example: For our own CRM system we need to have the next fields: \* Label - there will be our customer's name. \* Address - full address where our customer located. \* Description - additional description about the customer. Like the working hours or something specific. \* Tags - here we will add tags. They will be useful to ease up searching and using for tasks in UI. \* E-mail - customer's email. \* Phone - customer's phone number. \* The last visit date - we are interested to see the last visit of customer. If the last visit will be more than X days we will notify our employee about that. \* The last order № - to ease up the searching of the last customer's order\_id. \* The last visit result - a text field where our employee can specify information about results of his last visit. \* Responsible employee - this field is necessary to assign a place to our responsible employee for this address. He will be able to see and change necessary information using his mobile app.

## Custom fields

Some fields are default and can't be changed. They are Label, Address, Description and Tags. All other necessary fields should be created by ourselves.



Custom fields we change here will be added to all places we have and will create.

First we should get the entity ID to know - what entity we should update and where to add fields.

API request:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/entity/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

In a reply we will receive a necessary entity ID with already existing fields in it. We should add new fields in this entity. We add only not existing fields that's why we will not list them in our request.

API request:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/entity/fields/update' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3",
"delete_missing": true, "entity_id": 520, "fields": [{"label": "E-mail", "required": false, "type": "email", "description": "Customer's email"}, {"label": "Phone", "required": false, "type": "phone", "description": "Customer's phone"}, {"label": "The last visit date", "required": false, "type": "text", "description": null}, {"label": "The last order №", "required": false, "type": "text", "description": null}, {"label": "The last visit result", "required": false, "type": "text", "description": null}, {"label": "Responsible employee", "params": {"responsible": true}, "required": false, "type": "employee", "description": null}]}'
```

The platform will confirm our update with:

```
{
  "success": true,
  "list": [
    {
      "id": 2327,
      "label": "E-mail",
      "required": false,
      "description": "Customer's email",
      "type": "email"
    },
    {
      "id": 2328,
      "label": "Phone",
      "required": false,
      "description": "Customer's phone",
      "type": "phone"
    }
  ]
}
```

```

    {
      "id": 2329,
      "label": "The last visit date",
      "required": false,
      "description": null,
      "type": "text"
    },
    { "id": 2330,
      "label": "The last order №",
      "required": false,
      "description": null,
      "type": "text"
    },
    { "id": 2331,
      "label": "The last visit result",
      "required": false,
      "description": null,
      "type": "text"
    },
    { "id": 2332,
      "label": "Responsible employee",
      "required": false,
      "description": null,
      "params": { "responsible": true },
      "type": "employee"
    }
  ]
}

```

When the reply received we know what IDs our fields have, and we can change their order in the entity. Now we should update our entity.

API request:

#### cURL

```

curl -X POST 'https://api.navixy.com/v2/entity/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "entity":
{"allowed": true, "id": 520, "type": "place", "settings":
{"layout": {"sections": [{"label": "Places", "field_order":
["label", "location", "description", "tags", "2327", "2328",
"2329", "2330", "2331", "2332"]}}}}'

```

## POIs creation

We have successfully set the fields for all locations, and now we need to create a location. The names of the fields already contain - what information we would like to see for each place. So we just need to fill in the parameters of these fields according to the client's data.

API request:

## cURL

```
curl -X POST 'https://api.navixy.com/v2/place/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "place":
{"label": "Company1", "description": "accepted one more deal for 7
devices on the next week", "files": [], "fields": {"2327":
{"value": "shop1@email.com", "type": "email"}}, {"2328": {"value":
"555231415221", "type": "phone"}}, {"2329": {"value":
"10/10/2021", "type": "text"}}, {"2330": {"value": "87292",
"type": "text"}}, {"2331: {value: "Sold 10 devices", type:
"text"}}, {"2332": {"value": 71247, "type": "employee"}}},
"location": {"address": "Lovell House, 6 Archway, Hulme,
Manchester M15 5RN, UK", "lat": 53.46583133200717, "lng":
-2.2464680671691895, "radius": 50}, "tags": [218916]}'
```

The platform will confirm creation with:

```
{
  "success":true,
  "id":1521307
}
```

- `id` - int. An ID of the created place. It can be used for obtaining and updating the place object.

## Obtaining and updating information about places

- To get information about place objects (for example, to pull this data to your CRM) use the [place/list](#) API call.
- To get a count of visits for places [generate](#) a report with [ID 85](#).
- To update information about place use [place/update](#) API call.

## Getting POI name by a tracker's location

It may be necessary to get the POI name or ID where a device is located. In this case, use [place/search\\_location](#). For example, we want to get a place, where our device is located, or we want to count how many devices are in some place.

To get this information we should request a device's [state and location](#) first. With received lat and lng parameters we can check places.

Last update: December 26, 2022





# How to use rules

Rules used to set up conditions according to which the system logs the events and sends notifications to user.

When a server receives a new portion of data from the device, it checks whether the conditions set are true or false for this data. If they are true, the server generates an event in history, logs it and immediately sends SMS, push message or email and saves event in history.

## Create

To start work the rule must be created. Let's create a rule with conditions according to which the platform will generate events and schedule intervals when this rule should work using the [rule/create](#). The user must have access to rule update.

Necessary parameters for this call. Availability of some parameters depends on used rule type:

- `name` - A string containing a name of created rule.
- `description` - A string containing rule's description.
- `zone_ids` - An int array. A list of zones to bind where the rule will work. Leave it empty if rule should work everywhere. Parameter `zone_ids` is not allowed for rule `offline` and required for `route` and `inoutzone` rule types.
- `trackers` - An int array. A list of tracker IDs belong to user for which the rule will work.
- `type` - A string containing one of pre-defined types of rules. See [rule types](#).
- `primary_text` - A string with primary text of rule notification when condition is `true`.
- `secondary_text` - An optional string with secondary text of rule notification when condition is `false`. The availability of this parameter depends on rule type. Not every rule has the `secondary_text`.
- `param` - An optional integer. A common parameter that responsible for integer conditions. The availability of this parameter depends on rule type. See [rule types](#).
- `alerts` - An object with destinations for notifications. Answers the question - who and how will receive notifications. Described in [rule object](#).
- `suspended` - A boolean which starts and stops the rule. `true` if the rule suspended.

- `schedule` - An optional object which configures the time - when the rule works. Described in [rule object](#).
- `extended_params` - An optional object. Specified for concrete rule type. See [rule types](#).

API request:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/rule/create' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "rule": {
"description": "", "type": "work_status_change", "primary_text":
"status changed", "alerts": {"push_enabled": true, "emails":
["example@gmail.com"], "emergency": false, "sms_phones":
["745494878945"], "phones": []}, "suspended": false, "name":
"Status changing", "trackers": [123456], "extended_params":
{"emergency": false, "zone_limit_inverted": false, "status_ids":
[319281,319282,319283]}, "schedule": [{"from": {"weekday": 1,
"time": "00:00:00"}, "to": {"weekday": 7,"time": "23:59:59"},
"type": "weekly"}], "zone_ids": []}}'
```

You will get ID of created rule in response.

```
{
  "success": true,
  "id": 123
}
```

## Bind/Unbind

When a rule created, [bind](#) devices to it. For example, a newly registered device must have the same rule. Unnecessary to create another rule. Bind this device to an already existing rule. Unbinding works similarly. When a rule is not necessary for some devices, [unbind](#) them without deleting rules.

Necessary parameters for both calls the same.

- `rule_id` - An ID of a rule. You can get IDs using the [rule/list](#) call.
- `trackers` - An int array. List trackers' IDs. Trackers which do not exist, owned by other user or deleted ignored without errors.

API requests:

#### Bind

```
curl -X POST 'https://api.navixy.com/v2/tracker/rule/bind' \
-H 'Content-Type: application/json' \
```



```
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "rule_id": 123, "trackers": [265489]}'
```

## Unbind

```
curl -X POST 'https://api.navixy.com/v2/tracker/rule/unbind' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "rule_id": 123, "trackers": [265489]}'
```

## Update

If the rule must be updated, for example, one more phone number must be added for SMS notifications, you can use the [rule/update](#) call. It is much better than deleting an existing rule and creating a new one.

List of necessary parameters is the same as in [rule/create](#) call plus `id` parameter.

- `id` - An integer with ID of the updating rule. You can get IDs using the [rule/list](#) call.

API request:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/rule/update' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "rule": {"id": 123, "description": "", "type": "work_status_change", "primary_text": "status changed", "alerts": {"push_enabled": true, "emails": ["example@gmail.com"], "emergency": false, "sms_phones": ["745494878945"], "phones": []}, "suspended": false, "name": "Status changing", "trackers": [123456], "extended_params": {"emergency": false, "zone_limit_inverted": false, "status_ids": [319281, 319282, 319283]}, "schedule": [{"from": {"weekday": 1, "time": "00:00:00"}, "to": {"weekday": 7, "time": "23:59:59"}, "type": "weekly"}], "zone_ids": []}}'
```

## Suspend

To suspend the rule use the [rule/update](#) call and change only one parameter `suspended` to `true`. All other parameters should present in the call without changes.

API request:

## cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/rule/update' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "rule":  
{"id": 123, "description": "", "type": "work_status_change",  
"primary_text": "status changed", "alerts": {"push_enabled": true,  
"emails": ["example@gmail.com"], "emergency": false, "sms_phones":  
["745494878945"], "phones": []}, "suspended": true, "name":  
"Status changing", "trackers": [123456], "extended_params":  
{"emergency": false, "zone_limit_inverted": false, "status_ids":  
[319281,319282,319283]}, "schedule": [{"from": {"weekday": 1,  
"time": "00:00:00"}, "to": {"weekday": 7, "time": "23:59:59"},  
"type": "weekly"}], "zone_ids": []}}'
```

Last update: December 26, 2022





# How to work with notifications

Notifications important part of the tracking. A [created a rule](#) will track triggering of specified conditions and send events to emails and phones. It sends notifications to know that a condition triggered. Sometimes, necessary to store those notifications and history entries to use them in special reports, or they can be used for scripts build on APIs. Let's see how to work with them.

## Obtain a list of history entries

### All unread events of user

Here can be used the call [history/unread/list](#) to get all unread events.

The call contains only two optional parameters:

- `limit` - int with a maximum count of entries in response
- `from` - a string containing the start [date/time](#) for searching. Without this parameter you will get all unread entries for the last 30 days.

In our example we need to obtain no more than 100 entries for last month. If today is 26-01-2021 then API request will be:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/history/unread/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "limit": 100,  
    "from": "2020-12-26 00:00:00"}'
```

Response will contain a list of history entries with [information](#) that could be used for different purposes:

```
{  
  "success": true,  
  "list": [{  
    "id": 1,  
    "type": "tracker",  
    "is_read": false,  
    "message": "Alarm",  
    "time": "2020-12-31 00:00:00",  
    "event": "offline",  
    "tracker_id": 2,  
    "rule_id": 3,  
    "track_id": 4,  
  }]
```

```

    "location":{
        "lat": 50.0,
        "lng": 60.0,
        "precision": 50
    },
    "address": "address",
    "extra": {
        "task_id": null ,
        "parent_task_id": null,
        "counter_id": null,
        "service_task_id": null,
        "checkin_id": null,
        "place_ids": null,
        "last_known_location": false,
        "tracker_label": "Tracker label",
        "emergency": false,
        "employee_id": 4563
    }
}
}]
}

```

## Events for specific trackers and time period

Here can be used the [history/tracker/list](#) call to get all events for a specific tracker or trackers per necessary time period. Also, this call can return only specific event types with sorting by time if necessary.

The necessary parameters for the call:

- `trackers` - an int array. A list of [tracker IDs](#) belong to user for which events will be searched.
- `from` - a string containing the start [date/time](#) for searching.
- `to` - a string containing the end [date/time](#) of searching. Must be after `from` date.

Optional parameters that could be used to get more specific information:

- `events` - a string array with necessary event types. All other events will be ignored. Default: all. To get the list of events use [tracker/history/type](#) call.
- `limit` - integer with a maximum count of entries in result.
- `ascending` - a boolean where sort ascending by time when it is `true` and descending when `false`.

In our example we need to obtain no more than 100 entries for December for one device sorted descending by time. Also, necessary to know only when the device entered and exited the geofence. API request will be:

## cURL

```
curl -X POST 'https://api.navixy.com/v2/history/tracker/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "trackers":
[123985], "from": "2020-12-01 00:00:00", "to": "2020-12-31
23:59:59", "events": ["inzone", "outzone"], "limit": 100,
"ascending": false}'
```

Response will contain the [history entries](#) that match to our request.

## All events of a user per specific time period

To obtain a list of all tracker events in the user received between the specified "from" and "to" dates, use the [history/user/list](#) method. You can also filter the results to include only the necessary event types.

Here are the required parameters for the call:

- `from` - a string containing the start [date/time](#) for the search.
- `to` - a string containing the end [date/time](#) for the search. Must be after `from` date.

You can also use optional parameters to narrow down your search:

- `events` - a string array with necessary event types. All other events will be ignored. Default: all. To get a list of events, use the [tracker/history/type](#) call.
- `limit` - an integer specifying the maximum number of entries in the result.
- `ascending` - a boolean value that sorts the results in ascending order by time when set to `true` and descending when `false`.

For example, we want to get state field events for the last five minutes on all trackers of a user. In this case, we will use `to =CURRTIME` and `from =CURRTIME-5 minutes`.

Filtering must be by `state_field_control` events.

## cURL

```
curl -X POST 'https://api.navixy.com/v2/history/user/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "from":
"2023-06-13 18:42:10", "to": "2023-06-13 18:47:10", "events":
["state_field_control"], "limit": 100, "ascending": true}'
```

Response will contain the [history entries](#) that match to our request.

Last update: August 1, 2023







# How to get push notifications for your app

You can subscribe your app to work with push notifications. They allow you to get new events immediately and without requesting such an event with an API call. These notifications could be used by your program for triggering some actions with trackers, configs, tasks, sending them into your Telegram bot, etc.

Apps can be mobile or web-based. In each case, you need to subscribe it for push notifications differently. Let's examine each of these cases separately.

## Mobile apps

At this moment the platform supports Firebase Cloud Messaging

To get push notifications on mobile devices, you need get app's push token. It may be done in several steps:

1. Firebase projects support Google [service accounts](#), which you can use to call Firebase server APIs from the app server. To authenticate a service account and authorize it to access Firebase services, you must [generate a private key file in JSON format](#).
2. Contact our support team ([support@navixy.com](mailto:support@navixy.com)) with the generated private key, platform (Android/iOS) and your app's name.
3. We will provide you with the application for an API call to bind your app.
4. Get the push token of your app from Google Play Market or App Store.
5. Then use the [push\\_token/bind](#) API call from your app. Substitute the push token and received from our support team application ID into it.

## Web apps

In order for your web application to start receiving pushes, do the following steps:

When creating a subscription, you must specify the [applicationServerKey](#). It should be in BYTES, not as a base64 string.

Our public key in base64 is

BKPE9clw-\_CxWE-

WlqSkVpLnHT-7rE637udxtfGRUfXshjfCgatSNqNtRp5HjwEukACcdhIPMwPc9Ch7UsZXXY

function example:

```
return navigator.serviceWorker
    .register('/service-worker.js')
    .then(function (registration) {
        const subscribeOptions = {
            userVisibleOnly: true,
            applicationServerKey: urlBase64ToUint8Array(
                'BKPE9clw-_CxWE-
WlqSkVpLnHT-7rE637udxtfGRUfXshjfCgatSNqNtRp5HjwEukACcdhIPMwPxc9Ch7UsZX
            ),
        };

        return registration.pushManager.subscribe(subscribeOptions);
    })
    .then(function (pushSubscription) {
        console.log(
            'Received PushSubscription: ',
            JSON.stringify(pushSubscription),
        );
        return pushSubscription
    })
```

Take the endpoint and keys p256dh and auth from the pushSubscription object.

example:

```
{
  "endpoint": "https://some.pushservice.com/something-unique",
  "keys": {
    "p256dh":
"BIPUL12DLfytvTajnrYr2PRdAgXS3HGKiLqndGcJGabyhHheJYlNGCeXl1dn18gSJ1Wak
    "auth": "FPssNDTKnInHVndSTdbkFw=="
  }
}
```

Use the [push\\_token/bind](#) API call with parameters:

- application=w3c\_pushapi
- token=whole endpoint from pushSubscription, full URL like <https://fcm.googleapis.com/fcm/send/f6kicrBn7S0:APA91b.....>
- parameters=object with keys from pushSubscription {"p256dh": "...", "auth": "..."}

You will receive the notification in `event.data` in JSON format.

Last update: February 8, 2024





# How to create and assign tasks

Tasks are a handy feature for the Field Service. This tool allowed planning and monitoring the work of field workers. The number of possible goals for tasks is truly great. You can use them for service, delivery, transportation, merchandising, trade, and more. The employee will receive all the necessary information for the day and time, addresses, task description, contact numbers, etc.

To start work with tasks, they must be created. It will be a task with one point or several? This will determine whether we create a single task, or a route task.

## Single task

Creation of a new single task.

The list of necessary parameters is next:

`task` - a JSON object that contains all necessary information about the [task](#).  
`create_form` - a boolean parameter that responsible for a form creation for this task. If `true` then check additional `form_template_id` field in `task` object and [create form template](#) if it is not null. Default value is `false` for backward compatibility.

For example, we want to create the next task:

George will deliver new devices to the office on 16th of March, from 12 to 2 PM. His car has a tracker with ID 203190. Today may be some traffic jams that's why he may be late on one hour. Also, I know that he needs 30 minutes to get to the office, put in new devices, and fill in documents.

In this case, the `task` object will have the next parameters:

- `tracker_id` - to which tracker this task should be assigned.
- `location` - where the task should be.
- `label` - the name of a new task.
- `description` - a note about the task.
- `from` and `to` - when the task should be completed.
- `max_delay` - the employee may be late with the execution for a maximum of this time in minutes.
- `min_stay_duration` - the task will not be considered completed if the employee spends less than this time in the task's zone.

API request:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/create' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task":
{"tracker_id": 203190, "location": {"lat": 34.178868, "lng":
-118.599672, "radius": 150}, "label": "New devices to office",
"description": "16 new devices", "from": "2021-03-16 12:00:00",
"to": "2021-03-16 14:00:00", "max_delay": 60, "min_stay_duration":
30}, "create_form": false}'
```

The response will contain ID of a new task.

```
{
  "success": true,
  "id": 111
}
```

## Route task

Creation of a new route task.

The list of necessary parameters is next:

- `route` - JSON object containing all necessary information about the `route` without *IGNORED* fields.
- `checkpoint` - array of `checkpoint objects` without *IGNORED* fields.
- `create_form` - boolean. If `true` then check additional `form_template_id` field in every **checkpoint** object and create form if it is not null. Default value is `false` for backward compatibility.

For example, we need to create the next route:

John needs to deliver our products to three customers on 18th of March, from 10 AM to 4 PM. His car has a tracker with ID 669673. He can't get late because our customers will wait for production at the exact time and if he is late - the checkpoint will be considered failed. This is how we know about the quality of delivery. Also, I know that he needs a minimum of 10 minutes to hand over the goods to the client and fill out the documents.

In this case, every `checkpoint` object will have the next parameters:

- `tracker_id` - an ID of the tracker to which checkpoint should be assigned.
- `location` - location associated with this checkpoint. cannot be null.

- `label` - the name of the checkpoint.
- `description` - a note about the checkpoint.
- `from` and `to` - the time when this checkpoint should be completed.
- `external_id` - this is a delivery code. It is necessary for a checkpoint because I have the plugin "Courier on the map". Customers can specify this ID to the plugin and see - where the driver at the moment.
- `max_delay` - the employee may be late with the execution for a maximum of this time in minutes. In our case, it is 0 minutes.
- `min_stay_duration` - the task will not be considered completed if the employee spends less than this time in the task's zone.
- `tags` - for every client, I created a tag. This allows me to keep statistics and facilitate the search for delivery to this particular client.
- `form_template_id` - when employees hand over the objects, they fill this form that contains information about the quality of delivery, photos of delivered products, bill, and customer's signature.

The route object will have its own parameters too:

- `tracker_id` - an ID of the tracker to which a route should be assigned.
- `label` - route name.
- `description` - additional information about the whole route.
- `from` and `to` - the time when this route should be completed.

[API request:](#)



## cURL

```
curl -X POST 'https://api.navixy.com/v2/task/route/create' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "route":
{"tracker_id": 669673, "label": "Products delivery",
"description": "12 trackers of model 1 and 37 trackers of model
2", "from": "2020-03-18 10:00:00", "to": "2020-03-18 16:00:00"},
"checkpoints": [{"tracker_id":669673,"location":{"lat":
34.178868,"lng":-118.599672,"radius":
100},"label":"Company1","description":"5 trackers of model 1 and
15 trackers of model 2","from":"2021-03-18
10:00:00","to":"2021-03-18
12:00:00","external_id":"10100","max_delay":0,"min_stay_duration":
10,"tags":[1,4],"form_template_id":132985},{ "tracker_id":
669673,"location":{"lat":33.492830,"lng":-112.177673,"radius":
100},"label":"Company2","description":"4 trackers of model 1 and
12 trackers of model 2","from":"2021-03-18
10:00:00","to":"2021-03-18
14:00:00","external_id":"10101","max_delay":0,"min_stay_duration":
10,"tags":[2,4],"form_template_id":132985},{ "tracker_id":
669673,"location":{"lat":39.801066,"lng":-105.028685,"radius":
100},"label":"Company3","description":"3 trackers of model 1 and
10 trackers of model 2","from":"2021-03-18
10:00:00","to":"2021-03-18
16:00:00","external_id":"10102","max_delay":0,"min_stay_duration":
10,"tags":[3,4],"form_template_id":132985}], "create_form": false}'
```

The response will be:

```
{
  "success": true,
  "result": {
    "id": 7115375,
    "user_id": 184541,
    "tracker_id": 669673,
    "label": "Products delivery",
    "description": "12 trackers of model 1 and 37 trackers of
model 2",
    "from": "2021-03-18 10:00:00",
    "to": "2021-03-18 16:00:00",
    "creation_date": "2021-03-17 14:45:49",
    "status": "assigned",
    "status_change_date": "2021-03-17 14:45:49",
    "origin": "manual",
    "checkpoint_ids": [
      7115376,
      7115377,
      7115378
    ],
    "external_id": null,
    "type": "route"
  }
}
```

## Route optimization

If we need to get the optimized route that will minimize transit time and costs, it may be beneficial to reorder route checkpoints before route creation. Our platform provides a way to perform such optimization. Provide data required to optimize, and the algorithm returns order in which points should be visited. Specify checkpoint objects in this order when you will create the route.

Necessary parameters:

- `start_point` - JSON object with the point and time from that our driver will start the move.
- `route_points` - an array of JSON objects with points and time that driver should visit.

API request:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/route/points/optimize' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"start_point": {"lat": 34.178868, "lng": -118.599672, "departure":
"2021-03-18 10:00:00"}, "route_points": [{"location":{"lng":
33.492830,"lat":-112.177673},"from":"2021-03-18
10:00:00","to":"2021-03-18 12:00:00"}, {"location":{"lng":
39.801066,"lat":-105.028685},"from":"2021-03-18
10:00:00","to":"2021-03-18 14:00:00"}, {"location":{"lat":
35.365948,"lng":-108.112104},"from":"2021-03-18
10:00:00","to":"2021-03-18 16:00:00"}]}'
```

Response will consist the order in that checkpoint objects should be specified in `checkpoints` parameter of route creation:

```
{
  "success": true,
  "result": [
    0,
    1,
    2
  ]
}
```

## Association with address

To associate the task or checkpoint with an address it should be specified in the location object. In this case, location object in the create action will have an additional

field - address. To get an address when you have location use the [geocoder/search\\_location](#) call.

Last update: January 15, 2024





## Forms usage

Task forms can be used in many areas of business. Delivery, sales, inspections, customer surveys, field reports. The results can be used to improve certain areas of your business, concentration in an area, or market analysis. The forms can be filled out by employees using the X-GPS tracker application. Employees can fill out forms when completing tasks or sending check-ins.

## Forms creation

To make it possible for employees to fill out forms, and for users to assign these forms to tasks, the form templates must be [created](#). Let's create a form for different needs. For example, we have a delivery service. Customers order certain products, such as trackers, which are delivered and, if necessary, installed by our staff. Find form fields that will be used [here](#).

We expect to see results on every task that's why we create a form that should be submitted only in a zone of a task. It is necessary to avoid task completion after our employee visited and spent some time in a task point. All fields we create are required to submit except one.

- The first field will be a text field to get customer's and company's name.
- The second text field will contain information about what was delivered to our customer.
- The third we will use is a checkbox field where additional provided services will be checked. Minimum checked fields 1, maximum 3 and this field will be not required because if our customer will not order additional options our employee will be unavailable to send this form and complete a task. \*Also, we add a signature field. It will help us to confirm that the customer received the order.

API request:

## cURL

```
curl -X POST 'https://api.navixy.com/v2/form/template/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "template":
{"label": "Trackers delivery", "description": "Employee, fill this
form with every delivery", "fields": [{"id": "Text-1", "type":
"text", "label": "Customer's name", "required": true,
"description": "Specify here customer's and company's name",
"max_length": 1000, "min_length": 1}, {"id": "Text-2", "type":
"text", "label": "Delivered", "required": true, "description": "Specif
here all delivered models and its amount", "max_length": 1000,
"min_length": 1}, {"id": "Checkbox", "type": "checkbox_group",
"label": "Additional options", "description": "Specify here all
provided additional options", "group": [{"label": "Presentation
and training"}, {"label": "Additional configuration"}, {"label":
"Installation"}], "max_checked": 3, "min_checked": 1, "required":
false}, {"id": "Signature", "type": "signature", "label":
"Customer's signature", "description": "Let a customer add his
signature about receiving the order", "required": true}],
"submit_in_zone": true, "default": true}}'
```

The platform will respond with:

```
{
  "success": true,
  "id": 111
}
```

## Form filling

Forms can be filled in two ways:

- Check-in. An employee sends information about his location with a filled form. Every employee of a user can choose a created form to send it with check-ins. Additional assigning is not necessary.
- Task completion. An employee performs a task and sends a form as progress report. A form should be assigned to a task before an employee will have a possibility to fill it in a task completion zone.

## Form assigning

A form can be assigned to an existing task with [task update call](#) or can be used in the process of [task creation](#).



`create_form` parameter should be `false` to add an already created form.

## Obtaining information from submitted forms

We can get submitted forms to analyze all information our employees specified in several ways.

### Specific forms as they sent in tasks

- Obtain a [list of templates](#) to get a `template_id` of a form we are interested in.
- Obtain the [list of tasks](#) with the necessary form `template_id` and per specific time period.
- With this `task_id` we can request [downloading](#) or [reading](#) the necessary form.

Or

- Get a [list of tasks](#) to find a specific task and obtain a `form_id` from it.
- Use this `form_id` to [read](#) and [download](#) forms.

### Specific forms as they sent in check-ins

- Obtain a list of [check-ins](#) to get `form_id` we are interested in.
- Use this `form_id` to [read](#) and [download](#) forms.

### To get counted information in the report format

- [Generate](#) a form completion report with [plugin\\_id 70](#).
- [Download](#) this report.

Last update: August 1, 2023







# How to work with statuses

Statuses are used to track current employee activity (in fact, of tracking devices owned by employees). The simplest example is "busy" | "not busy". This is a status listing consisting of two elements (working statuses). Different trackers can be assigned different status lists.

## Create

We need to create a working status list that we will assign to the device. Based on the working statuses that are created for the sheet - we will have a choice - what working status can be assigned to the tracker.

To create the working status list we need only one parameter: \* `listing` - [status\\_listing](#) object without "id" and "entries" fields.

For example, we will create a working status list for the delivery service to allow drivers and supervisors to change the working status. Drivers can change their working status using the X-GPS app. Supervisors can change working status using the UI.

API request:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/listing/create' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "listing":
{"label": "Delivery_service", "employee_controlled": true,
"supervisor_controlled": true}'
```

The response will contain ID of a new working status list:

```
{
  "success": true,
  "id": 1111
}
```

When we created a working status list, we need to fill it with working statuses, and we should use one request per one working status.

For example, we have 4 working statuses: "Free", "Break", "Pick up the goods from storage", "Deliver goods".

API request:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/create' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "listing_id": 1111, "status": {"label": "Free", "color": "E57373"}}'
```

The response will contain ID of a new working status:

```
{
  "success": true,
  "id": 1
}
```

## Assign

To assign the working status list to some devices, we use the `tracker_id` and `listing_id`.

For example, we have 10 drivers. We should create 10 requests for assigning the working status list to them.

API request:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/listing/tracker/assign' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 615487, "listing_id": 111}'
```

The platform will notify you about success in reply.

After that, our drivers and supervisors will have access to change working statuses. Where they could be used: We can create a script that will assign a new task to a driver with the working status "Free". After assigning the task, this script will change the working status itself to "Pick up the goods from storage". Or when the task was completed, a script changes the working status to "Free" and gets fields from the ERP system, parses them to create a new task assigns this task to a driver and so on.

Last update: December 26, 2022



# How to send commands to device via GPRS

Many devices can be reconfigured using GPRS commands. If we have commands in a protocol-dependent manner, and the device is online - we are able to change the device's configuration, or create an app for sending commands to devices from the UI by users. Also, that app can allow users to customize commands according to their needs.

Another way of usage is to bind commands sending to the rules, status changing, some parameters triggering. The count of possible variations is great.

## Sending of a command

To send a command to a device we need only the next two parameters:

`tracker_id` - ID of the device to which we want to send the command. `command` - Text or hexadecimal representation of the command in a protocol-dependent manner.

For example, we have a Teltonika FMB140 device with ID on the platform 231402. And I want to send a command to reconfigure its IP address to a new one 52.57.1.136.

According to the protocol the command should be the next:

```
setparam 2004:52.57.1.136
```

API request:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/raw_command/send' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 231402, "command": "setparam 2004:52.57.1.136"}'
```

The platform will notify you about success in reply.

Last update: November 10, 2021



# How to obtain report's information

Reports consider information that can be used to manage your fleet successfully. Sometimes it is necessary to get a report's information that can be used in programs or specific reports in needs for business. For example, necessary information about trips + fuel consumption, drains and refills. Follow the next steps, to obtain report's information.

## Generate report

To receive data for processing, it must be generated. This can be done using a call [report/tracker/generate](#).

Parameters that necessary for this call:

- `from` - A string containing [date/time](#). Data in a report will be from that moment.
- `to` - A string containing [date/time](#). Specified date must be after `from` date. Data in a report will be till specified moment.
- `title` - Report title. Default title will be used if null.
- `trackers` - List of [trackers' IDs](#) to be included in report (if report is by trackers).
- `employees` - List of [employees' IDs](#) to be included in report (if report is by employees).
- `time_filter` - An object which contains everyday time and weekday limits for processed data, e.g. `{"to": "18:00", "from": "12:00", "weekdays": [1, 2, 3, 4, 5]}`.
- `plugin` - A plugin object. The list of all [report plugins](#).

API request:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/generate' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "title": "Trip report", "trackers": [669673], "from": "2020-10-05 00:00:00", "to": "2020-10-06 23:59:59", "time_filter": {"from": "00:00:00", "to": "23:59:59", "weekdays": [1, 2, 3, 4, 5, 6, 7]}, "plugin": {"hide_empty_tabs": true, "plugin_id": 4, "show_seconds": false, "include_summary_sheet_only": false, "split": true, "show_idle_duration": false, "show_coordinates": false, "filter": true, "group_by_driver": false}}'
```



It will respond with generated report\_id.

```
{
  "success": true,
  "id": 222
}
```

## Retrieve report

To obtain all generated analytic data from the report in JSON format use [report/tracker/retrieve](#).

Use the report\_id from the previous call response.

API request:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/retrieve' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "report_id": 1234567}'
```

### HTTP GET

```
https://api.navixy.com/v2/report/tracker/retrieve?
hash=a6aa75587e5c59c32d347da438505fc3&report_id=1234567
```

You will get the report in a JSON format:

## Response

```
{
  "success": true,
  "report": {
    "created": "2020-10-06 16:01:46",
    "time_filter": {
      "from": "00:00:00",
      "to": "23:59:59",
      "weekdays": [
        1,
        2,
        3,
        4,
        5,
        6,
        7
      ]
    },
    "title": "Trip report",
    "id": 5602232,
    "sheets": [
      {
        "header": "Samantha (Ford Focus)",
        "sections": [
          {
            "data": [
              {
                "rows": [
                  {
                    "to": {
                      "v": "02:39 - Serpukhov,
Moscow Oblast, Russia, 142253",
                      "raw": 1601941188000.0,
                      "type": "value",
                      "location": {
                        "lat": 54.9218516,
                        "lng": 37.335545
                      }
                    },
                    "from": {
                      "v": "00:47 - Selyatino, Naro-
Fominskii gor. okrug, Moscow Oblast, Russia, 143370",
                      "raw": 1601934439000.0,
                      "type": "value",
                      "location": {
                        "lat": 55.5311083,
                        "lng": 36.96743
                      }
                    },
                    "time": {
                      "v": "01:52",
                      "raw": 6749.0,
                      "type": "value"
                    },
                    "length": {
                      "v": "106.29",
                      "raw": 106.29,
                      "type": "value"
                    }
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
  }
}
```

```

    },
    "avg_speed": {
      "v": "57",
      "raw": 57.0,
      "type": "value"
    },
    "max_speed": {
      "v": "94",
      "raw": 94.0,
      "type": "value"
    }
  },
  {
    "to": {
      "v": "05:10 - Selyatino, Naro-
Fominskii gor. okrug, Moscow Oblast, Russia, 143370",
      "raw": 1601950218000.0,
      "type": "value",
      "location": {
        "lat": 55.5308216,
        "lng": 36.967315
      }
    },
    "from": {
      "v": "03:11 - Serpukhov,
Moscow Oblast, Russia, 142253",
      "raw": 1601943083000.0,
      "type": "value",
      "location": {
        "lat": 54.9218116,
        "lng": 37.3354833
      }
    },
    "time": {
      "v": "01:58",
      "raw": 7135.0,
      "type": "value"
    },
    "length": {
      "v": "106.97",
      "raw": 106.97,
      "type": "value"
    },
    "avg_speed": {
      "v": "54",
      "raw": 54.0,
      "type": "value"
    },
    "max_speed": {
      "v": "94",
      "raw": 94.0,
      "type": "value"
    }
  },
  {
    "to": {
      "v": "07:54 - Khievskii
pereulok, 10, TNKh, Rassudovo, Troitsky Administrative Okrug,
Moscow, Russia, 143340",
      "raw": 1601960075000.0,
      "type": "value",

```

```

        "location": {
            "lat": 55.4666366,
            "lng": 36.9216966
        }
    },
    "from": {
        "v": "07:38 - Selyatino, Naro-
Fominskii gor. okrug, Moscow Oblast, Russia, 143370",
        "raw": 1601959081000.0,
        "type": "value",
        "location": {
            "lat": 55.53122,
            "lng": 36.9672916
        }
    },
    "time": {
        "v": "00:16",
        "raw": 994.0,
        "type": "value"
    },
    "length": {
        "v": "10.03",
        "raw": 10.03,
        "type": "value"
    },
    "avg_speed": {
        "v": "36",
        "raw": 36.0,
        "type": "value"
    },
    "max_speed": {
        "v": "85",
        "raw": 85.0,
        "type": "value"
    }
},
{
    "to": {
        "v": "09:36 - Serpukhov,
Moscow Oblast, Russia, 142253",
        "raw": 1601966165000.0,
        "type": "value",
        "location": {
            "lat": 54.926835,
            "lng": 37.3341066
        }
    },
    "from": {
        "v": "07:58 - Khievskii
pereulok, 10, TNKh, Rassudovo, Troitsky Administrative Okrug,
Moscow, Russia, 143340",
        "raw": 1601960315000.0,
        "type": "value",
        "location": {
            "lat": 55.46661,
            "lng": 36.9216516
        }
    },
    "time": {
        "v": "01:37",
        "raw": 5850.0,

```

```

        "type": "value"
    },
    "length": {
        "v": "95.31",
        "raw": 95.31,
        "type": "value"
    },
    "avg_speed": {
        "v": "59",
        "raw": 59.0,
        "type": "value"
    },
    "max_speed": {
        "v": "91",
        "raw": 91.0,
        "type": "value"
    }
},
{
    "to": {
        "v": "09:53 - Serpukhov,
Moscow Oblast, Russia, 142253",
        "raw": 1601967190000.0,
        "type": "value",
        "location": {
            "lat": 54.921935,
            "lng": 37.33551
        }
    },
    "from": {
        "v": "09:43 - Serpukhov,
Moscow Oblast, Russia, 142253",
        "raw": 1601966585000.0,
        "type": "value",
        "location": {
            "lat": 54.9264033,
            "lng": 37.3336633
        }
    },
    "time": {
        "v": "00:10",
        "raw": 605.0,
        "type": "value"
    },
    "length": {
        "v": "0.95",
        "raw": 0.95,
        "type": "value"
    },
    "avg_speed": {
        "v": "6",
        "raw": 6.0,
        "type": "value"
    },
    "max_speed": {
        "v": "13",
        "raw": 13.0,
        "type": "value"
    }
},
{

```

```

        "to": {
          "v": "12:36 - Selyatino, Naro-
Fominskii gor. okrug, Moscow Oblast, Russia, 143370",
          "raw": 1601977017000.0,
          "type": "value",
          "location": {
            "lat": 55.5309666,
            "lng": 36.9674183
          }
        },
        "from": {
          "v": "10:27 - Serpukhov,
Moscow Oblast, Russia, 142253",
          "raw": 1601969226000.0,
          "type": "value",
          "location": {
            "lat": 54.9219933,
            "lng": 37.335495
          }
        },
        "time": {
          "v": "02:09",
          "raw": 7791.0,
          "type": "value"
        },
        "length": {
          "v": "108.48",
          "raw": 108.48,
          "type": "value"
        },
        "avg_speed": {
          "v": "50",
          "raw": 50.0,
          "type": "value"
        },
        "max_speed": {
          "v": "89",
          "raw": 89.0,
          "type": "value"
        }
      },
      {
        "to": {
          "v": "16:01 - KhP \"Lesnoe
ozero\", Dernopol'e, gor. okrug Serpukhov, Moscow Oblast, Russia,
142279",
          "raw": 1601989300000.0,
          "type": "value",
          "location": {
            "lat": 54.9875133,
            "lng": 37.3093183
          }
        },
        "from": {
          "v": "13:34 - Selyatino, Naro-
Fominskii gor. okrug, Moscow Oblast, Russia, 143370",
          "raw": 1601980444000.0,
          "type": "value",
          "location": {
            "lat": 55.5309966,
            "lng": 36.96738
          }
        }
      }
    ]
  }
}

```

```

        },
        "time": {
            "v": "02:27",
            "raw": 8856.0,
            "type": "value"
        },
        "length": {
            "v": "95.79",
            "raw": 95.79,
            "type": "value"
        },
        "avg_speed": {
            "v": "39",
            "raw": 39.0,
            "type": "value"
        },
        "max_speed": {
            "v": "88",
            "raw": 88.0,
            "type": "value"
        }
    },
    ],
    "total": {
        "text": "In total:",
        "time": {
            "v": "10:33",
            "raw": 37980.0,
            "type": "value"
        },
        "length": {
            "v": "523.8",
            "raw": 523.8,
            "type": "value"
        },
        "avg_speed": {
            "v": "50",
            "raw": 50.0,
            "type": "value"
        },
        "max_speed": {
            "v": "94",
            "raw": 94.0,
            "type": "value"
        }
    },
    "header": "Oct 6, 2020 (Tue) : 7"
}
],
"type": "table",
"header": "Trips",
"columns": [
    {
        "align": "left",
        "field": "from",
        "title": "Movement start",
        "width": 4,
        "weight": 3,
        "highlight_min_max": false
    },

```

```

        {
            "align": "left",
            "field": "to",
            "title": "Movement end",
            "width": 4,
            "weight": 3,
            "highlight_min_max": false
        },
        {
            "align": "right",
            "field": "length",
            "title": "Total trips length,\nkm",
            "width": 1,
            "weight": 0,
            "highlight_min_max": false
        },
        {
            "align": "right",
            "field": "time",
            "title": "Travel time",
            "width": 1,
            "weight": 0,
            "highlight_min_max": false
        },
        {
            "align": "right",
            "field": "avg_speed",
            "title": "Average speed,\nkm/h",
            "width": 1,
            "weight": 0,
            "highlight_min_max": false
        },
        {
            "align": "right",
            "field": "max_speed",
            "title": "Max. speed,\nkm/h",
            "width": 1,
            "weight": 0,
            "highlight_min_max": false
        }
    ],
    "column_groups": []
},
{
    "rows": [
        {
            "v": "7",
            "raw": 7.0,
            "name": "Trips",
            "highlight": false
        },
        {
            "v": "523.8",
            "raw": 523.8,
            "name": "Total trips length, km",
            "highlight": false
        },
        {
            "v": "10:33",
            "raw": 633.0,
            "name": "Travel time",

```



```

        "highlight": false
      },
      {
        "v": "50",
        "raw": 50.0,
        "name": "Average speed, km/h",
        "highlight": false
      },
      {
        "v": "94",
        "raw": 94.0,
        "name": "Max. speed, km/h",
        "highlight": false
      },
      {
        "v": "515855",
        "raw": 515855.0,
        "name": "Odometer value *, km",
        "highlight": false
      }
    ],
    "type": "map_table",
    "header": "Summary"
  },
  {
    "text": "Odometer value at the end of the
selected period.",
    "type": "text",
    "style": "small_print"
  }
],
"entity_ids": [
  311852
],
"additional_field": ""
}
],
"from": "2020-10-06 00:00:00",
"to": "2020-10-06 23:59:59"
}

```

## Deleting reports

When the information has been received and processed, there is no need to leave the generated report. It can be removed. Use [report/tracker/delete](#).

Use the `report_id` from `generate` call response.

API request:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/delete' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "report_id":  
1234567}'
```

### HTTP GET

```
https://api.navixy.com/v2/report/tracker/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&report_id=1234567
```

Last update: December 26, 2022





# Tags usage

"Tag" is a label for convenient and fast search of the desired information. In our system tags help you find desired places, geofences, employees, tasks, trackers, and vehicles. You can create custom tags according to your needs. One object may have several tags. Tags are entities that could be assigned with objects.

## Case

I need to get easier searching by objects. For example, we have several places that are served by a certain team of employees, who in turn can only use the specified vehicles. We also want to easily find the tasks that we set for these teams. To do this, we will assign a specific tag to all these objects.

## Creation

The first step is to [create](#) this tag. Let's name it "team1".

API request:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tag/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tag": {"name": "team1", "color": "#00BFFF"}}'
```

### HTTP GET

```
https://api.navixy.com/v2/tag/create?
hash=a6aa75587e5c59c32d347da438505fc3&tag={"name": "team1",
"color": "#00BFFF"}
```

The platform will reply with the created tag id. We can find this tag using the [tag/list](#) call.

## Assigning

Now we need to assign this tag to our objects. We can do it using the update call for these objects. Also, we can assign this tag to a new object while creating. It can be done with adding the "tags" parameter to objects in these calls:

- [place](#) object - [update/create](#).
- [task](#) object - [update/create](#).
- [task\\_schedule](#) object - [update/create](#).
- [employee](#) object - [update/create](#).
- [vehicle](#) object - [update/create](#).
- [zone](#) object - [update/create](#).
- [tracker](#) object - There are no create and update calls for trackers. We should use [tags/set](#) call for them.

## Searching objects with tag

Tags assigned with objects. To find all these objects with information we should use [tag/search](#) call.

For example:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tag/search' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tag_ids": [179227]}'
```

### HTTP GET

```
https://api.navixy.com/v2/tag/search?
hash=22eac1c27af4be7b9d04da2ce1af111b&tag_ids=[179227]
```

The platform will provide us with objects with assigned tag in the response:

## Response

```
{
  "success": true,
  "result": {
    "place": [
      {
        "id": 1446571,
        "location": {
          "lat": 34.178868,
          "lng": -118.599672,
          "address": "21550 W Oxnard St, Woodland Hills,
CA 91367, USA",
          "radius": 100
        },
        "label": "New place",
        "description": "",
        "external_id": null,
        "tags": [
          179227
        ]
      }
    ],
    "task": [
      {
        "id": 8280866,
        "user_id": 184541,
        "tracker_id": 669673,
        "status": "assigned",
        "status_change_date": "2021-06-17 12:41:52",
        "tags": [
          179227
        ],
        "label": "New task ",
        "description": "",
        "external_id": null,
        "creation_date": "2021-06-17 12:41:37",
        "origin": "manual",
        "location": {
          "lat": 33.492830,
          "lng": -112.177673,
          "address": "3836-3820 N 55th Ave, Phoenix, AZ
85031, USA",
          "radius": 152
        },
        "from": "2021-06-17 00:00:00",
        "to": "2021-06-17 23:59:59",
        "arrival_date": null,
        "stay_duration": 0,
        "min_stay_duration": 0,
        "min_arrival_duration": 0,
        "max_delay": 0,
        "type": "task"
      }
    ],
    "employee": [
      {
        "id": 55693,
        "tracker_id": 669673,
```

```

        "first_name": "Artem",
        "middle_name": "",
        "last_name": "",
        "email": "",
        "phone": "",
        "driver_license_number": "",
        "driver_license_cats": "",
        "driver_license_issue_date": null,
        "driver_license_valid_till": null,
        "hardware_key": null,
        "department_id": null,
        "location": {
            "lat": 39.801066,
            "lng": -105.028685,
            "address": "5735 Hooker St, Denver, CO 80221,
United States",
            "radius": 150
        },
        "personnel_number": "1235341231",
        "tags": [
            179227
        ]
    }
}

```

Last update: January 15, 2024







## Vehicles and service works usage

Vehicle object is used to describe information about the vehicle. Its parameters include the VIN, chassis number, license plate, type, dimensions, load capacity, size and number of wheels, year of manufacture, fuel type used and consumption, as well as the insurance number and its validity period. You can also link a vehicle to one of the trackers.

For example, if the vehicle has an average fuel consumption, then we can see the average fuel consumption per trip in the device's trip report. Also, information about the mileage or engine hours can be used for timely maintenance.

For this purpose, service works are used, which can be applied to cost and risk management. For example, if there is a fleet of several cars, it is necessary to service each of them on time: to replace spare parts according to the spent resource, replace wheels according to mileage, and so on. Know the cost of all the work and notify in advance about the upcoming service work, so that you can order the necessary parts.

After all, service work not performed on time can lead to an undesirable outcome - the car will break down, and the order will not be delivered to the customer at the right hour. Not to mention the danger of losing the cargo completely or greatly reducing the service life of the vehicle.

### Vehicle creation

Let's create our first [vehicle object](#). For example, we have a cargo van Ford Transit.

We should specify all information we have about this vehicle in the [vehicle create](#) call. To link vehicle with the tracker - specify its ID in parameter `tracker_id`.

API request:

## cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "vehicle":
{"additional_info": "January 2021", "avatar_file_name": null,
"chassis_number": "", "color": "Blue", "frame_number": "",
"free_insurance_policy_number": "", "free_insurance_valid_till":
null, "fuel_cost": 4, "fuel_grade": "", "fuel_tank_volume": 80,
"fuel_type": "diesel", "garage_id": null, "gross_weight": null,
"icon_color": "1E96DC", "icon_id": null, "label": "Ford 53196",
"liability_insurance_policy_number": "54687965555er2152",
"liability_insurance_valid_till": "2022-01-12",
"manufacture_year": 2019, "max_speed": 100, "model": "Transit",
"norm_avg_fuel_consumption": 8.3, "passengers": 3,
"payload_height": 2550, "payload_length": 5531, "payload_weight":
1529, "payload_width": 2059, "reg_number": "A53196BC", "subtype":
"minivan", "tags": [], "tracker_id": 841400, "trailer": null,
"type": "truck", "tyre_size": "R15", "tyres_number": 4, "vin":
"XTA235KM35698512", "wheel_arrangement": "4x2"}}'
```

The platform will respond with:

```
{
  "success": true,
  "id": 96175
}
```

- `id` - It is ID of the created vehicle object.

## Service work creation

Now we have a vehicle object with an assigned device. The data from a device about mileage and engine hours can be used in our service works. For every part we should create a separate [service work](#).

For example, we bought new brakes and spark plugs and replaced the oil in the engine. Brakes better to change after 65k km, spark plugs after 40k km, and engine oil after 150 engine hours.

We should create three service works then. Also, I want to be notified before I must perform service work, that's why I will use advance notifications. We should use [service work create](#) API call.

API request for oil:

## cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/service_task/
create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "task":
{"vehicle_id": 96175, "comment": "Oil Ford Formula F 5W30",
"conditions": {"engine_hours": {"limit": 328, "notification_interval":
18}}, "cost": 28, "description": "Oil Change", "file_ids": [],
"notifications": {"sms_phones": ["79995699997"], emails:
["myemail@gmail.com"], push_enabled: true}, "repeat": false,
"unplanned": false}'
```

API request for brakes:

## cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/service_task/
create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "task":
{"vehicle_id": 96175, "comment": "ATE", "conditions": {"mileage":
{"limit": 78000, "notification_interval": 75000}}, "cost": 200,
"description": "Brakes Change", "file_ids": [], "notifications":
{"sms_phones": ["79995699997"], emails: ["myemail@gmail.com"],
push_enabled: true}, "repeat": false, "unplanned": false}'
```

The platform will respond with ID of created service work:

```
{
  "success": true,
  "id": 42401
}
```

Last update: July 19, 2022





## Driver journals usage

Driver Journal logs all the trips an employee made during a selected time period and groups them by their status: business, private or other. Users can choose to show either the summary of all trips with highlighted trip statuses or to view the required status only. The displayed data is already a finished document exportable as a PDF/Excel file or printed out as it is.

Driver Journal itemizes all the trips, providing exact mileage, accurate location, date and time, so clients can easily report business miles versus personal miles and deduct the exact amount of business-related expenses (e.g., by percentage of the actual expenses).

However, some situations are not that easy to identify. For those cases, use "other" trip status (specific comments can be given in notes). Such trips can be reviewed individually and using valid metrics be later converted to business or private as the case might be.

All the trip data will be safely and securely stored in one digital place. No need to keep the piles of paperwork. However, Revenue Services require to keep supporting receipts (for gas/fuel) to prove the actual amount of expenses.

For example, you need to distribute all trips of the device for the last day and distribute them by type in order to generate a bill for fuel payment. Also, they will contain important information about trips like driver, who was assigned to a tracker, start/end dates of the trip, start/end locations, length, start/end odometer values. You can operate with this information or use it in your CRM.

## Work with driver journals

### Getting all possible trips per period for journal

In order to generate a driver journal, we first need to get a list of possible trips.

API request:



## cURL

```
curl -X POST 'https://api.navixy.com/v2/driver/journal/proposal/
list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "from":
"2021-10-26 00:00:00", "to": "2021-10-26 23:59:59", "tracker_id":
311852}'
```

The platform will reply with the information about all trips per period:

```
{
  "success":true,
  "list":[{
    "tracker_id":311852,
    "employee_id":2183,
    "start_date":"2021-10-26 00:00:00",
    "end_date":"2021-10-26 01:39:22",
    "start_location":{
      "address":"Central'naya kol'cevaya avtomobil'naya doroga,
gor. okrug Istra, Moscow Oblast, Russia, 143540",
      "lat":55.8906183,
      "lng":36.944505
    },
    "end_location": {
      "address":"Klin, Moscow Oblast, Russia, 141609",
      "lat":56.356175,
      "lng":36.8077733
    },
    "length":70.83,
    "start_odometer":620741.0,
    "end_odometer":620812.0
  },
  {
    "tracker_id":311852,
    "employee_id":2183,
    "start_date":"2021-10-26 01:47:22",
    "end_date":"2021-10-26 03:30:58",
    "start_location":{
      "address":"Klin, Moscow Oblast, Russia, 141609",
      "lat":56.3562966,
      "lng":36.8079016
    },
    "end_location":{
      "address":"S/kh Lesnye ozera, Pyatnickoe shosse, Novaya,
Moscow Oblast, Russia, 141591",
      "lat":56.082615,
      "lng":36.9091333
    },
    "length":45.32,
    "start_odometer":620812.0,
    "end_odometer":620856.0
  },
  {
    "tracker_id":311852,
    "employee_id":2183,
```

```
    "start_date": "2021-10-26 03:37:58",
    "end_date": "2021-10-26 04:53:18",
    "start_location": {
      "address": "S/kh Lesnye ozera, Pyatnickoe shosse, Novaya,
Moscow Oblast, Russia, 141591",
      "lat": 56.082615,
      "lng": 36.9091333
    },
    "end_location": {
      "address": "Selyatino, Naro-Fominskii gor. okrug, Moscow
Oblast, Russia, 108806",
      "lat": 55.5309516,
      "lng": 36.967255
    },
    "length": 77.6,
    "start_odometer": 620856.0,
    "end_odometer": 620934.0
  }
}
```

## Entries creation

Once we have a list of all trips, we need to create a driver journal entries. In addition to the received objects in the previous call, we must specify the type of entry to be created, and we can add a comment to each of them.

API request:

## cURL

```
curl -X POST 'https://api.navixy.com/v2/driver/journal/entry/
create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "entries":
[{"tracker_id":311852, "employee_id":2183,
"start_date":"2021-10-26 00:00:00", "end_date":"2021-10-26
01:39:22", "start_location":{"address":"Central'naya kol'cevaya
avtomobil'naya doroga, gor. okrug Istra, Moscow Oblast, Russia,
143540", "lat":55.8906183, "lng":36.944505}, "end_location":
{"address":"Klin, Moscow Oblast, Russia, 141609", "lat":56.356175,
"lng":36.8077733}, "length":70.83, "start_odometer":620741.0,
"end_odometer":620812.0, "type": "work", "comment":
"order_ID=23415"}, {"tracker_id":311852, "employee_id":2183,
"start_date":"2021-10-26 01:47:22", "end_date":"2021-10-26 03:
30:58", "start_location":{"address":"Klin, Moscow Oblast, Russia,
141609", "lat":56.3562966, "lng":36.8079016}, "end_location":
{"address":"S/kh Lesnye ozera, Pyatnickoe shosse, Novaya, Moscow
Oblast, Russia, 141591", "lat":56.082615, "lng":36.9091333},
"length":45.32, "start_odometer":620812.0, "end_odometer":
620856.0, "type": "personal", "comment": "trip to a cafe"},
{"tracker_id":311852, "employee_id":2183, "start_date":"2021-10-26
03:37:58", "end_date":"2021-10-26 04:53:18", "start_location":
{"address":"S/kh Lesnye ozera, Pyatnickoe shosse, Novaya, Moscow
Oblast, Russia, 141591", "lat":56.082615, "lng":36.9091333},
"end_location":{"address":"Selyatino, Naro-Fominskii gor. okrug,
Moscow Oblast, Russia, 108806", "lat":55.5309516, "lng":
36.967255}, "length":77.6, "start_odometer":620856.0,
"end_odometer":620934.0, "type": "work", "comment":
"order_ID=31024"}]}'
```

The platform will confirm creation with:

```
{
  "success": true
}
```

## Driver journal obtaining

After all entries have been created, we can download the driver journal in the format we want by [download](#) API call.

To simply display the driver journal in a specialized application, for example, use the [list](#) request.

Last update: August 1, 2023





# Create check-ins via API



## Important

Check-ins are created using X-GPS Tracker. All the description below is necessary only for exceptional cases, such as creating your Mobile Tracker app.

Step 1. Create a form from a template with [checkin/form/create](#) API call. In the X-GPS Tracker, the form is created when the template is selected by a user.

Step 2. Create files for photos of check-in with [checkin/image/create](#) and upload photo data (see below). In the X-GPS Tracker, checkin photos are created as each photo is added.

Step 3. Create form files with [checkin/form/file](#) API call and upload their data (see below). In the X-GPS Tracker, form files are created when they are added when the form is filled out.

Step 4. Create a check-in itself with [checkin/create](#) API call, where all the data is attached. If the form includes optional fields that should be left empty for your check-in, simply refrain from adding these fields to the form submission object.

## File upload process

This is how files can be uploaded to the platform. If you have multiple files to upload, be sure to add a brief delay between uploading each one to ensure a smooth process.

Using the API calls [checkin/image/create](#) and [checkin/form/file](#) the app asks - may I provide you with the file? This file has size of X MB, file name is example.png, metadata of the image should be in EXIF format. This is already saved as metadata for your image in this format, you need to copy this info and add into request as a JSON object. The platform says, okay. I'm ready to receive this file from your app. It will provide you with the location and token if it is a local storage on your server:

```
{
  "success": true,
  "value": {
    "file_id": 111,
    "url": "http://bla.org/bla",
    "expires": "2020-02-03 03:04:00",
    "file_field_name": "example.png",
```

```

    "fields": {
      "token": "a43f43ed4340b86c808ac"
    }
  }
}

```

And for Cloud version of the platform Amazon S3 is used:

```

{
  "success": true,
  "value": {
    "file_id": 111,
    "url": "https://bla.s3.amazonaws.com/",
    "expires": "2020-02-03 03:04:00",
    "file_field_name": "file",
    "fields": {
      "policy": "<Base64-encoded policy string>",
      "key": "user/user1/${filename}",
      "success_action_status": "200",
      "x-amz-algorithm": "AWS4-HMAC-SHA256",
      "x-amz-credential": "AKIAIOSFODNN7EXAMPLE/20151229/us-east-1/
s3/aws4_request",
      "x-amz-date": "20151229T000000Z",
      "x-amz-signature": "<signature-value>",
      "x-amz-server-side-encryption": "AES256",
      "content-type": "image/png"
    }
  }
}

```

After it, the app must send the file. It should contain information from the platform's reply to the app's request.

Here's an example of upload you must make after receiving such response (assuming you uploading image named `actual_file_name.png`):

Internal storage example:

```

POST /bla HTTP/1.1 - part of URL after the host part.
Host: bla.org - only the host part.
Content-Length: 1325 - The content length in this case is the size
of data being sent, measured in bytes.
Origin: http://bla.org
... other headers ...
Content-Type: multipart/form-data; boundary=----
WebKitFormBoundaryePkpFF7tjBAqx29L - boundary is a marker used to
separate the different pieces of data sent in a multipart form. It
can be generated by the app randomly.

-----WebKitFormBoundaryePkpFF7tjBAqx29L
Content-Disposition: form-data; name="token"

a43f43ed4340b86c808ac - here add the token, received from the
platform

```

```

-----WebKitFormBoundaryePkpFF7tjBAqx29L
Content-Disposition: form-data; name="example.png";
filename="example.png" - the file name.
Content-Type: image/png

... contents of file goes here ... The contents of the file in
a .png or other format will depend on the image itself but
generally it will be composed of binary data that represents the
colors and shapes of the image.
-----WebKitFormBoundaryePkpFF7tjBAqx29L--

```

### Amazon S3 example:

```

POST / HTTP/1.1 - part of URL after the host part
Host: https://bla.s3.amazonaws.com - only the host part.
Content-Length: 1972 - The content length in this case is the size
of data being sent, measured in bytes.
Origin: https://bla.s3.amazonaws.com/
... other headers ...
Content-Type: multipart/form-data; boundary=WebAppBoundary -
boundary is a marker used to separate the different pieces of data
sent in a multipart form. It can be generated by the app randomly.

--WebAppBoundary
Content-Disposition: form-data; name="policy"
Content-Type: text/plain

eyJleHBpcmF0aW9uIjogIjIwMjMtMDMtMjdUMjE6MTU6MzYuMDc31dfQ==
--WebAppBoundary
Content-Disposition: form-data; name="key"
Content-Type: text/plain

nj9relv6m52qp01t0wv47wyk1ozd309g/${filename}
--WebAppBoundary
Content-Disposition: form-data; name="success_action_status"
Content-Type: text/plain

200
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-algorithm"
Content-Type: text/plain

AWS4-HMAC-SHA256
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-credential"
Content-Type: text/plain

AKIAIBQ6SRB65EVSSRMA/20230327/eu-central-1/s3/aws4_request
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-date"
Content-Type: text/plain

20230327T210036Z
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-signature"
Content-Type: text/plain

```



```
2df7efa0c0e0c5b97d0d9483acd77c9ec37360df921b019a4c4a93180a6136ad
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-server-side-encryption"
Content-Type: text/plain
```

```
AES256
--WebAppBoundary
Content-Disposition: form-data; name="file";
filename="actual_file_name.png"
Content-Type: image/png
```

```
... contents of file goes here ...
--WebAppBoundary--
```

Last update: August 1, 2023





# Tracking of stationary objects

Any object can be integrated into the Internet of Things. The platform allows tracking not only movable objects but also stationary ones, like heavy equipment, agricultural equipment, cargo, goods, or security equipment. Installing GPS devices on each of these objects can be very expensive. Instead, it's more cost-effective to install one device on a vehicle or site and track all others with cheaper BLE tags.

In this tutorial, we'll discuss how to organize tracking for stationary objects, which GPS devices and tags will help gather the necessary data, and how to set them up using truck trailers as an example. We'll also cover how to obtain information about trips and usage for subsequent service work and what API calls will provide information about the tags. Additionally, we'll share other use cases based on real situations.

Find this instruction including BLE sensors configuration example in our [Expert center](#).

## What you need to track stationary objects

Various devices are able to read data from BLE beacons: Galileosky, Quecklink, Ruptela, Teltonika, TopFlyTech. We will describe on example of Teltonika FMB920 model and BLE beacon Eye Sensor. To begin tracking stationary objects, you'll need the following: 1. A GPS device that can read BLE tags and is supported by the platform.

1. BLE tags that are compatible with the GPS device. It's worth noting that many BLE tags can transmit information about temperature and humidity, as well as their battery charge. This enhances the ability of these tags to track information, but for our purpose, we'll focus on stationary objects specifically.
2. Platform APIs that provide information about which GPS device a particular tag is near. To create custom solutions for your users using APIs, you'll need developers. Clients typically hire their own developers or contract third-party teams.

Now let's examine the procedure for implementing a real-world case study - tracking truck trailers for trip and usage information and subsequent service work.

## How to get information about BLE beacons near the GPS device

On the platform side, there's a BLE beacon data entry object:

```
{
  "tracker_id": 10181654,
  "hardware_id": "7cf9501df3d6924e423cabcd4c924ff",
  "rssi": -101,
  "get_time": "2023-04-17 17:14:42",
  "latitude": 50.3487321,
  "longitude": 7.58238,
  "ext_data": {
    "voltage": 3.075,
    "temperature": 24.0
  }
}
```

You can read information from it:

- `tracker_id` - int. An ID of the tracker (aka "object\_id").
- `hardware_id` - string. An ID of the beacon.
- `rssi` - int. RSSI stands for received signal strength indicator and represents the power of received signal on a device. According to it, you can understand how far away the beacon is from the tracker.
- `get_time` - date/time. When this data received.
- `latitude` - float. Latitude.
- `longitude` - float. Longitude.
- `ext_data` - object. Additional beacon data.

## API calls to get information about BLE tags

There are two API calls that allow you to get all the necessary information about BLE beacons:

### Historical data from BLE tags

The first call retrieves [historical data from devices](#). You can set the `from` and `to` parameters for obtaining data during a specific period about connected BLE beacons. Since we need the information from the BLE tags' point of view, i.e., the trailers, let's request the information using the `beacons` parameter.

Request example:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/beacon/data/read' \
-H 'Content-Type: application/json' \
-d '{"hash": "59be129c1855e34ea9eb272b1e26ef1d", "from": "2023-04-17 17:00:00", "to": "2023-04-17 18:00:00", "beacons": ["7cf9501df3d6924e423cabcd4c924ff"]}'
```

This will show which devices were in the vicinity of this BLE beacon during period.

```
{
  "list": [
    {
      "tracker_id": 10181654,
      "hardware_id": "7cf9501df3d6924e423cabcd4c924ff",
      "rssi": -101,
      "get_time": "2023-04-17 17:05:42",
      "latitude": 50.3487321,
      "longitude": 7.58238,
      "ext_data": {
        "voltage": 3.075,
        "temperature": 24.0
      }
    }, {
      "tracker_id": 10181654,
      "hardware_id": "7cf9501df3d6924e423cabcd4c924ff",
      "rssi": -101,
      "get_time": "2023-04-17 17:40:22",
      "latitude": 55.348890,
      "longitude": 6.59403,
      "ext_data": {
        "voltage": 3.075,
        "temperature": 24.0
      }
    }
  ],
  "success": true
}
```

## Last data from BLE tags

The second call retrieves information about [currently connected beacons](#) to a specific device. For example, if you want to know which trailer is currently near the device, use the following request:

Request example:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/beacon/data/last_values' \
-H 'Content-Type: application/json' \
-d '{"hash": "59be129c1855e34ea9eb272b1e26ef1d", "trackers":
[10181654], "skip_older_than_seconds": 1200}'
```

This will provide information that there's a trailer "7cf..." next to the device.

```
{
  "list": [
    {
      "tracker_id": 10181654,
      "hardware_id": "7cf9501df3d6924e423cabcd4c924ff",
      "rssi": -101,
      "get_time": "2023-04-17 17:40:22",
      "latitude": 55.348890,
      "longitude": 6.59403,
      "ext_data": {
        "voltage": 3.075,
        "temperature": 24.0
      }
    }
  ],
  "success": true
}
```

## How to obtain information on usage times and trip details

We've already gathered historical data using the first of the presented API calls, which showed on which devices the trailer was displayed at a specific time. To get information about the journeys and usage time of this trailer, we simply need to use one of the two API calls:

### Overall trip info

API call [track/list](#) to get trip information for the period. This will provide general information about the trips, such as where and when they started and ended, maximum speed, mileage, and more.

Request example:

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/beacon/data/last_values' \
-H 'Content-Type: application/json' \
-d '{"hash": "59be129c1855e34ea9eb272b1e26ef1d", "trackers":
[10181654], "skip_older_than_seconds": 1200}'
```

Response:

```
{
  "id": 11672,
  "start_date": "2023-04-17 17:05:42",
  "start_address": "10470, County Road, Town of Clarence, Erie
County, New York, United States, 14031",
  "max_speed": 62,
  "end_date": "2023-04-17 17:40:22",
  "end_address": "Fast Teddy's, 221, Main Street, City of
Tonawanda, New York, United States, 14150",
  "length": 18.91,
  "points": 59,
  "avg_speed": 49,
  "event_count": 3,
  "norm_fuel_consumed": 6.32,
  "type": "regular",
  "gsm_lbs": false
}
```

From this data, we can see that the trip lasted nearly 35 minutes (end\_date - start\_date), with an average speed of 49 km/h and a maximum speed of 62 km/h. The trip length was 18.91 km. This information allows us to determine how much to pay the driver for transporting the cargo, whether the contractual speed was exceeded, and other details. Additionally, the trip length can be used in the future to calculate the number of kilometers until the next maintenance of the trailer.

## Detailed trip info

If you want a detailed track record of the trailer where the beacon is installed for displaying it in a report, for example, you can use the track/read request. This will give us data on all the points received by the platform during the journey.

Request example:

### cURL

```
curl -X POST 'https://api.navixy.com/v2/track/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
10181654, "from": "2023-04-17 17:00:00", "to": "2023-04-17
18:00:00", "filter": true}'
```

Response:

```
{
  "success": true,
  "limit_exceeded": true,
  "list": [
    {
      "address": "10470, County Road, Town of Clarence, Erie
```



```
County, New York, United States, 14031",
  "satellites": 10,
  "mileage": 0,
  "heading": 173,
  "speed": 42,
  "get_time": "2023-04-17 17:05:42",
  "alt": 0,
  "lat": 43.0318683,
  "lng": -78.5985733
}
]
```

You can use these points together with your preferred maps API to display them on a map.

## Other examples of using BLE tags within Navixy API

Here are some other examples of how to use BLE tags with a short algorithm to get the necessary results need:

### Child seats

Child seats are mandatory for passengers traveling with children. If you or the user operates a passenger transportation service, knowing whether a child seat is available in a vehicle can help you quickly determine which drivers are suitable for certain passengers and avoid wasting time and fuel. You can also find out which driver currently has a child seat installed in their vehicle. Additionally, it's important to consider passengers with two or more children and identify cars equipped with more than one child seat.

To address this, you'll need to install a BLE beacon on each child seat. Next, let's say your transport booking app needs to request information from all drivers who have a child seat installed. To do this, use the `beacon/last_values` API call to gather information about which drivers can be assigned to a particular order.

You can also use the `RSSI` parameter to determine if the seat is located inside the vehicle or in the trunk. To accomplish this, you'll need to conduct a few tests. For example, if the `RSSI` value is lower in the passenger compartment than in the trunk, the seat is likely in the trunk. As a result, you can prioritize your search for vehicles – first, those with a child seat in the passenger compartment, and then those with a child seat in the trunk. This approach ensures that you efficiently match passengers with appropriate vehicles and drivers.

## Agricultural machinery

Suppose your client has agricultural machinery that can be connected to various equipment. How can you track which tractor is using a seeder and which has a plow? This information will help you understand the frequency and extent of tool usage, and also determine their current location. This way, workers can spend more time working in the field rather than searching for equipment. To achieve this, install devices on tractors and combines, as well as in tool storage areas. Place one BLE beacon on each tool in a secure spot where it is difficult to remove, preventing it from getting lost during work. Next, to determine how long the tools have been in use, query the `beacon/read` API call. The information from the response will be helpful, just like with the trailers in our detailed example. To determine the location of a specific tool, query `beacon/last_values` with a search for beacons to identify where and on which device the tool is installed. This approach ensures efficient tracking and utilization of your agricultural equipment, ultimately increasing productivity.

## Use on construction sites

Construction sites often have numerous tools and expensive equipment. While installing a beacon for tracking purposes is beneficial, another concern arises – how can you ensure that the equipment is tracked frequently, and that the GPS tracker doesn't run out of power? To monitor the usage and location of the equipment, BLE beacons can also come in handy.

The solution for construction sites can be similar to that of agricultural machinery – install devices on the machinery as well as on storage sites. This approach allows you to effectively track your valuable equipment, ensuring that it's being used efficiently and minimizing the risk of loss or misplacement. By keeping a close eye on your tools and machinery, you can optimize productivity at the construction site.

## Indoor tracking

You can effectively track items indoors using the platform and BLE tags. All you need to do is install GPS devices in different parts of the warehouse or building and tag the objects you want to track. Here are a few examples:

- Tracking employees in various areas of a warehouse or store: This allows you to know which area an employee is in or how many sales assistants are near the information desk. Having this information helps improve efficiency and ensures that staff members are where they need to be.

- Tracking goods or machinery in different areas of the warehouse: Knowing the location of goods or equipment saves time, as you don't have to search for them throughout the warehouse. This streamlines the retrieval process, making your operations more efficient.

## Tracking goods with BLE beacons

Utilizing BLE beacons for tracking can greatly benefit transport companies by allowing them to determine which truck is carrying a specific pallet of goods at any given moment. This method not only enables the tracking of goods' paths but also helps calculate transport costs more accurately.

By adopting this innovative approach, transport companies can enhance their operations, making them more efficient and precise. This ultimately leads to better service for clients and more streamlined business processes.

Last update: August 8, 2023





# Bill

Bill object description and API calls for work with user's bills.

## Bill object

```
{
  "order_id": 63602,
  "created": "2012-03-05 11:55:03",
  "sum": 150.0,
  "status": "created",
  "positions": ["The subscription fee for the services of
Account W3"],
  "link": "http://bill.navixy.com/xK1QEYK"
}
```

- `order_id` - int. Unique bill ID.
- `created` - [date/time](#). When the bill created.
- `sum` - float. A bill sum in default currency of the panel.
- `status` - [enum](#). Bill order status. Can be one of:
  - `created` – but not settled.
  - `settled`.
  - `canceled`.
- `positions` - string array. List of position names. Usually contains one element for a bill.
- `link` - string. URL to order.

## API actions

API path: `/bill`.

### create

Creates a new bill for the user.

**required sub-user rights:** `payment_create`.

## parameters

| name  | description                                  | type   |
|-------|--|--------|
| payer | Some payer description.                      | string |
| sum   | A bill sum in default currency of the panel. | double |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/bill/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "payer": "Jon Doe", "sum": 100.0}'
```

## response

```
{
  "success": true,
  "value": 6421
}
```

- `value` - int. Created bill ID.

## errors

- 222 – Plugin not found - when plugin **29** not available for user.

## list

Shows list of bills with their parameters in array.

**required sub-user rights:** `payment_create`.

## parameters

| name   | description   | type |
|--------|---|------|
| limit  | Optional. A maximum number of bills in list. Maximum and default is 10 000. | int  |
| offset | Optional. Get bills starting from <code>offset</code> . Default 0.          | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/bill/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/bill/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "count": 7,
  "bills": [{
    "order_id": 63602,
    "created": "2012-03-05 11:55:03",
    "sum": 150.0,
    "status": "created",
    "positions": ["The subscription fee for the services of
Account W3"],
    "link": "http://bill.navixy.com/xK1QEYK"
  }]
}
```

- `count` - int. Total number of bills.
- `bills` - array of objects. A list of [bill objects](#).

If bill created using [/bill/create](#) call then **positions** will contain exactly one element.



**For Standalone version base part of link may be changed by `billing.orders.baseUrl` config option.**

## errors

- 222 – Plugin not found - when plugin **29** not available for user.

Last update: December 26, 2022







# Payment system

Payment system settings object and API calls for working with payment systems and make payments.

## Payment system settings object

```
{
  "type": "rbkmoney",
  "url": "https:rbkmoney.com/acceptpurchase.aspx",
  "account": "John Doe",
  "currency": "EUR",
  "payment_code": "Navixy Demo",
  "subscription_code": "4671292",
  "methods": ["method1", "method2"],
  "prices": {
    "Loccate_default_pay_1": 0.99,
    "Loccate_default_pay_5": 4.99,
    "Loccate_default_pay_10": 9.99,
    "Loccate_default_pay_20": 19.99
  }
}
```

- `type` - string. Payment system type.
- `url` - string. URL to send payment info.
- `account` - optional string. Dealer account in payment system (eshopId for RBK).
- `currency` - string. 3-letter ISO 4217 currency code.
- `payment_code` - optional string. Code for payments.
- `subscription_code` - string. Subscription code. The same as "payment\_code" for 2Checkout (formerly Avangate) but for subscriptions.
- `methods` - optional string array. List of available payment methods (it may be empty).
- `prices` - optional object with prices. For type == `ios_inapp` only.

## API actions

API path: `/payment_system`.

## list

Returns list of payment systems available for specified user.

**required sub-user rights:** `payment_create`.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/payment_system/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/payment_system/list?  
hash=a6aa75587e5c59c32d347da438505fc3
```

### response

```
{  
  "success": true,  
  "list": [{  
    "type": "bill"  
  }]  
}
```

- `list` - array of objects. List of [payment system objects](#).

### errors

- 201 – Not found in the database.

## estimate/get

Returns the estimate of the monthly payment amount

**required sub-user rights:** `payment_create`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/payment_system/estimate/get' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/payment_system/estimate/get?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "value": 400.0
}
```

- `value` - float. Payment amount, rounded up to hundreds for rubles or to tens for other currencies.

Last update: July 19, 2022





# Subscription

API calls to interact with payment subscriptions

## API actions

API path: `/subscription`.

`/subscription/avangate/`

Working with [2Checkout](#) (formerly [Avangate](#)) subscriptions (renewals).

### cancel

Unsubscribe from auto-renewal by reference.

**required sub-user rights:** `payment_create`.

### parameters

| name      | description   | type   |
|-----------|---|--------|
| reference | Internal 2Checkout (formerly Avangate) subscription code.<br>Get it from <a href="#">list</a> call. | string |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/subscription/avangate/cancel' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "reference": "5EAD4B0B2F"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/subscription/avangate/cancel?
hash=a6aa75587e5c59c32d347da438505fc3&reference=5EAD4B0B2F
```

### response



```
{
  "success": true
}
```

## errors

- 215 – External service error.

## list

List active [2Checkout formerly Avangate](#) subscriptions (renewals).

**required sub-user rights:** `payment_create`.

## parameters

Only API key `hash`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/subscription/avangate/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/subscription/avangate/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [{
    "reference": "5EAD4B0B2F",
    "code": "4679109",
    "quantity": 123,
    "expiration_date": "2021-01-28 13:32:11"
  }]
}
```

- `reference` - string. Internal 2Checkout (formerly Avangate) subscription code. Pass it to `/subscription/avangate/cancel`.
- `code` - string. 2Checkout (formerly Avangate) product code.
- `quantity` - int. Count.
- `expiration_date` - [date/time](#). Next renew date/time.

## **errors**

- 215 – External service error.

Last update: December 26, 2022





# Transaction

Transaction object description and API call to get list of user's billing transactions for the specified period.

## Transaction object

```
{
  "description": "Recharge bonus balance during tracker
registration",
  "type": "bonus_charge",
  "subtype": "register",
  "timestamp": "2021-01-28 08:16:40",
  "user_id": 12203,
  "dealer_id": 5001,
  "tracker_id": 303126,
  "amount": -10.0000,
  "new_balance": 800.0000,
  "old_balance": 810.0000,
  "bonus_amount": 10.0000,
  "new_bonus": 10.0000,
  "old_bonus": 0.0000
}
```

- `description` - string. Transaction description.
- `type` - [enum](#). Type of transaction.
- `subtype` - [enum](#). Subtype of transaction.
- `timestamp` - [date/time](#). When transaction created.
- `user_id` - int. ID of a user which made a transaction.
- `dealer_id` - int. ID of a dealer.
- `tracker_id` - int. Tracker id. 0 if transaction not associated with tracker.
- `amount` - double. Amount of money in transaction, can be negative. e.g. -10.0000 means 10 money units removed from user's balance.
- `new_balance` - double. User's money balance after transaction.
- `old_balance` - double. User's money balance before transaction.
- `bonus_amount` - double. Amount of bonus used in transaction, can be negative. e.g. 10.0000 means 10 bonuses units added to user's bonus balance.
- `new_bonus` - double. User's bonus balance after transaction.
- `old_bonus` - double. User's bonus balance before transaction.

## API actions

API path: `/transaction`.

### list

Gets list of user's billing transactions for the specified period.

**required sub-user rights:** `payment_create`.

### parameters

| name  | description  | type                      |
|-------|--|---------------------------|
| from  | Start date/time for searching.                                     | <a href="#">date/time</a> |
| to    | End date/time for searching. Must be after <code>from</code> date. | <a href="#">date/time</a> |
| limit | Optional. Maximum number of returned transactions.                 | int                       |

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/transaction/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "from":  
    "2021-01-20 08:16:40", "to": "2021-01-28 08:16:40"}'
```

### response

```
{  
  "success": true,  
  "list": [{  
    "description": "Recharge bonus balance during tracker  
registration",  
    "type": "bonus_charge",  
    "subtype": "register",  
    "timestamp": "2021-01-28 08:16:40",  
    "user_id": 12203,  
    "dealer_id": 5001,  
    "tracker_id": 303126,  
    "amount": -10.0000,  
    "new_balance": 800.0000,  
    "old_balance": 810.0000,  
    "bonus_amount": 10.0000,  
    "new_bonus": 10.0000,  
  }]
```

```
    "old_bonus": 0.0000  
  }]  
}
```

- `list` - array of objects. List of [transactions objects](#).

#### errors

- 211 – Requested time span is too big - more than [report.maxTimeSpan](#).

Last update: December 26, 2022







# Tariff

Tariff object description and API call to get the list of device's tariffs available to user.

## Tariff object

```
{
  "id": 10,
  "name": "Business",
  "group_id": 2,
  "active": true,
  "type": "monthly",
  "price": 13.0,
  "early_change_price": 23.0,
  "device_limit": 1000,
  "has_reports": true,
  "paas_free": false,
  "store_period": "12m",
  "features": [
    "map_layers"
  ],
  "map_filter": {
    "exclusion": true,
    "values": []
  }
}
```

- `id` - int. Unique ID.
- `name` - string. Tariff's label.
- `group_id` - int. Group of tariffs. User can change the tariff only on the tariff in the same group.
- `active` - boolean. Tariff is active if `true`. User can change the tariff only on the active tariff.
- `type` - `enum`. Tariff type. Can be "monthly", "everyday", "activeday".
- `price` - double. Price per month for "monthly" and "everyday" tariff or price per "active" day for "activeday" tariff.
- `early_change_price` - double. Price of change tariff from current to another. With the last change in less than 30 days (**tariff.freeze.period** config option). When not passed or "null" user cannot change tariff frequently.
- `device_limit` - int. Maximum number of devices per account.
- `has_reports` - boolean. `true` if reports allowed, `false` otherwise.

- `paas_free` - boolean. `true` if this tariff is free for PaaS owner, `false` otherwise.
- `store_period` - string. Data storage period, e.g. "2h" (2 hours), "3d" (3 days), "5m" (5 months), "1y" (one year).
- `features` - string array. Available features for the user.
- `map_filter` - object with available maps for the user.
  - `exclusion` - boolean. If `true` maps from `values` will be not active, `false` - maps from values will be active.

## API actions

API path: `/tariff`.

### list

Gets list of device's tariffs available to user.

If user's dealer is **default dealer** or **paas** then listed tariffs of that dealer, else listed tariffs of parent dealer.

Listed only tariffs [available for user's legal type](#).

### parameters

Only API key `hash`.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tariff/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tariff/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

### response

```
{
  "success": true,
  "list": [{
    "id": 10,
    "name": "Business",
    "group_id": 2,
    "active": true,
    "type": "monthly",
```

```
    "price": 13.0,  
    "early_change_price": 23.0,  
    "device_limit": 1000,  
    "has_reports" : true,  
    "paas_free": false,  
    "store_period": "12m",  
    "features": [  
        "map_layers"  
    ],  
    "map_filter": {  
        "exclusion": true,  
        "values": []  
    }  
  }  
}]  
}
```

- `list` - array of objects. List of [tariff objects](#).

#### errors

- [General](#) types only.

Last update: December 26, 2022





# Tariff tracker

API calls on user's actions with tracker tariffs.

User of **dealer** can switch tracker from the tariff **t1** to tariff **t2** if:

1. Tracker belongs to user and isn't a **clone**.
2. Tracker's tariff last changed more than **tariff.freeze.period** (config option. default 30 days) ago.
3. **t1.tariff\_id** != **t2.tariff\_id**, i.e. the new tariff must be different from the current.
4. **t1.dealer\_id** = **t2.dealer\_id** = **dealer.effectiveDealerId**, i.e. current and new tariffs must belong to user's effective dealer.
5. **t2.active** = 1, i.e. new tariff is **active** (tariff's option "Allow users to switch to this tariff independently" in **panel** is set **on**).
6. **t1.grouping** = **t2.grouping**, i.e. user can change tariff only within one group of tariffs.
7. **t2.device** = **tracker**, i.e. new tariff must be for trackers.
8. The new tariff is [available to user's legal type](#).

User's **effective dealer** is

1. User's dealer if its **dealer\_id** = **defaultDealerId** (config option) or **dogovor\_type** = 'paas'.
2. Parent of user's dealer otherwise.

## API actions

API path: `/tariff/tracker/`.

### change

Changes tariff of tracker (with `tracker_id`) to new tariff (with `tariff_id`).

**required sub-user rights:** `admin` (available only to master users).

| name       | description  | type |
|------------|--|------|
| tracker_id | ID of the tracker (aka "object_id"). Tracker must belong to authorized user. | int  |
| tariff_id  | Id of the new tariff.  | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tariff/tracker/change' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 345215, "tariff_id": 12}'
```

### HTTP GET

```
https://api.navixy.com/v2/tariff/tracker/change?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=345215&tariff_id=12
```

## response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - if user doesn't have trackers with given `tracker_id`.
- 219 – Not allowed for clones of the device.
- 237 – Invalid tariff - if there are no tariff with specified `tariff_id` and belongs to user's **effective dealer**.
- 221 - Device limit exceeded – when new tariff device limit is less than count of trackers in account.
- 238 - Changing tariff is not allowed – user can't switch tracker to that tariff.
- 239 – New tariff doesn't exist.
- 240 - Not allowed changing tariff too frequently – tariff last changed less or equal to 30 days (**tariff.freeze.period** config option).

## list

List tariffs on which user can switch the passed tracker (even when tariff last changed less or equal than **tariff.freeze.period** time ago).



## parameters

| name       | description  | type |
|------------|--|------|
| tracker_id | ID of the tracker (aka "object_id"). Tracker must belong to authorized user. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tariff/tracker/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 345215}'
```

### HTTP GET

```
https://api.navixy.com/v2/tariff/tracker/list?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=345215
```

## response

```
{
  "success": true,
  "list": [{
    "id": 10,
    "name": "Business",
    "group_id": 2,
    "active": true,
    "type": "monthly",
    "price": 13.0,
    "early_change_price": 23.0,
    "device_limit": 1000,
    "has_reports": true,
    "paas_free": false,
    "store_period": "12m",
    "features": [
      "map_layers"
    ],
    "map_filter": {
      "exclusion": true,
      "values": []
    }
  }],
  "days_to_next_change": 11
}
```

- `list` - array of objects. List of [tariff objects](#).
- `days_to_next_change` - int. Days to the next free change, or 0 if free change available.

## errors

- [General](#) types only.

Last update: December 26, 2022






# API Keys

The API key is the main thing that is needed for the integration. This is the same as the hash of the user's session gotten by the [auth call](#), only with an infinite lifetime.

Unlike the user's session:

- the API key will not be deleted if the user logs out or changes the password,
- you do not need to [renew](#) the key periodically,
- you do not transfer or store the username and password,
- you can delete the key at any time if there is a suspicion of compromise,
- you can create a separate key for each individual integration.
- if request rate limit is exceeded, regular users will not be blocked, because API keys have a separate counter.

 **You can get an API key in user's web interface. This is the recommended way instead of user session hash.**

In one user's account, you can have up to 20 API keys intended for different external integrations. To distinguish keys from each other, you should give them meaningful names.

## Security

Do not publish API keys anywhere. Having a key, you can perform almost any action in the user's account. Make API calls only over HTTPS because the key is transmitted in cleartext.

Find more details on API keys usage in our [instructions](#).

## API Key object

```
{  
  "hash": "c915157ac483e7319b0b257408bc04e1",  
  "create_date": "2021-10-29 12:00:36",  
}
```

```
"title": "Integration with My Super App"
}
```

- `hash` - string, 32 chars. Hash of an API key.
- `create_date` - `date/time`. Key creation date.
- `title` - string. Key title.

## Actions

API path: `/api/key`.

### create

Creates a new API key.

This call is available only to the master user and only with a standard session obtained using a login/password via [/user/auth](#).

#### parameters

| name  | description                 | type   | restrictions   |
|-------|-----------------------------|--------|--|
| hash  | Master user's session hash. | String | Not empty.   |
| title | New key title               | String | Not empty, only printable characters. Max length: 255. |

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/api/key/create' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "title": "My Super App"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/api/key/create?  
hash=a6aa75587e5c59c32d347da438505fc3&title=My+Super+App
```

#### response

```
{
  "success": true,
  "value": {
    "hash": "c915157ac483e7319b0b257408bc04e1",
    "create_date": "2021-10-29 12:00:36",
    "title": "My Super App"
  }
}
```

## errors

- 4 - User or API key not found or session ended. If the user session ( `hash` param) is invalid or a non-standard session is used (for example, another API key).
- 13 - Operation not permitted. If a call with subuser's session hash.
- 268 - Over quota. If 20 keys have already been created in the user's account.

## delete

Deletes API key.

This call is available only to the master user and only with a standard session obtained using a login/password via [/user/auth](#).

## parameters

| name | description                 | type   | restrictions |
|------|-----------------------------|--------|--------------|
| hash | Master user's session hash. | String | Not empty.   |
| key  | The API key to delete.      | String | Not empty.   |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/api/key/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "key": "5063e191d734e87e17987953c7a9a086"}'
```

### HTTP GET

```
https://api.navixy.com/v2/api/key/delete?
hash=a6aa75587e5c59c32d347da438505fc3&key=5063e191d734e87e17987953c7a9
```

## response

```
{
  "success": true
}
```

## errors

- 4 - User or API key not found or session ended. If the user session ( `hash` param) is invalid or a non-standard session is used (for example, another API key).
- 13 - Operation not permitted. If a call with subuser's session hash.
- 201 – Not found in the database - if there is no specified API key in account.

## list

Gets all of API keys for an account.

## parameters

| name | description                 | type   | restrictions |
|------|-----------------------------|--------|--------------|
| hash | Master user's session hash. | String | Not empty.   |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/api/key/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/api/key/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "list": [{
    "hash": "c915157ac483e7319b0b257408bc04e1",
    "create_date": "2021-10-29 12:00:36",
    "title": "My Super App"
  }, {
    "hash": "e3b7d1d727d21e064a190239b3403ee3",
    "create_date": "2021-11-19 16:06:03",
    "title": "AmoCRM integration"
  }],
}
```



```
"success": true  
}
```

#### errors

- 4 - User or API key not found or session ended. If the user session ( `hash` param) is invalid or a non-standard session is used (for example, another API key).
- 13 - Operation not permitted. If a call with subuser's session hash.

Last update: February 8, 2024





# Base

Contains API calls to health-check and send email.

## API actions

API path: `/base`.

### nothing

The report for health-check. It will do nothing.

#### parameters

Only API key `hash`.

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/base/nothing' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/base/nothing?  
hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{ "success": true }
```

#### errors

- [General](#) types only.

## send\_email

Sends email from the platform to any email address with specified title and text. Needs ROOT access level.

## parameters

| name         | description         | type   |
|--------------|---------------------|--------|
| from         | From email address. | string |
| to           | To email address.   | string |
| title        | Title of the email. | string |
| message      | Text of the email.  | string |
| service_id   | Service parameter.  | int    |
| service_pass | Service parameter.  | int    |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/base/send_email' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "from":  
"gps@navixy.com", "to" : "customer@email.com", "title": "test  
email", "message": "this email for test", "service_id": 1,  
"service_pass": 28}'
```

## response

```
{ "success": true }
```

## errors

- [General](#) types only.

Last update: December 26, 2022





# Data

API call to parse the spreadsheet data.

/data/spreadsheet/parse

Parse spreadsheet file (.xlsx, .xls, .csv) and store it in internal storage.

## parameters

| name          | description  | type        |
|---------------|--|-------------|
| file          | File to upload.  | file        |
| preview_count | Size of preview. Min=1, max=20.  | int         |
| parse_header  | Parse first row as header.   | boolean     |
| header_map    | If <code>parse_header</code> is <code>true</code> should contains map of matching column name to field identifier,<br><code>{"Label": "label", "Latitude": "lat"}</code> . | JSON object |

If `parse_header` is set to `true`, first row of the uploaded file will be treated as header corresponding to given `header_map`.

## response

```
{
  "file_id": "568539",
  "header": ["header1", "header2"],
  "preview": ["preview of file 1", "preview of file 2"]
}
```

- `file_id` - string. Unique file ID.
- `header` - optional string array. List of files' headers.
- `preview` - string array. First N rows of file.

## errors

- 234 – Invalid data format.



## /data/import/list

Returns the list of the user's import processes.

### parameters

| name  | description   | type         |
|-------|---|--------------|
| types | Optional. Types of the imported entities, e.g. ["vehicle", "employee"]. | string array |

### response

```
{
  "success" : true,
  "list" : [ {
    "id": <int>,
    "user_id": <int>,
    "created": <date>,
    "type": <string>, // vehicle | employee
    "params": {
      "headers": [<string>, <string>,...] // List of files' headers
    },
    "filename": <string>, // Name of preloaded TSV.
    "status": <string>, // created | in_progress | done | failed
    "status_change_date": <date>,
    "progress": {
      "imported": <int>,
      "failed": <int>,
      "percent": <int>, // approximate percentage of processed
      "processed_lines": <int>,
      "warnings": [{line:<int>, error: <string>}], // first 25
      "errors": [{line:<int>, error: <string>}], // first 25
    }
  }, ...]
}
```

### example

#### cURL

```
curl -X POST "https://api.navixy.com/v2/data/import/list" \
  -H "Content-Type: application/json" \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

## /data/import/read

Returns an import process with specified ID.

## parameters

| name       | description                   | type   |
|------------|-------------------------------|--------|
| process_id | Process ID                    | int    |
| type       | Type of the imported entities | string |

## response

```
{
  "success": true,
  "value": {
    "id": <int>,
    "user_id": <int>,
    "created": <date>,
    "type": <string>, // vehicle | employee
    "params": {
      "headers": [<string>, <string>, ...] // List of files' headers
    },
    "filename": <string>, // Name of preloaded TSV.
    "status": <string>, // created | in_progress | done | failed |
finished
    "status_change_date": <date>,
    "progress": {
      "imported": <int>,
      "failed": <int>,
      "percent": <int>, // approximate percentage of processed
      "processed_lines": <int>,
      "warnings": [{line:<int>, error: <string>}], // first 25
      "errors": [{line:<int>, error: <string>}], // first 25
    }
  }
}
```

## example

### cURL

```
curl -X POST "https://api.navixy.com/v2/data/import/read" \
-H "Content-Type: application/json" \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "type":
"employee", "process_id": 1}'
```

## errors

- 201 – Not found in database (if import is not found)

Last update: October 6, 2023





# Dealer

Contains API call to get dealer info and dealer-specific UI settings.

## API actions

API path: `/dealer`.

### get\_ui\_config

Gets dealer info and dealer-specific UI settings by a domain or hash.

It doesn't require authentication and available in **UNAUTHORIZED** access level.

#### parameters

| name   | description  | type   |
|--------|--|--------|
| domain | Dealer's monitoring interface domain, e.g. "panel.navixy.com".           | string |
| hash   | Used instead of a domain to identify a dealer if there is a user session | string |

Params `domain` and `hash` is not required both, but one of them must be specified. If `hash` is specified the `domain` shouldn't be used.

#### example

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/dealer/get_ui_config' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "domain": "panel.navixy.com"}'
```

#### response

```
{
  "success": true,
  "dealer": {
    "id": 5001,
    "ui_domain": "demo.navixy.com",
  }
}
```

```

    "company_url": "navixy.com"
  },
  "settings": {
    "domain" : "demo.navixy.com",
    "service_title": "Navixy Demo",
    "locale": "at_AT",
    "demo_login": "demo",
    "demo_password": "demo",
    "maps": ["roadmap", "osm"],
    "default_map": {
      "type": "roadmap",
      "location": {
        "lat": 57.0,
        "lng": 61.0
      },
      "zoom": 10
    },
    "currency": "EUR",
    "payment_link": "http://site.de/pay.php",
    "promo_url": "http://site.de/about/",
    "google_client_id": "clientID",
    "favicon": "paas/5001/custom.ico",
    "logo": "paas/5001/logo.png",
    "app_logo": "paas/5001/app_logo.png",
    "login_wallpaper": "paas/5001/login.png",
    "desktop_wallpaper": "http://test.com/test.jpg",
    "monitoring_logo": "http://test.com/test.jpg",
    "login_footer": "All rights reserved.",
    "allow_registration": true,
    "show_mobile_apps" : true,
    "show_call_notifications" : true,
    "default_user_settings": {
      "geocoder": "google",
      "route_provider": "progorod",
      "measurement_system": "metric",
      "translit": false
    },
    "display_model_features_link" : true,
    "color_theme": "aqua",
    "app_color_theme": "blue_1",
    "privacy_policy_link": "http://privacy-policy-url",
    "tos": "Terms Of Service text",
    "tracker_model_filter": {
      "exclusion": true,
      "values": []
    },
    "internal": {
      "light_registration": true,
      "demo_tracker_source_id": 14,
      "demo_tracker_label": "Demo tracker"
    },
    "no_register_commands": false
  },
  "demo_ends": "2014-01-01",
  "premium_gis": true,
  "features": ["branding_web"],
  "platform": {

```

```

    "iso_datetime_support": true,
    "history.max_limit": 10,
    "report.max_time_span": "P90D",
    "stats.max_allowed_trackers": 128,
    "stats.max_time_span": "P31D",
    "file_storage.hard_max_file_size": 16777216,
    "form.max_fields_count": 128,
    "form.file_field.max_file_size": 16777216,
    "form.file_field.max_files_per_field": 6,
    "form.file_field.max_count": 16
  }
}

```

- `id` - int. Dealer's ID.
- `ui_domain` - string. Dealer's UI domain.
- `company_url` - string. Dealer's promo site URL.
- `settings` - object. Custom settings. May be null if dealer has not set any custom settings.
  - `domain` - string. The same as `dealer.ui_domain`.
  - `service_title` - string. Title of the service.
  - `locale` - [enum](#). Default locale of the dealer.
  - `demo_login` - string. Dealer's login for demo user or empty string if no demo user available.
  - `demo_password` - string. Dealer's password for demo user or empty string if no demo user available.
  - `maps` - string array. List of available maps, e.g.  
`["roadmap", "cdcom", "osm", "wikimapia", "yandexpublic", "hybrid", "satellite"]`.
  - `default_map` - object. Default map settings.
  - `type` - [enum](#). Default map type.
  - `location` - object. Default map center location.
  - `lat` - float. Latitude.
  - `long` - float. Longitude.
  - `zoom` - int. Default map zoom level.
  - `currency` - [enum](#). Dealer's currency ISO 4217 code.
  - `payment_link` - string. PaaS-dependent link that can be used to refill user's account. Can be null or empty.
  - `promo_url` - string. Customizable "About company" url.
  - `google_client_id` - string. Client ID which must be used to work with Google API or null.



- `favicon` - string. Path or URL to dealer's interface favicon.
- `logo` - string. Path or URL to dealer's logotype.
- `app_logo` - string. Nullable, path or URL to dealer's mobile app logotype.
- `login_wallpaper` - string. Path or URL to dealer's interface login wallpaper.
- `desktop_wallpaper` - string. Path to dealer's interface wallpaper or null.
- `monitoring_logo` - string. Path to dealer's interface monitoring logo or null.
- `login_footer` - string. Footer which will be included in login page.
- `allow_registration` - boolean. If `true` then registration is available for dealer's users. All HTML special chars escaped using HTML entities.
- `show_mobile_apps` - boolean. If `true` then mobile applications are available for dealer's users.
- `show_call_notifications` - boolean. If `true` then call notifications are available for dealer's users.
- `geocoder` - [enum](#). Default geocoder.
- `route_provider` - [enum](#). Default router.
- `measurement_system` - [enum](#). Measurement system.
- `display_model_features_link` - boolean. When `true` show in model info link to [squaregps.com](https://squaregps.com) (UI option).
- `color_theme` - [enum](#). Color theme code or empty string (for default theme).
- `app_color_theme` - [enum](#). Mobile app color theme code or empty string (for default theme).
- `tos` - string. Terms of service text.
- `tracker_model_filter` - object. A filter which describes tracker models available for registration.
- `exclusion` - boolean. If `true` models in the `values` will be excluded.
- `values` - string array. If it is empty - all models available.
- `internal` - object with additional options.
- `light_registration` - boolean. If `true` use "very simple" registration with demo tracker.
- `demo_tracker_source_id` - int. An ID of tracker created on `light_registration`.
- `demo_tracker_label` - string. Label of tracker created on `light_registration`.
- `no_register_commands` - boolean. If `true` then do not send commands to devices on activation.

- `demo_ends` - string. A date when demo for this dealer ends. Is null when dealer is not on Trial tariff.
- `premium_gis` - boolean. If `true` dealer has Premium GIS package.
- `features` - string array. Set of the allowed features for a dealer (all list see below in "Dealer features").
- `platform` - key-value object. Global platform settings.
  - `iso_datetime_support` - boolean, if `true` platform supports ISO 8601 [date/time format](#).
  - `history.max_limit` - int, max limit for [history](#) list actions.
  - `report.max_time_span` - ISO8601 period, max timespan for [reports generation](#).
  - `stats.max_allowed_trackers` - int, max allowed trackers for [stats actions](#).
  - `stats.max_time_span` - ISO8601 period, max timespan for [stats actions](#).
  - `file_storage.hard_max_file_size` - long, hard max file size in bytes for uploading files to the file storage.
  - `form.max_fields_count` - integer, max fields per form.
  - `form.file_field.max_file_size` - long, max file size in bytes for the form file.
  - `form.file_field.max_files_per_field` - integer, max files per form field.
  - `form.file_field.max_count` - integer, max file fields per form.

### Dealer features

| name            | description   |
|-----------------|---|
| branding_web    | Allow to use custom logos, color theme, domain and favicon in UI for web version.     |
| branding_mobile | Allow to use custom icon, logo, color theme in the mobile applications.               |
| subpaas         | Allow to use Sub-Dealers (can be used only together with <code>navixy_label</code> ). |
| navixy_label    | Show "Powered by Navixy" in UI (required for subpaas feature).                        |

### errors

- 12 – Dealer not found (if corresponding dealer not found in the database).

- 201 – Not found in the database (if there is no Ui settings data for corresponding dealer).

Last update: January 22, 2023





# Feedback

Contains feedback object API call to send a feedback email, ask for help or suggest a new feature.

## Feedback object

```
{
  "text": "My feedback",
  "useragent": "Chrome/87.0.4280.88",
  "platform": "Windows NT 10.0; Win64; x64",
  "screenshots": ["encoded image1", "encoded image2"],
  "log": <log_file>
}
```

- `text` - string. Feedback text. May not be null.
- `useragent` - optional string. Information about the browser of user.
- `platform` - optional string. Information about the platform of user.
- `screenshots` - optional string array. base64-encoded data:url image, example: data:image/jpeg;base64, [encoded image] .
- `log` - optional log file. Contains log of the browser.

## API actions

API path: `/feedback` .

### send\_email

Sends an email with user's feedback, ask for help, or suggestion a new feature. The message will be sent to dealer's email address for feedback.

#### parameters

| name     | description  | type        |
|----------|--|-------------|
| feedback | Message from the user. Screenshot and log will be added to email as attachments. | JSON object |

| name | description  | type |
|------|--|------|
| type | Optional. One of strings: <code>support_request</code> (default), <code>feature_request</code> and <code>review</code> . | enum |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/feedback/send_email' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "feedback": {"text": "I love this platform"}, "type": "review"}'
```

## response

```
{ "success": true }
```

## errors

- [General](#) types only.

Last update: December 26, 2022







# File

Contains an API call to get user's file statistic.

## API actions

API path: `/file`.

### stats/read

Gets user's files statistic.

#### parameters

Only API key `hash`.

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/file/stats/read' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/file/stats/read?
hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{
  "success": true,
  "value": {
    "file_count": 24,
    "total_size": 40192953,
    "quota": 104857600
  }
}
```

- `file_count` - int. Count of all uploaded files.
- `total_size` - int. Total files size in bytes.
- `quota` - int. Space available to the user in bytes.

## errors

- [General](#) types only.

Last update: July 19, 2022





# Notification

Contains an API call to get list of user notifications.

## API actions

API path: `/notification`.

### list

Lists user notifications.

#### parameters

Only API key `hash`.

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/notification/list' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/notification/list?  
hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{  
  "success": true,  
  "list": [{  
    "id": 12451529,  
    "message": "notification",  
    "show_till": "2020-12-31 17:27:28"  
  }]  
}
```

- `id` - int. An ID of notification.
- `message` - string. Message of notification.
- `show_till` - date/time. Date until notification should be shown.

## errors

- [General](#) types only.

Last update: December 26, 2022







# Timezone

Contains an API call to get information about all supported timezones.

## API actions

API path: `/timezone`.

### list

Information about all supported timezones for the specified locale. Does not require user authorization.

#### parameter

| name   | description     | type                 |
|--------|-----------------|----------------------|
| locale | Name of locale. | <a href="#">enum</a> |

#### example

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/timezone/list' \
-H 'Content-Type: application/json' \
-d '{"locale": "En-en"}'
```

#### response

```
{
  "success": true,
  "list": [{
    "zone_id": "Australia/Sydney",
    "description": "Sydney",
    "base_offset": 10.0,
    "dst_offset": 1,
    "country_code": "AU",
    "alt_ids": ["Australia/ACT", "Australia/Canberra",
"Australia/NSW"]
  }]
}
```

- `zone_id` - string. Timezone ID, which is used throughout the API.

- `description` - string. Localized description of the timezone.
- `base_offset` - double. Base timezone offset in hours, e.g. 10 means UTC +10. May be negative or fractional!
- `dst_offset` - int. DST offset in hours (0 if no DST rules for this timezone).
- `country_code` - string. ISO country code for the timezone.
- `alt_ids` - string array. List of strings, optional, alternative timezone IDs.

#### **errors**

- [General](#) types only.

Last update: December 26, 2022





# Entity actions

Contains entity object description and API calls to interact with it.

Entity describes a class of objects for which representation and editable fields can be customized. For example, you can add your own custom fields of **places** entity or rearrange existing fields.

## Entity object

```
{
  "id": 123,
  "type": "place",
  "settings": {
    "layout": {
      "sections": [{
        "label": "Section label",
        "field_order": [
          "label",
          "location",
          "131212",
          "tags",
          "description"
        ]
      }]
    }
  }
}
```

- `id` - int. Entity identifier.
- `type` - [enum](#). Currently, only "place" is supported.
- `layout` - object describes layout of fields for entity.
- `sections` - array of objects. Each section can contain one or more fields. At least one section must exist in a layout.
- `label` - string. Name of section.
- `field_order` - string array. Built-in fields and IDs of custom fields (as strings).

### entity types:

**place** - a place object, the same as is available through [place API](#).

Builtin fields:

- `label`.

- location.
- tags.
- description.

## API actions

API path: `/entity`.

### list

Get list of entities which are available for customization.

#### parameters

Only API key `hash`.

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/entity/list' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/entity/list?  
hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{  
  "success": true,  
  "list": [{  
    "id": 123,  
    "type": "place",  
    "settings": {  
      "layout": {  
        "sections": [{  
          "label": "Section label",  
          "field_order": [  
            "label",  
            "location",  
            "131212",  
            "tags",  
            "description"  
          ]  
        }  
      ]  
    }  
  ]  
}
```

```
}]
}
```

## errors


- [General](#) types only.

## read

Gets entity by the ID or by type.

## parameters

| name | description                                       | type   |
|------|---|--------|
| id   | ID of an entity.                                  | int    |
| type | Type of an entity. Entity type string, see above. | string |

 Exactly one of these parameters must be specified. They can't be both null or both non-null.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/entity/read' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id": 131312}'
```

### HTTP GET

```
https://api.navixy.com/v2/entity/read?
hash=a6aa75587e5c59c32d347da438505fc3&id=131312
```

## response

```
{
  "success": true,
  "entity": {
    "id": 123,
    "type": "place",
    "settings": {
      "layout": {
        "sections": [{
          "label": "Section label",
          "field_order": [
```



```

        "label",
        "location",
        "131212",
        "tags",
        "description"
      ]
    }
  }
},
"fields": [{
  "id": 131312,
  "label": "Additional info",
  "type": "text",
  "required": true,
  "description": "Info about place"
}]
}

```

- `fields` - array of objects. Fields associated with this entity. Described in [field object](#).


#### errors

- 201 - Not found in the database – if there is no entity with such ID.

#### update(entity)

Updates settings of customizable entity. Entity must have a valid ID.

**required sub-user rights:** `places_custom_fields_update` for entities with type `place`.

 `entity.settings.layout.sections` **must contain IDs of all builtin and custom fields which are associated with this entity. No fields can be omitted from layout, only reordering allowed. Fields cannot be duplicated, even in different sections.**

#### parameters

| name   | description                               | type        |
|--------|---|-------------|
| entity | Entity object with valid ID and settings. | JSON object |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/entity/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "entity":
{"id": 123, "type": "place", "settings": {"layout": {"sections":
[{"label": "Section label", "field_order": ["label", "location",
"131212", "tags", "description"]}}}}}'
```

## response

```
{
  "success": true
}
```

## errors

- 201 - Not found in the database – if there is no entity with such ID.
- 7 - Invalid parameters - if entity object violates restrictions described above.

Last update: December 26, 2022





# Entity fields

Contains field object description and API calls to interact with it.

## Field object

```
{
  "id": 131312,
  "label": "Additional info",
  "type": "employee",
  "required": false,
  "description": "Responsibility",
  "params": {"responsible": true}
}
```

- `id` - int. Field identifier. Null for the new object.
- `label` - string. Name of the field.
- `type` - [enum](#). Type of field, see below.
- `required` - boolean. Whether field required to be filled or not.
- `description` - string. Additional info about the field, max 512 characters.
- `params` - object. Type-specific parameters. If no specific params, this field should be omitted.

### field types:

Without *Special params*:

- `text` - text field up to 700 unicode symbols.
- `bigtext` - bigger text field, up to 20000 unicode symbols with reduced search and sorting capabilities.
- `email` - field for storing email, validated to contain valid email address.
- `phone` - field for storing phone number, validated to contain valid phone number.
- `decimal` - decimal number from -999999999999.999999 to 999999999999.999999. Values stored up to the sixth decimal place.
- `integer` - integer number from  $-2^{63}$  to  $2^{63} - 1$ .

With *Special params*:

- `employee` - link to employee.

*Special params:*

```
{
  "responsible": true
}
```

`responsible` - boolean. Entities with this set to `true` can be shown to the employee in the mobile app. Only one employee field can have this value set to `true`. If there's an [employee assigned](#) to a Mobile Tracker App ([Android](#) / [iOS](#)), and a [place](#) has a custom field of type "responsible employee", such place will be available in the mobile app to view. Thus, field employee can view all places assigned to him to visit them, etc.

## Fields actions

API path: `/entity/fields`.

Field allows adding custom information to a customizable entity. Each field belongs to one entity.

### read(entity\_id)

Gets a set of custom fields associated with the specified entity. Note that you must know entity ID, which can be obtained from [entity/list](#).

#### parameters

| name      | description      | type |
|-----------|------------------|------|
| entity_id | ID of an entity. | int  |

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/entity/fields/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "entity_id": 131312}'
```

##### HTTP GET

```
https://api.navixy.com/v2/entity/fields/read?
hash=a6aa75587e5c59c32d347da438505fc3&entity_id=131312
```

## response

```
{
  "success": true,
  "list": [{
    "id": 131312,
    "label": "Additional info",
    "type": "employee",
    "required": false,
    "description": "Responsibility",
    "params": {"responsible": true}
  }]
}
```

## errors

- 201 - Not found in the database - if there is no entity with such ID.

## update(entity\_id, fields, delete\_missing)


Updates a set of custom fields associated with the specified entity.

**required sub-user rights:** `places_custom_fields_update` for fields associated with `place` entity.

Fields passed with `id` equal to `null` will be created. If field already exists, its `type` must be equal to type of already stored field (i.e. you can't change a type of field).

All fields associated with the same entity must have different `label`s.

Passing fields with `id` from non-existent fields or fields bound to another entity will result in an error.

 If `delete_missing` is `true`, all existing fields which are missing from the `fields` list will be permanently deleted! Otherwise, they are unaffected.

## parameters

| name      | description  | type        |
|-----------|--|-------------|
| entity_id | ID of an entity.                                   | int         |
| fields    | List of new/existing fields to be created/updated. | JSON object |

| name           | description  | type    |
|----------------|--|---------|
| delete_missing | Optional. Default is <code>false</code> . Delete fields not present in <code>fields</code> list. | boolean |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/entity/fields/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "entity_id": 131312, "fields": {"label": "Additional info", "type": "employee", "required": false, "description": "Responsibility", "params": {"responsible": true}}'
```

## response

A list of **all** fields associated with the specified entity. Newly created fields will have their IDs filled.

```
{
  "success": true,
  "list": [{
    "id": 131312,
    "label": "Additional info",
    "type": "employee",
    "required": false,
    "description": "Responsibility",
    "params": {"responsible": true}
  }]
}
```

## errors

- 201 - Not found in the database – if there is no entity with such ID.
- 7 - Invalid parameters - if fields violate restrictions described above.

Last update: December 26, 2022







# Entity search Conditions

Contains search conditions object description and condition types.

Search conditions used to search and filter list of certain entities by built-in and/or custom fields.

## Search conditions object

```
[
  {
    "type": "and",
    "conditions": [
      {
        "type": "or",
        "conditions": [
          {
            "type": "eq",
            "field": "18",
            "value": 1111
          },
          {
            "type": "contains",
            "field": "27",
            "value": "qqq"
          }
        ]
      },
      {
        "type": "contains",
        "field": "label",
        "value": "who"
      }
    ]
  }
]
```

Conditions represented by an array, each condition during search evaluated, and the result is either `true` or `false`. Thus, boolean operations such as `AND` or `OR` can be applied to them. All conditions in a top-level array joined using `AND` operator.



**A maximum of 72 conditions can be used at once, including nested conditions.**

## Condition types

### "And" condition

Evaluates all specified conditions and joins them using `AND` boolean operator.

```
{
  "type": "and",
  "conditions": [{
    "type": "eq",
    "field": "18",
    "value": 1111
  },
  {
    "type": "contains",
    "field": "27",
    "value": "qqq"
  }]
}
```

### "Or" condition

Evaluates all specified conditions and joins them using `OR` boolean operator.

```
{
  "type": "or",
  "conditions": [{
    "type": "eq",
    "field": "18",
    "value": 1111
  },
  {
    "type": "contains",
    "field": "27",
    "value": "qqq"
  }]
}
```

### "Number equals" condition

Checks if specified field is equal to provided number value. Works for text fields too (e.g. "111" is considered equal to 111). For linked entity fields, it matches linked entity ID to number value.

```
{
  "type": "eq",
  "field": "18",
  "value": 1111
}
```

- `field` - string. A built-in field or field id.
- `value` - int. Number value to which field matched against. Can be decimal. Must be between  $-2^{63}$  and  $2^{63}-1$ . No more than 6 fractions digits.

## "Contains string" condition

Checks if specified field contains substring equal to provided value. Works for number fields too, e.g. (123123 contains "123"). For linked entity fields, it matches value against linked entity label or other similar field (first name, last name, etc.)

```
{
  "type": "contains",
  "field": "label",
  "value": "who"
}
```

- `field` - string. A built-in field or field id.
- `value` - int. string value to which field matched against. Cannot be null or empty, max length is 760.

Last update: December 26, 2022





# Events history

Contains history entry object description and API calls to interact with it.

Find instructions on getting notifications [here](#).

## Tracker history entry

```
{
  "id": 1,
  "type": "tracker",
  "is_read": false,
  "message": "Alarm",
  "time": "2020-01-01 00:00:00",
  "event": "offline",
  "tracker_id": 2,
  "rule_id": 3,
  "track_id": 4,
  "location": {
    "lat": 50.0,
    "lng": 60.0,
    "precision": 50
  },
  "address": "address",
  "extra": {
    "task_id": null,
    "parent_task_id": null,
    "counter_id": null,
    "service_task_id": null,
    "checkin_id": null,
    "place_ids": null,
    "last_known_location": false,
    "tracker_label": "Tracker label",
    "emergency": false,
    "employee_id": 4563
  }
}
```

- `id` - long. An ID of event.
- `type` - [enum](#). Type of device. Can be "socket", "tracker", or "camera".
- `is_read` - boolean. If `true` the notification seen by user and marked as read.
- `message` - string. Notification message.
- `time` - [date/time](#). When this notification received.
- `event` - [enum](#). Type of history event extension. Available event types can be obtained by [/history/type/list](#) action.



- `tracker_id` - int. An ID of the tracker (aka "object\_id"). Tracker must belong to authorized user and not be blocked.
- `rule_id` - int. An ID of assigned rule.
- `track_id` - int. An ID of a track on which the event happened.
- `location` - location object. Location where the event happened.
- `address` - string. Address of location or "" (empty string) if no address for location.
- `extra` - object. Extra fields for events. Like for what task or tracker the event was.
  - `task_id` - int. Related task identifier.
  - `parent_task_id` - int. Related parent task identifier (for task checkpoint related history entries).
  - `counter_id` - int. Related counter identifier.
  - `service_task_id` - int. Related service task ID.
  - `checkin_id` - int. Related check-in marker.
  - `place_ids` - int. Related place identifiers.
  - `last_known_location` - boolean. `true` if location may be outdated.
  - `tracker_label` - string. Tracker label.
  - `emergency` - boolean. `true` for emergency events with the same flag in a rule.
  - `employee_id` - int. Driver ID at the time of the event.

Date/time type described in [data types description section](#).

## API actions

API path: `/history`.

### read

Returns history entry with the specified ID.

#### parameters

| name            | description                        | type |
|-----------------|------------------------------------|------|
| <code>id</code> | <a href="#">History entry ID</a> . | long |

| name              | description  | type    |
|-------------------|--|---------|
| add_tracker_label | Optional. If <code>true</code> tracker label will be added to message. | boolean |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/history/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id": 11231,
    "add_tracker_label": true}'
```

### HTTP GET

```
https://api.navixy.com/v2/history/read?
hash=a6aa75587e5c59c32d347da438505fc3&id=11231&add_tracker_label=true
```

## response

```
{
  "success": true,
  "value": {
    "id": 1,
    "type": "tracker",
    "is_read": false,
    "message": "Alarm",
    "time": "2020-01-01 00:00:00",
    "event": "offline",
    "tracker_id": 2,
    "rule_id": 3,
    "track_id": 4,
    "location": {
      "lat": 50.0,
      "lng": 60.0,
      "precision": 50
    },
    "address": "address",
    "extra": {
      "task_id": null,
      "parent_task_id": null,
      "counter_id": null,
      "service_task_id": null,
      "checkin_id": null,
      "place_ids": null,
      "last_known_location": false,
      "tracker_label": "Tracker label",
      "emergency": false,
      "employee_id": 4563
    }
  }
}
```

## errors

- 201 – Not found in the database - when there are no history entries with that ID.

## mark\_read

Marks history entry as read by `id` (see: [Tracker history entry](#)).

### parameters

| name | description                              | type |
|------|--|------|
| id   | <a href="#">Tracker history entry ID</a> | long |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/history/mark_read' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id": 11231}'
```

#### HTTP GET

```
https://api.navixy.com/v2/history/mark_read?  
hash=a6aa75587e5c59c32d347da438505fc3&id=11231
```

### response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - when there are no history entries with that ID.

## mark\_read\_all

Marks all the user's history entries read.

### parameters

Only API key `hash`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/history/mark_read_all' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/history/mark_read_all?  
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{ "success": true }
```

## errors

- [General](#) types only.

Last update: January 9, 2024





# Tracker events

Contains list method to get tracker's events.

## API actions

API path: `/history/tracker/`.

### list

List less than or equal to `limit` of tracker events filtered by event types ( `events` ) between `from` date/time and `to` date/time sorted by **time** field.

Described this API call usage details in our [how-tos](#).

### parameters

| name      | description   | type                             |
|-----------|---|----------------------------------|
| trackers  | List of tracker's IDs.  | int array                        |
| from      | Start date/time for searching.  | string <a href="#">date/time</a> |
| to        | End date/time for searching. Must be after "from" date.   | string <a href="#">date/time</a> |
| events    | Optional. Default: all. List of history types.  | string array                     |
| limit     | Optional. Default: <a href="#">history.maxLimit</a> . Max count of entries in result.   | int                              |
| ascending | Optional. Default: <code>true</code> . Sort ascending by time when it is <code>true</code> and descending when <code>false</code> . | boolean                          |

If `events` (event types) not passed then list all event types.

Available event types can be obtained by [/history/type/list](#) action.

Default and max limit is 1000. (Note for StandAlone: this value configured by maxHistoryLimit config option).

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/history/tracker/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "trackers":
[131312, 123985], "from": "2020-12-10 16:44:00", "to": "2020-12-22
16:44:00"}'
```

### response

```
{
  "success": true,
  "list": [{
    "id": 1,
    "type": "tracker",
    "is_read": false,
    "message": "Alarm",
    "time": "2020-01-01 00:00:00",
    "event": "offline",
    "tracker_id": 2,
    "rule_id": 3,
    "track_id": 4,
    "location": {
      "lat": 50.0,
      "lng": 60.0,
      "precision": 50
    },
    "address": "address",
    "extra": {
      "task_id": null,
      "parent_task_id": null,
      "counter_id": null,
      "service_task_id": null,
      "checkin_id": null,
      "place_ids": null,
      "last_known_location": false,
      "tracker_label": "Tracker label",
      "emergency": false,
      "employee_id": 4563
    }
  }],
  "limit_exceeded": false
}
```

- `list` - list of zero or more `history_entry` objects which described in [Tracker history entry](#).
- `limit_exceeded` - boolean. `false` when listed all history entries satisfied with conditions and `true` otherwise.



## errors

- 211 – Requested time span is too big - time span between `from` and `to` is more than `report.maxTimeSpan` days.
- 212 – Requested `limit` is too big - `limit` is more than `history.maxLimit`.
- 217 – List contains nonexistent entities – if one of the specified trackers does not exist or is blocked.

Last update: August 1, 2023





# Event type

Contains list method to get event types available to user with localized descriptions.

## API actions

API path: `/history/type`.

### list

Returns available history event types with localized descriptions.

#### parameters

| name                | description  | type    |
|---------------------|--|---------|
| locale              | Locale code to set language of descriptions.   | enum    |
| only_tracker_events | Optional. Default is <code>true</code> . Will return only tracker type events if <code>true</code> . | boolean |

#### example

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/history/type/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "locale": "En-en"}'
```

#### response

```
{
  "success": true,
  "list": [{
    "type": "alarmcontrol",
    "description": "Car alarm"
  }]
}
```

- `type` - string. History event type.
- `description` - string. Localized description.

## errors

- [General](#) types only.

Last update: December 26, 2022





# Unread events

Contains API calls to interact with unread history events.

## API actions

API path: `/history/unread`.

### list

List less than or equal to `limit` of the latest user's unread history events. Described how it works in our [instructions](#).

#### parameters

| name  | description  | type          |
|-------|--|---------------|
| limit | Optional. Limit of entries in response.  | int           |
| from  | Optional. Start <a href="#">date/time</a> for searching. Default <code>from</code> is <b>now</b> minus one year. | date/<br>time |

Default and max limit is [history.maxLimit](#).

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/history/unread/list' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/history/unread/list?  
hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{  
  "success": true,  
  "list": [{  
    "id": 1,
```



```

    "type": "tracker",
    "is_read": false,
    "message": "Alarm",
    "time": "2020-01-01 00:00:00",
    "event": "offline",
    "tracker_id": 2,
    "rule_id": 3,
    "track_id": 4,
    "location": {
      "lat": 50.0,
      "lng": 60.0,
      "precision": 50
    },
    "address": "address",
    "extra": {
      "task_id": null,
      "parent_task_id": null,
      "counter_id": null,
      "service_task_id": null,
      "checkin_id": null,
      "place_ids": null,
      "last_known_location": false,
      "tracker_label": "Tracker label",
      "emergency": false,
      "employee_id": 4563
    }
  }
}

```

- `list` - array of objects. list of zero or more [Tracker history entry](#) objects.

## errors

- 212 – Requested limit is too big (more [history.maxLimit](#) config option).

## count

Get count of user's unread history messages starting `from` date.

## parameters

| name | description  | type                 |
|------|--|----------------------|
| from | Optional. Start <a href="#">date/time</a> for searching. Default <code>from</code> is <b>now</b> minus one year. | date/<br>time        |
| type | Optional. Type of devices that should be count. Can be "socket", "tracker", or "camera".                         | <a href="#">enum</a> |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/history/unread/count' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/history/unread/count?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "count": 1
}
```

## errors

- [General](#) types only.

Last update: August 1, 2023





# User events

Contains list method to get user's events.

## API actions

API path: `/history/user/`.

### list

List less than or equal to `limit` of tracker events filtered by event types ( `events` ) between `from` date/time and `to` date/time sorted by **time** field.

Added more information about this API call usage in our [instructions](#).

#### parameters

| name      | description   | type   |
|-----------|---|--|
| from      | Start date/time for searching.  | string <a href="#">date/</a><br><a href="#">time</a> |
| to        | End date/time for searching. Must be after "from" date.   | string <a href="#">date/</a><br><a href="#">time</a> |
| events    | Optional. Default: all. List of history types.  | string array   |
| limit     | Optional. Default: <a href="#">history.maxLimit</a> . Max count of entries in result.   | int  |
| ascending | Optional. Default: <code>true</code> . Sort ascending by time when it is <code>true</code> and descending when <code>false</code> . | boolean  |

If `events` (event types) not passed then list all event types.

Available event types can be obtained by [/history/user/list](#) action.

Default and max limit is 1000. (Note for StandAlone: this value configured by `maxHistoryLimit` config option).

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/history/user/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "from": "2020-12-10 16:44:00", "to": "2020-12-22 16:44:00"}'
```

## response

```
{
  "success": true,
  "list": [{
    "id": 1,
    "type": "tracker",
    "is_read": false,
    "message": "Alarm",
    "time": "2020-01-01 00:00:00",
    "event": "offline",
    "tracker_id": 2,
    "rule_id": 3,
    "track_id": 4,
    "location": {
      "lat": 50.0,
      "lng": 60.0,
      "precision": 50
    },
    "address": "address",
    "extra": {
      "task_id": null,
      "parent_task_id": null,
      "counter_id": null,
      "service_task_id": null,
      "checkin_id": null,
      "place_ids": null,
      "last_known_location": false,
      "tracker_label": "Tracker label",
      "emergency": false,
      "employee_id": 4563
    }
  }],
  "limit_exceeded": false
}
```

- `list` - list of zero or more `history_entry` objects which described in [Tracker history entry](#).
- `limit_exceeded` - boolean. It indicates if the response has exceeded the `store_period` limit, set in the user's tariff plan. Will be `true` if you request a period that exceeds what the user's plan allows.

## errors

- 211 – Requested time span is too big - time span between `from` and `to` is more than `report.maxTimeSpan` days.
- 212 – Requested `limit` is too big - `limit` is more than `history.maxLimit`.

Last update: November 22, 2023







# Plugin

Contains plugin object description and API calls to interact with it.

Plugins are special software modules which modify the behavior of various API calls.

## Plugin object structure

```
{
  "id": 1,
  "type": "tracker_register",
  "ui_module": "Registration.appPlugins.BundledSim",
  "module": "com.navixy.plugin.tracker.register.bundled_sim",
  "filter": {
    "exclusion": true,
    "values": ["navixymobile", "mobile_unknown.*"]
  },
  "parameters" : {<parameter1>}
```

- `id` - int. An ID of plugin.
- `type` - string. Plugin type.
- `ui_module` - string. Plugin UI module name.
- `module` - string. Plugin module name.
- `filter` - object. A model filter which describes to which device models this plugin is applicable.
  - `exclusion` - boolean. If `true`, "models" lists models NOT supported by this plugin, if `false`, "models" contains all supported models.
  - `values` - string array. List of the regexes for models which are (not) supported by this plugin.
- `parameters` - plugin-specific parameters as JSON object. This field omitted if it's `null` (and it is `null` most of the time).

### object example

```
{
  "id": 4,
  "type": "tracker_report",
  "module": "com.navixy.plugin.tracker.report.trip",
  "ui_module": "Trip",
  "filter": {
    "exclusion": true,
    "values": []
  }
```

```
}  
}
```

## API actions

API path: `/plugin`.

### list

Get all plugins available to the user. List of available plugins may vary from user to user depending on platform settings and purchased features. Only these plugins can be used to register trackers, generate reports, etc.

### parameters

Only API key `hash`.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/plugin/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/plugin/list?  
hash=a6aa75587e5c59c32d347da438505fc3
```

### response

```
{  
  "success": true,  
  "list": [{  
    "id": 4,  
    "type": "tracker_report",  
    "module": "com.navixy.plugin.tracker.report.trip",  
    "ui_module": "Trip",  
    "filter": {  
      "exclusion": true,  
      "values": []  
    }  
  }]  
}
```

- `list` - array of objects. List of available plugins.

## errors

- [General](#) types only.

### Standalone-specific:

If no plugins enabled for user and his dealer then available plugins enabled by default (config options **plugin.tracker.register.defaultIds** and **plugin.tracker.report.defaultIds**).

Last update: December 26, 2022





# Report plugins

Contains report plugins with plugin-specific parameters.

## Trips report

A report on detailed trip history.

### parameters

Default **plugin\_id**: 4.

Plugin-specific parameters:

| name                       | description   | type    |
|----------------------------|---|---------|
| hide_empty_tabs            | If <code>true</code> , empty tabs will be hidden.   | boolean |
| show_seconds               | If <code>true</code> , timestamps will be with seconds.   | boolean |
| include_summary_sheet_only | If <code>true</code> , report will contain only a summary sheet for all chosen devices.   | boolean |
| include_summary_sheet      | If <code>true</code> , report will contain a summary sheet. Default is <code>true</code> .  | boolean |
| split                      | Trips will be split by stops if <code>true</code> .   | boolean |
| show_idle_duration         | Will show idle duration in report if <code>true</code> .  | boolean |
| show_coordinates           | Every address will contain longitude and latitude if <code>true</code> .  | boolean |
| filter                     | If <code>true</code> , short trips will hide (shorter than 300m/have less than 4 points total and if the device circles around one point (e.g., star pattern from GPS drifting)). | boolean |

| name            | description   | type    |
|-----------------|---|---------|
| group_by_driver | Group trips by driver assigned to the device if <code>true</code> . | boolean |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 4,
  "show_seconds": false,
  "include_summary_sheet_only": false,
  "include_summary_sheet": true,
  "split": true,
  "show_idle_duration": false,
  "show_coordinates": false,
  "filter": true,
  "group_by_driver": false
}
```

## Stops report

A report on detailed stops history.

### parameters

Default **plugin\_id**: 6.

Plugin-specific parameters:

| name             | description   | type    |
|------------------|---|---------|
| hide_empty_tabs  | If <code>true</code> , empty tabs will be hidden.   | boolean |
| show_seconds     | If <code>true</code> , timestamps will be with seconds.   | boolean |
| show_coordinates | Every address will contain longitude and latitude if <code>true</code> .  | boolean |
| filter           | If <code>true</code> , short trips will be part of stops (shorter than 300m/have less than 4 points total and if the device circles around one point (e.g., star pattern from GPS drifting)). | boolean |



## plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 6,
  "show_seconds": false,
  "show_coordinates": false,
  "filter": false
}
```

## Trips and stops by shifts report

A report on trips and stops by shifts.

### parameters

Default **plugin\_id**: 77.

Plugin-specific parameters:

| name              | description   | type                |
|-------------------|---|---------------------|
| hide_empty_tabs   | If <code>true</code> , empty tabs will be hidden.   | boolean             |
| show_seconds      | If <code>true</code> , timestamps will be with seconds.   | boolean             |
| shifts            | List of shifts with names, start and end time. e.g.<br>[{"name": "Shift1", "start_time": "00:00",<br>"end_time": "23:59"}]  | array of<br>objects |
| filter            | If <code>true</code> , short trips will not coincide (shorter than 300m/have less than 4 points total and if the device circles around one point (e.g., star pattern from GPS drifting)). | boolean             |
| show_coordinates  | Every address will contain longitude and latitude if <code>true</code> .  | boolean             |
| split_at_midnight | Split shifts at midnight if <code>true</code> .   | boolean             |

- `shifts` is:

```
{
  "shifts": [{
```

```

        "name": "Shift1",
        "start_time": "00:00",
        "end_time": "23:59"
    }]
}

```

### plugin example

```

{
  "hide_empty_tabs": true,
  "plugin_id": 77,
  "show_seconds": false,
  "shifts": [{
    "name": "Shift1",
    "start_time": "00:00",
    "end_time": "12:00"
  }, {
    "name": "Shift2",
    "start_time": "12:00",
    "end_time": "23:59"
  }],
  "filter": true,
  "show_coordinates": false,
  "split_at_midnight": true
}

```

## Geofence visits report

A report on date, time, and mileage in geofence.

### parameters

Default **plugin\_id**: 8.

Plugin-specific parameters:

| name                   | description   | type    |
|------------------------|---|---------|
| hide_empty_tabs        | If <code>true</code> , empty tabs will be hidden.       | boolean |
| show_seconds           | If <code>true</code> , timestamps will be with seconds. | boolean |
| show_mileage           | Adds mileage to the report if <code>true</code> .       | boolean |
| show_not_visited_zones | Will show non visited zones if <code>true</code> .      | boolean |

| name                       | description   | type         |
|----------------------------|---|--------------|
| min_minutes_in_zone        | Minimum minutes in a zone to start determining visit. If the device was in a zone less than a specified time - the visit not count. | int          |
| zone_ids                   | List of zone IDs.   | int<br>array |
| hide_charts                | If <code>true</code> , charts will be hidden.   | boolean      |
| include_summary_sheet_only | If <code>true</code> , report will contain only a summary sheet.  | boolean      |
| include_summary_sheet      | If <code>true</code> , report will contain a summary sheet. Default is <code>true</code> .  | boolean      |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 8,
  "show_seconds": false,
  "include_summary_sheet_only": false,
  "include_summary_sheet": false,
  "show_mileage": false,
  "show_not_visited_zones": false,
  "min_minutes_in_zone": 5,
  "hide_charts": false,
  "zone_ids": [2143181, 2143182]
}
```

## POI visits report

A report on date, time, and the number of POIs visits.

### parameters

Default **plugin\_id**: 85.

Plugin-specific parameters:

| name                       | description   | type      |
|----------------------------|---|-----------|
| hide_empty_tabs            | If <code>true</code> , empty tabs will be hidden.   | boolean   |
| show_seconds               | If <code>true</code> , timestamps will be with seconds.   | boolean   |
| show_mileage               | Adds mileage to the report if <code>true</code> .   | boolean   |
| show_not_visited_places    | Will show non visited POIs if <code>true</code> .   | boolean   |
| min_minutes_in_place       | Minimum minutes in a place to start determining visit. If the device was in a place less than a specified time - the visit not count. | int       |
| place_ids                  | List of place IDs.  | int array |
| hide_charts                | If <code>true</code> , charts will be hidden.   | boolean   |
| include_summary_sheet_only | If <code>true</code> , report will have only a summary sheet.   | boolean   |
| include_summary_sheet      | If <code>true</code> , report will contain a summary sheet. Default is <code>true</code> .  | boolean   |
| fetch_places_by_employees  | If <code>true</code> , places will show assigned employee. Place should be assigned to an employee to show his name.                  | boolean   |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 85,
  "show_seconds": false,
  "include_summary_sheet_only": false,
  "show_mileage": false,
  "show_not_visited_places": false,
  "min_minutes_in_place": 5,
  "hide_charts": false,
  "fetch_places_by_employees": false,
```

```
"place_ids": [1612957, 1886863, 1886864]
}
```

## Car security report

A report on alarms, towing alerts, AutoControl events, and crashes.

### parameters

Default **plugin\_id**: 15.

Plugin-specific parameters:

| name            | description   | type    |
|-----------------|---|---------|
| hide_empty_tabs | If <code>true</code> , empty tabs will be hidden.       | boolean |
| show_seconds    | If <code>true</code> , timestamps will be with seconds. | boolean |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 15,
  "show_seconds": false
}
```

## Emergency button (SOS) report

A report on SOS button events log

### parameters

Default **plugin\_id**: 16.

Plugin-specific parameters:

| name            | description   | type    |
|-----------------|---|---------|
| hide_empty_tabs | If <code>true</code> , empty tabs will be hidden.       | boolean |
| show_seconds    | If <code>true</code> , timestamps will be with seconds. | boolean |

## plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 16,
  "show_seconds": false
}
```

## Fall detection report

A report on fall detection sensor log.

### parameters

Default **plugin\_id**: 17.

Plugin-specific parameters:

| name            | description   | type    |
|-----------------|---|---------|
| hide_empty_tabs | If <code>true</code> , empty tabs will be hidden.       | boolean |
| show_seconds    | If <code>true</code> , timestamps will be with seconds. | boolean |

## plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 17,
  "show_seconds": false
}
```

## Tracker detach report

A report on demounting devices from tracking objects.

### parameters

Default **plugin\_id**: 18.

Plugin-specific parameters:

| name            | description   | type    |
|-----------------|---|---------|
| hide_empty_tabs | If <code>true</code> , empty tabs will be hidden.       | boolean |
| show_seconds    | If <code>true</code> , timestamps will be with seconds. | boolean |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 18,
  "show_seconds": false
}
```

## Overall security report

A report on all events related to security and safety.

### parameters

default **plugin\_id**: 19.

Plugin-specific parameters:

| name            | description   | type    |
|-----------------|---|---------|
| hide_empty_tabs | If <code>true</code> , empty tabs will be hidden.       | boolean |
| show_seconds    | If <code>true</code> , timestamps will be with seconds. | boolean |
| group_by_type   | If <code>true</code> , events will group by type.       | boolean |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 19,
  "show_seconds": false,
  "group_by_type": false
}
```

## Engine hours report

A report on time spent in motion and on idling.

### parameters

default **plugin\_id**: 7.

Plugin-specific parameters:

| name                       | description   | type    |
|----------------------------|---|---------|
| hide_empty_tabs            | If <code>true</code> , empty tabs will be hidden.   | boolean |
| show_seconds               | If <code>true</code> , timestamps will be with seconds.   | boolean |
| show_detailed              | If <code>true</code> , report will contain detailed engine hours tab.   | boolean |
| include_summary_sheet_only | If <code>true</code> , report will contain only a summary sheet for all chosen devices.   | boolean |
| include_summary_sheet      | If <code>true</code> , report will contain a summary sheet. Default is <code>true</code> .  | boolean |
| filter                     | If <code>true</code> , short trips will not coincide (shorter than 300m/have less than 4 points total and if the device circles around one point (e.g., star pattern from GPS drifting)). | boolean |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 7,
  "show_seconds": false,
  "show_detailed": false,
  "include_summary_sheet_only": false,
  "filter": true
}
```



## Fuel volume report

A report on fuel refills, drains, consumption (based on fuel level sensor).

### parameters

default **plugin\_id**: 10.

Plugin-specific parameters:

| name                              | description  | type    |
|-----------------------------------|--|---------|
| show_seconds                      | If <code>true</code> , timestamps will be with seconds.  | boolean |
| graph_type                        | The type of X-axis. Can be "time" or "mileage".  | enum    |
| detailed_by_dates                 | If <code>true</code> , show final data on fuel traffic for each day in the period.   | boolean |
| include_summary_sheet_only        | If <code>true</code> , report will contain only a summary sheet for all chosen devices.  | boolean |
| include_summary_sheet             | If <code>true</code> , report will contain a summary sheet. Default is <code>true</code> .   | boolean |
| use_ignition_data_for_consumption | Calculate consumption only when the ignition was on if <code>true</code> .   | boolean |
| include_mileage_plot              | Optional. Used if <code>graph_type = time</code> . Show mileage plot if <code>true</code> .  | boolean |
| filter                            | If <code>true</code> , short trips will not coincide (shorter than 300m/ have less than 4 points total and if the device circles around one point (e.g., star pattern from GPS drifting)). | boolean |

| name                   | description   | type    |
|------------------------|---|---------|
| include_speed_plot     | If <code>true</code> , show speed plot.   | boolean |
| smoothing              | Smooth graph if <code>true</code> .<br>Smoothing reduces the accuracy of calculating refills or drains. | boolean |
| surge_filter           | If <code>true</code> , enables surge filter.  | boolean |
| surge_filter_threshold | Defines a level of surge filter.<br>Can be 0.01 - 0.99.   | float   |
| speed_filter           | If <code>true</code> , enables speed filter.  | boolean |
| speed_filter_threshold | Defines a speed filter threshold.   | int     |

### plugin example

```
{
  "show_seconds": false,
  "plugin_id": 10,
  "graph_type": "mileage",
  "detailed_by_dates": true,
  "include_summary_sheet_only": false,
  "use_ignition_data_for_consumption": false,
  "include_mileage_plot": false,
  "filter": true,
  "include_speed_plot": false,
  "smoothing": false,
  "surge_filter": true,
  "surge_filter_threshold": 0.2,
  "speed_filter": false,
  "speed_filter_threshold": 10
}
```

## Flow meter report

A report on fuel consumption counted by flow meter sensors.

### parameters

default **plugin\_id**: 78.

Plugin-specific parameters:

| name                       | description   | type    |
|----------------------------|---|---------|
| detailed_by_dates          | If <code>true</code> , a table with statistics for every single day in selected date range will be added to the report.   | boolean |
| filter                     | If <code>true</code> , short trips will not coincide (shorter than 300m/have less than 4 points total and if the device circles around one point (e.g., star pattern from GPS drifting)). | boolean |
| include_summary_sheet_only | If <code>true</code> , report will contain only a summary sheet for all chosen devices.   | boolean |
| include_summary_sheet      | If <code>true</code> , report will contain a summary sheet. Default is <code>true</code> .  | boolean |

#### plugin example

```
{
  "detailed_by_dates": true,
  "plugin_id": 78,
  "include_summary_sheet_only": false,
  "filter": true
}
```

## Vehicle sensors report

A report on CAN-bus and OBD2-port data.

#### parameters

default **plugin\_id**: 22.


Plugin-specific parameters:

| name                     | description                                       | type    |
|--------------------------|---|---------|
| hide_empty_tabs          | If <code>true</code> , empty tabs will be hidden. | boolean |
| details_interval_seconds |   | int     |

| name                     | description  | type             |
|--------------------------|--|------------------|
|                          | The interval in seconds. From 30 to 21600.                         |                  |
| details_interval_minutes | Deprecated! The interval in minutes. Can be [5, 30, 60, 180, 360]. | int              |
| graph_type               | The type of X-axis. Can be "time" or "mileage".                    | enum             |
| smoothing                | Smooth data if true.   | boolean          |
| sensors                  | List of objects containing tracker_id and sensor_id.               | array of objects |

- `sensors` is:

```
{
  "sensors": [{
    "tracker_id": 37714,
    "sensor_id": 57968
  }]
}
```

 **Parameter** `details_interval_minutes` **is deprecated. Please use** `details_interval_seconds`.

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 22,
  "details_interval_seconds": 60,
  "graph_type": "time",
  "smoothing": false,
  "sensors": [{
    "tracker_id": 993495,
    "sensor_id": 1378566
  }]
}
```

## Speed violation

A report on speeding instances.

### parameters

default **plugin\_id**: 27.

Plugin-specific parameters:

| name                 | description   | type    |
|----------------------|---|---------|
| hide_empty_tabs      | If <code>true</code> , empty tabs will be hidden.   | boolean |
| show_seconds         | If <code>true</code> , timestamps will be with seconds.   | boolean |
| min_duration_minutes | A minimum time in seconds when speed is more than <code>max_speed</code> to determine violation.  | int     |
| max_speed            | A maximum speed to determine violation.   | int     |
| group_by_driver      | Group violations by driver assigned to the device if <code>true</code> .  | boolean |
| filter               | If <code>true</code> , short trips will not coincide (shorter than 300m/have less than 4 points total and if the device circles around one point (e.g., star pattern from GPS drifting)). | boolean |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 27,
  "show_seconds": false,
  "min_duration_minutes": 5,
  "max_speed": 60,
  "group_by_driver": false,
  "filter": true
}
```

## Device switching ON/OFF report

A report on switching device using hardware switch.

## parameters

default **plugin\_id**: 23.

Plugin-specific parameters:

| name            | description   | type    |
|-----------------|---|---------|
| hide_empty_tabs | If <code>true</code> , empty tabs will be hidden.       | boolean |
| show_seconds    | If <code>true</code> , timestamps will be with seconds. | boolean |

## plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 23,
  "show_seconds": false
}
```

## GSM connection lost

A report on long disruptions of server connection

## parameters

default **plugin\_id**: 13.

Plugin-specific parameters:

| name            | description   | type    |
|-----------------|---|---------|
| hide_empty_tabs | If <code>true</code> , empty tabs will be hidden.       | boolean |
| show_seconds    | If <code>true</code> , timestamps will be with seconds. | boolean |

## plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 13,
  "show_seconds": false
}
```

## Measuring sensors report

A report on detailed sensor reading history.

### parameters

default **plugin\_id**: 9.

Plugin-specific parameters:

| name                     | description   | type             |
|--------------------------|---|------------------|
| hide_empty_tabs          | If <code>true</code> , empty tabs will be hidden.   | boolean          |
| details_interval_seconds | The interval in seconds. From 30 to 21600.  | int              |
| details_interval_minutes | Deprecated! The interval in minutes. Can be <code>[5, 30, 60, 180, 360]</code> .  | int              |
| graph_type               | The type of X-axis. Can be "time" or "mileage".   | enum             |
| smoothing                | Smooth data if <code>true</code> .  | boolean          |
| show_address             | Address of each reading appears in report if <code>true</code> .  | boolean          |
| filter                   | If <code>true</code> , short trips will not coincide (shorter than 300m/have less than 4 points total and if the device circles around one point (e.g., star pattern from GPS drifting)). | boolean          |
| sensors                  | List of objects containing tracker_id and sensor_id.  | array of objects |

- `sensors` is:

```
{
  "sensors": [{
    "tracker_id": 37714,
    "sensor_id": 57968
  }]
```

```
    }  
  }  
}
```



**Param** `details_interval_minutes` **is deprecated. Please sue**  
`details_interval_seconds`.

### plugin example

```
{  
  "hide_empty_tabs": true,  
  "plugin_id": 9,  
  "details_interval_seconds": 60,  
  "graph_type": "time",  
  "smoothing": false,  
  "show_address": false,  
  "filter": true,  
  "sensors": [{  
    "tracker_id": 993495,  
    "sensor_id": 1378566  
  }]  
}
```

## Equipment working time

A report on activity and idle time of the equipment.

### parameters

default **plugin\_id**: 12.

| name                        | description   | type    |
|-----------------------------|---|---------|
| hide_empty_tabs             | If <code>true</code> , empty tabs will be hidden.                             | boolean |
| show_seconds                | If <code>true</code> , timestamps will be with seconds.                       | boolean |
| min_working_period_duration | A minimum time in seconds the equipment works to determine activity. Min = 1. | int     |
| show_idle_percent           | If <code>true</code> , show percentage of idling.                             | boolean |
| filter                      |   | boolean |



| name    | description   | type             |
|---------|---|------------------|
|         | If <code>true</code> , short trips will not coincide (shorter than 300m/have less than 4 points total and if the device circles around one point (e.g., star pattern from GPS drifting)). |                  |
| sensors | List of objects containing tracker_id and sensor_id.  | array of objects |

- `sensors` is:

```
{
  "sensors": [{
    "tracker_id": 37714,
    "sensor_id": 57968
  }]
}
```

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 12,
  "show_seconds": false,
  "min_working_period_duration": 60,
  "show_idle_percent": false,
  "filter": false,
  "sensors": [{
    "tracker_id": 993495,
    "sensor_id": 1378562
  }]
}
```

## Tasks report

A report on tasks statuses.

### parameters

default **plugin\_id**: 42.

| name                  | description   | type    |
|-----------------------|---|---------|
| hide_empty_tabs       | If <code>true</code> , empty tabs will be hidden.       | boolean |
| show_seconds          | If <code>true</code> , timestamps will be with seconds. | boolean |
| show_external_id      | Show external ID of task, if <code>true</code> .        | boolean |
| show_description      | Show description of task, if <code>true</code> .        | boolean |
| show_forms            | Show forms when the task has it, if <code>true</code> . | boolean |
| show_places_and_zones | Show places and geofences, if <code>true</code> .       | boolean |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 42,
  "show_seconds": false,
  "show_external_id": false,
  "show_description": false,
  "show_forms": true,
  "show_places_and_zones": false
}
```

## Form completion statistics report

A report on form fields completion rate.

### parameters

default **plugin\_id**: 70.

| name             | description   | type    |
|------------------|---|---------|
| hide_empty_tabs  | If <code>true</code> , empty tabs will be hidden.                   | boolean |
| show_nonselected | If <code>true</code> , not selected options in forms will be shown. | boolean |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 70,
  "show_nonselected": true
}
```

## Work statuses report

A report on status changes history.

### parameters

default **plugin\_id**: 47.

| name            | description   | type    |
|-----------------|---|---------|
| hide_empty_tabs | If <code>true</code> , empty tabs will be hidden.       | boolean |
| show_seconds    | If <code>true</code> , timestamps will be with seconds. | boolean |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 47,
  "show_seconds": false
}
```

## Check-in report

A report on markers for Check-in function. Available only for X-GPS Trackers.

### parameters

default **plugin\_id**: 80

Plugin-specific parameters:

| name                  | description   | type    |
|-----------------------|---|---------|
| show_coordinates      | If <code>true</code> , coordinates will be added to the report.           | boolean |
| hide_empty_tabs       | If <code>true</code> , empty tabs will be hidden.                         | boolean |
| show_coordinates      | Every address will contain longitude and latitude, if <code>true</code> . | boolean |
| show_places_and_zones | Show places and geofences, if <code>true</code> .                         |         |
| show_forms            | Show forms when the task has it, if <code>true</code> .                   | boolean |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 80,
  "show_coordinates": false,
  "show_places_and_zones": false,
  "show_forms": true
}
```

## Driver shift change report

A report on driver identification.

### parameters

default **plugin\_id**: 66.

Plugin-specific parameters:

| name            | description   | type    |
|-----------------|---|---------|
| hide_empty_tabs | If <code>true</code> , empty tabs will be hidden.       | boolean |
| show_seconds    | If <code>true</code> , timestamps will be with seconds. | boolean |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 66,
  "show_seconds": false
}
```

## Trips by state

A report on trips breakdown by jurisdictions.

### parameters

default **plugin\_id**: 73.

Plugin-specific parameters:

| name                       | description   | type    |
|----------------------------|---|---------|
| hide_empty_tabs            | If <code>true</code> , empty tabs will be hidden.   | boolean |
| show_seconds               | If <code>true</code> , timestamps will be with seconds.   | boolean |
| filter                     | If <code>true</code> , short trips will not coincide (shorter than 300m/have less than 4 points total and if the device circles around one point (e.g., star pattern from GPS drifting)). | boolean |
| include_summary_sheet_only | If <code>true</code> , report will contain only a summary sheet for all chosen devices.   | boolean |
| include_summary_sheet      | If <code>true</code> , the report will contain a summary sheet. Default is <code>true</code> .  | boolean |
| group_type                 | A group type. Can be "province" or "country".   | enum    |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 73,
}
```

```
"show_seconds": false,
"filter": false,
"include_summary_sheet_only": false,
"group_type": "province"
}
```

## Report on all events

An overall report about any kind of events.

### parameters

default **plugin\_id**: 11.

Plugin-specific parameters:

| name            | description   | type         |
|-----------------|---|--------------|
| hide_empty_tabs | If <code>true</code> , empty tabs will be hidden.       | boolean      |
| show_seconds    | If <code>true</code> , timestamps will be with seconds. | boolean      |
| group_by_type   | Groups events by type if <code>true</code> .            | boolean      |
| event_types     | A list of event types that will be considered.          | string array |

- the object with all `event_types` is:

```
{
  "event_types": [
    "auto_geofence_in",
    "auto_geofence_out",
    "door_alarm",
    "forward_collision_warning",
    "gps_lost",
    "gps_recover",
    "gsm_damp",
    "harsh_driving",
    "headway_warning",
    "hood_alarm",
    "idle_end",
    "idle_start",
    "ignition",
    "inroute",
    "outroute",
    "lane_departure",
    "obd_plug_in",
  ]
}
```

```
"obd_unplug",
"pedals_collision_warning",
"pedals_in_danger_zone",
"odometer_set",
"online",
"output_change",
"security_control",
"tracker_rename",
"track_end",
"track_start",
"tsr_warning",
"sensor_inrange",
"sensor_outrange",
"work_status_change",
"call_button_pressed",
"driver_changed",
"driver_identified",
"driver_not_identified",
"fueling",
"drain",
"checkin_creation",
"tachometer",
"antenna_disconnect",
"check_engine_light",
"location_response",
"backup_battery_low",
"fatigue_driving",
"inzone",
"outzone",
"speedup",
"alarmcontrol",
"battery_off",
"bracelet_close",
"bracelet_open",
"case_closed",
"case_opened",
"crash_alarm",
"detach",
"g_sensor",
"input_change",
"light_sensor_bright",
"light_sensor_dark",
"lock_closed",
"lock_opened",
"lowpower",
"offline",
"parking",
"poweroff",
"poweron",
"sos",
"strap_bolt_cut",
"strap_bolt_ins",
"vibration_start",
"vibration_end",
"proximity_violation_start",
"proximity_violation_end",
"force_location_request",
```

```
    "info"  
  ]  
}
```

### plugin example

```
{  
  "hide_empty_tabs": true,  
  "plugin_id": 11,  
  "show_seconds": false,  
  "group_by_type": false,  
  "event_types": [  
    "force_location_request",  
    "info",  
    "inzone",  
    "outzone",  
    "speedup"  
  ]  
}
```

## Geofence entry/exit events

A report on ins ad outs of a certain geofence.

### parameters

default **plugin\_id**: 89.

Plugin-specific parameters:

| name                | description   | type    |
|---------------------|---|---------|
| hide_empty_tabs     | If <code>true</code> , empty tabs will be hidden.   | boolean |
| show_seconds        | If <code>true</code> , timestamps will be with seconds.   | boolean |
| min_minutes_in_zone | Minimum minutes in a zone to start determining visit. If the device was in a zone less than a specified time - the visit not count. | int     |

### plugin example

```
{  
  "hide_empty_tabs": true,  
  "plugin_id": 89,  
  "show_seconds": false,
```



```
"min_minutes_in_zone": 5
}
```

## SMS-locations report

A report on location requests over SMS channel.

### parameters

default **plugin\_id**: 20.

Plugin-specific parameters:

| name            | description   | type    |
|-----------------|---|---------|
| hide_empty_tabs | If <code>true</code> , empty tabs will be hidden.       | boolean |
| show_seconds    | If <code>true</code> , timestamps will be with seconds. | boolean |

### plugin example

```
{
  "hide_empty_tabs": true,
  "plugin_id": 20,
  "show_seconds": false
}
```

## Point report

Information on the points transmitted during the day. Maximum period is 24 hours.

### parameters

default **plugin\_id**: 91.

Plugin-specific parameters:

| name         | description   | type    |
|--------------|---|---------|
| show_seconds | If <code>true</code> , timestamps will be with seconds. | boolean |

### plugin example

```
{
  "show_seconds": true,
  "plugin_id": 91
}
```

## Eco-driving report by trackers

A report on safety driving by trackers. For [report/generate](#) request use trackers parameter.

### parameters

default **plugin\_id**: 46.

Plugin-specific parameters:

| name                    | description  | type             |
|-------------------------|--|------------------|
| harsh_driving_penalties | A list of penalties for harsh driving.   | array of objects |
| speeding_penalties      | A list of penalties for speeding.  | array of objects |
| speed_limit             | Max permitted speed value.   | int              |
| idling_penalty          | Penalty for idling.  | int              |
| min_idling_duration     | A minimum time in minutes to determine idling.   | int              |
| min_speeding_duration   | A minimum time in minutes when speed is more than <code>speed_limit</code> to determine violation. | int              |
| use_vehicle_speed_limit | If <code>true</code> vehicle speed limit used instead of <code>speed_limit</code> parameter.       | boolean          |
| show_seconds            | If <code>true</code> , timestamps will be with seconds.  | boolean          |

- `harsh_driving_penalties` is:

```
{
  "harsh_driving_penalties": {
    "harshAcceleration": 5,
    "harshBraking": 5,
    "harshTurn": 5,
    "harshAccelerationNTurn": 12,
    "harshBrakingNTurn": 12,
    "harshQuickLaneChange": 12
  }
}
```

- `speeding_penalties` is:

```
{
  "speeding_penalties": {
    "10": 2,
    "20": 10,
    "30": 25,
    "50": 75
  }
}
```

"10", "20", "30", "50" - the number of penalty points assigned for speeding by 10, 20, 30, and 50 km/h.

### plugin example

```
{
  "speeding_penalties": {
    "10": 2,
    "20": 10,
    "30": 25,
    "50": 75
  },
  "harsh_driving_penalties": {
    "harshAcceleration": 5,
    "harshBraking": 5,
    "harshTurn": 5,
    "harshBrakingNTurn": 12,
    "harshAccelerationNTurn": 12,
    "harshQuickLaneChange": 12
  },
  "speed_limit": 260,
  "idling_penalty": 5,
  "min_speeding_duration": 1,
  "min_idling_duration": 5,
  "use_vehicle_speed_limit": true,
  "plugin_id": 46,
  "show_seconds": false
}
```

## Eco-driving report by drivers

A report on safety driving by drivers. For [report/generate](#) request use employees parameter.

### parameters

default **plugin\_id**: 82.

Plugin-specific parameters:

| name                    | description  | type             |
|-------------------------|--|------------------|
| harsh_driving_penalties | A list of penalties for harsh driving.   | array of objects |
| speeding_penalties      | A list of penalties for speeding.  | array of objects |
| speed_limit             | Max permitted speed value.   | int              |
| idling_penalty          | Penalty for idling.  | int              |
| min_idling_duration     | A minimum time in minutes to determine idling.   | int              |
| min_speeding_duration   | A minimum time in minutes when speed is more than <code>speed_limit</code> to determine violation. | int              |
| use_vehicle_speed_limit | If <code>true</code> vehicle speed limit used instead of <code>speed_limit</code> parameter.       | boolean          |
| show_seconds            | If <code>true</code> , timestamps will be with seconds.  | boolean          |

### plugin example

```
{
  "speeding_penalties": {
    "10": 2,
    "20": 10,
    "30": 25,
    "50": 75
  },
  "harsh_driving_penalties": {
```

```
"harshAcceleration": 5,
"harshBraking": 5,
"harshTurn": 5,
"harshBrakingNTurn": 12,
"harshAccelerationNTurn": 12,
"harshQuickLaneChange": 12
},
"speed_limit": 260,
"idling_penalty": 5,
"min_speeding_duration": 1,
"min_idling_duration": 5,
"use_vehicle_speed_limit": true,
"plugin_id": 82,
"show_seconds": false
}
```

## Stay in zones report

### parameters

default **plugin\_id**: 84

plugin-specific parameters:

| name                | description   | type             |
|---------------------|---|------------------|
| show_seconds        | If <code>true</code> , time values in report should have format with seconds. Default is <code>false</code> . | boolean          |
| show_tags           | If <code>true</code> , tags fields will be added to the report. Default is <code>false</code> .               | boolean          |
| min_minutes_in_zone | Minimum time in zone (geofence). Default is <code>5</code> .  | int, min value 1 |
| zone_ids            | IDs of user zones, required, min size 1, max size 30  | int array        |

### plugin example

```
{
  "show_seconds": true,
  "show_tags": true,
  "min_minutes_in_zone": 1,
  "zone_ids": [2143181, 2143182],
  "plugin_id": 84
}
```

## Stay in places report

### parameters

default **plugin\_id**: 85

plugin-specific parameters:

| name                      | description   | type      |
|---------------------------|---|-----------|
| fetch_places_by_employees | If <code>true</code> , report will be built for places that are related to selected trackers via custom fields. Cannot be used in conjunction with <code>place_ids</code> | boolean   |
| hide_charts               | If <code>true</code> , charts will be hidden.   | boolean   |
| min_minutes_in_place      | Minimum time in spent in place. Minimum value is 1, default is <code>5</code>   | int       |
| place_ids                 | IDs of user's POI. Min size 1, max size 30  | int array |
| show_mileage              | Adds mileage to the report if <code>true</code> .   | boolean   |
| show_not_visited_places   | Will show non visited POIs if <code>true</code> .   | boolean   |
| show_seconds              | If <code>true</code> , time values in report should have format with seconds. Default is <code>false</code> .   | boolean   |

### plugin example

```
{
  "show_seconds": true,
  "min_minutes_in_place": 1,
  "fetch_places_by_employees": false,
  "hide_charts": true,
  "place_ids": [278645, 278646],
  "show_mileage": true,
  "show_not_visited_places": true,
  "plugin_id": 85
}
```

Last update: February 13, 2023







# Report schedule

Particular report can be delivered to user's mailbox regularly. Contains report schedule object description and API calls to interact with it.

schedule\_entry object:

```
{
  "id": 1,
  "enabled": true,
  "parameters": {
    "period": "1m",
    "schedule": {
      "type": "weekdays",
      "weekdays": [1, 2, 3, 4, 5]
    },
  },
  "report": {
    "trackers": [1],
    "title": "Title",
    "time_filter": {
      "from": "00:00:00",
      "to": "23:59:59",
      "weekdays": [1, 2, 3, 4, 5, 6, 7]
    },
  },
  "geocoder": "yandex",
  "plugin": {
    "plugin_id": 4,
    "show_idle_duration": false
  },
  "emails": ["email@example.ru"],
  "email_format": "pdf",
  "email_zip": false,
  "sending_time": "12:00:00"
},
"fire_time": "2014-09-05 00:00:00",
"last_result": {
  "success": true,
  "id": 1
}
}
```

- `id` - int. Schedule id, ignored on create.
- `enabled` - boolean. `true` if the scheduled report enabled.
- `period` - string. Report period, "Xm" | "w" | "d" | "y".
- `emails` - optional string array. List of emails.

- `email_format` - [enum](#). Can be "pdf" | "xls".
- `sending_time` - optional string. Local time for sending reports, default "00:00:00", hourly granularity.
- `fire_time` - optional string. Last schedule fire time, ignored on create/update.
- `last_result` object with last report creation result.
  - `id` - int. An ID of generated report.

## API actions

API path: `/report/schedule`.

### create

Creates a new report schedule entry.

**required sub-user rights:** `reports`.

### parameters

| name     | description  | type        |
|----------|--|-------------|
| schedule | Schedule object without fields "id", "fire_time", "last_result". | JSON object |

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/schedule/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "schedule":
{"enabled": true, "parameters": {"report": {"title": "Trip
report", "trackers": [669673], "time_filter": {"from": "00:00:00",
"to": "23:59:59", "weekdays": [1,2,3,4,5,6,7]}, "plugin":
{"hide_empty_tabs": true, "plugin_id": 4, "show_seconds": false,
"include_summary_sheet_only": false, "split": true,
"show_idle_duration": false, "show_coordinates": false, "filter":
true, "group_by_driver": false}}, "period": "1w", "email_zip":
false, "email_format": "xls", "emails": ["test@example.com"],
"sending_time": "00:00:00", "schedule": {"type": "weekdays",
"weekdays": [1]}]]}]}'
```

### response

```
{
  "success": true,
  "id": 111
}
```

- `id` - int. An ID of the created schedule entry.

#### errors

- 217 - List contains nonexistent entities - if one or more of tracker IDs belong to nonexistent tracker (or to a tracker belonging to different user).
- 222 - Plugin not found - if specified report plugin not found.
- 236 - Feature unavailable due to.

## delete

Deletes report schedule with the specified ID.

**required sub-user rights:** `reports`.

#### parameters

| name                     | description                          | type |
|--------------------------|--------------------------------------|------|
| <code>schedule_id</code> | ID of the report schedule to delete. | int  |

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/schedule/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3",
    "schedule_id": 1234567}'
```

##### HTTP GET

```
https://api.navixy.com/v2/report/schedule/delete?
hash=a6aa75587e5c59c32d347da438505fc3&schedule_id=1234567
```

#### response

```
{
  "success": true
}
```

## errors

- 201 - Not found in the database - if there is no schedule with specified ID.

## list

Get all report schedules belonging to user.

**required sub-user rights:** reports.

## parameters

Only API key `hash`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/schedule/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

### HTTP GET

```
https://api.navixy.com/v2/report/schedule/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [{
    "id": 1,
    "enabled": true,
    "parameters": {
      "period": "1m",
      "schedule": {
        "type": "weekdays",
        "weekdays": [1, 2, 3, 4, 5]
      },
    },
    "report": {
      "trackers": [1],
      "title": "Title",
      "time_filter": {
        "from": "00:00:00",
        "to": "23:59:59",
        "weekdays": [1, 2, 3, 4, 5, 6, 7]
      },
    },
    "geocoder": "yandex",
    "plugin": {
      "plugin_id": 4,
      "show_idle_duration": false
    }
  }]
}
```

```
    },
    "emails": ["email@example.ru"],
    "email_format": "pdf",
    "email_zip": false,
    "sending_time": "12:00:00"
  },
  "fire_time": "2014-09-05 00:00:00",
  "last_result": {
    "success": true,
    "id": 1
  }
}]
}
```

## errors

[General](#) types only.

## update

Update existing report schedule.

**required sub-user rights:** `reports`.

## parameters

| name     | description  | type        |
|----------|--|-------------|
| schedule | Schedule object without fields "fire_time", "last_result". | JSON object |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/schedule/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "schedule":
{"enabled": true, "parameters": {"report": {"title": "Trip
report", "trackers": [669673], "time_filter": {"from": "00:00:00",
"to": "23:59:59", "weekdays": [1,2,3,4,5,6,7]}, "plugin":
{"hide_empty_tabs": true, "plugin_id": 4, "show_seconds": false,
"include_summary_sheet_only": false, "split": true,
"show_idle_duration": false, "show_coordinates": false, "filter":
true, "group_by_driver": false}}, "period": "1w", "email_zip":
false, "email_format": "xls", "emails": ["test@example.com"],
"sending_time": "00:00:00", "schedule": {"type": "weekdays",
"weekdays": [1]}]]}'
```

## response

```
{  
  "success": true  
}
```

#### **errors**

- 217 - List contains nonexistent entities - if one or more of tracker IDs belong to nonexistent tracker (or to a tracker belonging to different user).
- 222 - Plugin not found - if specified report plugin not found.
- 236 - Feature unavailable due to tariff restrictions - if device's tariff does not allow usage of reports.

Last update: December 26, 2022







# Report tracker

User reports allow acquiring all-round statistics and analytics. The summary data can be shown in various perspectives, in tables and graphs. Contains API calls to interact with tracker reports.

Find information on obtaining data from report in our [how-tos](#).

## API actions

API path: `/report/tracker`.

### delete

Deletes a report from the database.

*required sub-user rights:* `reports`.

### parameters

| name      | description                            | type |
|-----------|--|------|
| report_id | ID of a report that should be deleted. | int  |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/delete' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "report_id":  
1234567}'
```

#### HTTP GET

```
https://api.navixy.com/v2/report/tracker/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&report_id=1234567
```

### response

```
{  
  "success": true  
}
```

## errors

- 101 – In demo mode this function disabled.

## download

Retrieve generated report as a file.

**required sub-user rights:** reports

## parameters

| name      | description  | type    |
|-----------|--|---------|
| report_id | ID of a report that should be deleted.   | int     |
| format    | A format of report that should be downloaded. Can be "xls", "xlsx" or "pdf".                     | enum    |
| headless  | Optional parameter. Default= false . If need report without title page and TOC, set it to true . | boolean |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/download' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "report_id":  
1234567, "format": "pdf"}'
```

### HTTP GET

```
https://api.navixy.com/v2/report/tracker/download?  
hash=a6aa75587e5c59c32d347da438505fc3&report_id=1234567&format=pdf
```

## response

A report rendered to file (standard file download).

## errors

- 204 - Entity not found - if report with the specified ID not found.
- 229 - Requested data is not ready yet - if report exists, but its generation is still in progress.

## generate

Requests a report generation with the specified parameters.

**required sub-user rights:** `reports`.

### parameters

| name        | description   | type           |
|-------------|---|----------------|
| from        | A string containing <a href="#">date/time</a> .   | string         |
| to          | A string containing <a href="#">date/time</a> . Specified date must be after "from" date.   | string         |
| title       | Report title. Default title will be used if null.   | string         |
| geocoder    | Which geocoder to use. See <a href="#">geocoder/</a> .  | string         |
| trackers    | List of trackers' IDs to be included in report (if report is by trackers).  | int<br>array   |
| employees   | List of employees' IDs to be included in report (if report is by employees. For example, <a href="#">plugin ID 82</a> ).  | int<br>array   |
| time_filter | An object which contains everyday time and weekday limits for processed data, e.g. <code>{ "to": "18:00", "from": "12:00", "weekdays": [1, 2, 3, 4, 5] }</code> . | JSON<br>object |
| plugin      | A plugin object (see below).  | JSON<br>object |

### Parameter object fields:

Part of parameters are plugin-specific. See "[Tracker report plugins](#)" section. Common parameters are:

| name      | description   | type |
|-----------|---|------|
| plugin_id | An ID of a tracker report plugin which will be used to generate report. | int  |

| name         | description  | type    |
|--------------|--|---------|
| show_seconds | Flag to define whether time values in report should have format with seconds. <code>true</code> - show seconds, <code>false</code> - don't show seconds. | boolean |

#### Plugin example:

```
{
  "details_interval_seconds": 300,
  "plugin_id": 9,
  "show_seconds": false,
  "graph_type": "time",
  "smoothing": false,
  "sensors": [
    {
      "tracker_id": 123456,
      "sensor_id": 123456
    }
  ]
}
```

#### example

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/generate' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "title": "Trip report", "trackers": [669673], "from": "2020-10-05 00:00:00", "to": "2020-10-06 23:59:59", "time_filter": {"from": "00:00:00", "to": "23:59:59", "weekdays": [1,2,3,4,5,6,7]}, "plugin": {"hide_empty_tabs": true, "plugin_id": 4, "show_seconds": false, "include_summary_sheet_only": false, "split": true, "show_idle_duration": false, "show_coordinates": false, "filter": true, "group_by_driver": false}}'
```

#### response

```
{
  "success": true,
  "id": 222
}
```

- `id` - int. An ID of the report queued for generation. Can be used to request report generation status and to retrieve generated report.

## errors

- 15 - Too many requests / rate limit exceeded - the number of reports created by one user in parallel limited.
- 211 - Requested time span is too big - interval from `from` to `to` is bigger then max allowed time span (see response).

```
{
  "success": false,
  "status": {
    "code": 211,
    "description": "Requested time span is too big"
  },
  "max_time_span": "P90D"
}
```

- `max_time_span` - string. ISO-8601 interval.
- 217 - List contains nonexistent entities - when one or more of tracker IDs belong to nonexistent tracker (or to a tracker belonging to different user).
- 222 - Plugin not found - when specified report plugin not found.
- 236 - Feature unavailable due to tariff restrictions - when one of the trackers has tariff with disabled reports (`"has_reports"` is false).

## list

Returns info about all available generated or in-progress reports.

**required sub-user rights:** `reports`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

### HTTP GET

```
https://api.navixy.com/v2/report/tracker/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true, "list": [
    {
```

```

    "created": "2020-10-08 21:59:30",
    "time_filter": {
      "from": "00:00:00",
      "to": "23:59:59",
      "weekdays": [1, 2, 3, 4, 5, 6, 7]},
    "title": "Trip report",
    "id": 5601797,
    "parameters": {
      "geocoder": "google",
      "trackers": [669673],
      "plugins": [{
        "plugin_id": 4,
        "filter": true,
        "hide_empty_tabs": true,
        "show_coordinates": false,
        "split": true,
        "include_summary_sheet_only": false,
        "show_seconds": false,
        "group_by_driver": false,
        "show_idle_duration": false
      }],
      "locale_info": {
        "locale": "ru_RU",
        "time_zone": "Asia/Yekaterinburg",
        "measurement_system": "metric"
      }
    },
    "percent": 100,
    "type": "user",
    "from": "2020-10-05 00:00:00",
    "to": "2020-10-06 23:59:59"
  }
}]

```

- `created` - string. Date when report created.
- `time_filter` - object.
  - `from` - string. Control time "from" of day.
  - `to` - string. Control time "to" of day.
  - `weekdays` - int array. Control "weekdays" of the report. Can be 1 - 7.
- `title` - string. Report title.
- `id` - int. Report ID which can be used to retrieve or download report.
- `parameters` - object with report parameters.
  - `trackers` - int array. List of tracker IDs used for report.
  - `plugins` - array of objects. List of parameters for all plugins which were used to generate report.
  - `locale_info` - object with information about the locale, timezone, and measurement system used for the report.

- `percent` - int. Report readiness in percent.
- `type` - [enum](#). Type of created report.
- `from` - string. "from" parameter from generate.
- `to` - string. "to" parameter from generate.

#### errors

- [General](#) types only.

## retrieve

Retrieves a generated report as JSON.

**required sub-user rights:** `reports`.

#### parameters

| name      | description                            | type |
|-----------|--|------|
| report_id | ID of a report that should be deleted. | int  |

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/retrieve' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "report_id":  
1234567}'
```

##### HTTP GET

```
https://api.navixy.com/v2/report/tracker/retrieve?  
hash=a6aa75587e5c59c32d347da438505fc3&report_id=1234567
```



**response**

## Response

```
{
  "success": true,
  "report": {
    "created": "2020-10-06 16:01:46",
    "time_filter": {
      "from": "00:00:00",
      "to": "23:59:59",
      "weekdays": [
        1,
        2,
        3,
        4,
        5,
        6,
        7
      ]
    },
    "title": "Trip report",
    "id": 5602232,
    "sheets": [
      {
        "header": "Samantha (Ford Focus)",
        "sections": [
          {
            "data": [
              {
                "rows": [
                  {
                    "to": {
                      "v": "02:39 - Serpukhov,
Moscow Oblast, Russia, 142253",
                      "raw": 1601941188000.0,
                      "type": "value",
                      "location": {
                        "lat": 54.9218516,
                        "lng": 37.335545
                      }
                    },
                    "from": {
                      "v": "00:47 - Selyatino, Naro-
Fominskii gor. okrug, Moscow Oblast, Russia, 143370",
                      "raw": 1601934439000.0,
                      "type": "value",
                      "location": {
                        "lat": 55.5311083,
                        "lng": 36.96743
                      }
                    },
                    "time": {
                      "v": "01:52",
                      "raw": 6749.0,
                      "type": "value"
                    },
                    "length": {
                      "v": "106.29",
                      "raw": 106.29,
                      "type": "value"
                    }
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
  }
}
```

```

    },
    "avg_speed": {
      "v": "57",
      "raw": 57.0,
      "type": "value"
    },
    "max_speed": {
      "v": "94",
      "raw": 94.0,
      "type": "value"
    }
  },
  {
    "to": {
      "v": "05:10 - Selyatino, Naro-
Fominskii gor. okrug, Moscow Oblast, Russia, 143370",
      "raw": 1601950218000.0,
      "type": "value",
      "location": {
        "lat": 55.5308216,
        "lng": 36.967315
      }
    },
    "from": {
      "v": "03:11 - Serpukhov,
Moscow Oblast, Russia, 142253",
      "raw": 1601943083000.0,
      "type": "value",
      "location": {
        "lat": 54.9218116,
        "lng": 37.3354833
      }
    },
    "time": {
      "v": "01:58",
      "raw": 7135.0,
      "type": "value"
    },
    "length": {
      "v": "106.97",
      "raw": 106.97,
      "type": "value"
    },
    "avg_speed": {
      "v": "54",
      "raw": 54.0,
      "type": "value"
    },
    "max_speed": {
      "v": "94",
      "raw": 94.0,
      "type": "value"
    }
  },
  {
    "to": {
      "v": "07:54 - Khievskii
pereulok, 10, TNKh, Rassudovo, Troitsky Administrative Okrug,
Moscow, Russia, 143340",
      "raw": 1601960075000.0,
      "type": "value",

```

```

        "location": {
            "lat": 55.4666366,
            "lng": 36.9216966
        }
    },
    "from": {
        "v": "07:38 - Selyatino, Naro-
Fominskii gor. okrug, Moscow Oblast, Russia, 143370",
        "raw": 1601959081000.0,
        "type": "value",
        "location": {
            "lat": 55.53122,
            "lng": 36.9672916
        }
    },
    "time": {
        "v": "00:16",
        "raw": 994.0,
        "type": "value"
    },
    "length": {
        "v": "10.03",
        "raw": 10.03,
        "type": "value"
    },
    "avg_speed": {
        "v": "36",
        "raw": 36.0,
        "type": "value"
    },
    "max_speed": {
        "v": "85",
        "raw": 85.0,
        "type": "value"
    }
},
{
    "to": {
        "v": "09:36 - Serpukhov,
Moscow Oblast, Russia, 142253",
        "raw": 1601966165000.0,
        "type": "value",
        "location": {
            "lat": 54.926835,
            "lng": 37.3341066
        }
    },
    "from": {
        "v": "07:58 - Khievskii
pereulok, 10, TNKh, Rassudovo, Troitsky Administrative Okrug,
Moscow, Russia, 143340",
        "raw": 1601960315000.0,
        "type": "value",
        "location": {
            "lat": 55.46661,
            "lng": 36.9216516
        }
    },
    "time": {
        "v": "01:37",
        "raw": 5850.0,

```

```

        "type": "value"
    },
    "length": {
        "v": "95.31",
        "raw": 95.31,
        "type": "value"
    },
    "avg_speed": {
        "v": "59",
        "raw": 59.0,
        "type": "value"
    },
    "max_speed": {
        "v": "91",
        "raw": 91.0,
        "type": "value"
    }
},
{
    "to": {
        "v": "09:53 - Serpukhov,
Moscow Oblast, Russia, 142253",
        "raw": 1601967190000.0,
        "type": "value",
        "location": {
            "lat": 54.921935,
            "lng": 37.33551
        }
    },
    "from": {
        "v": "09:43 - Serpukhov,
Moscow Oblast, Russia, 142253",
        "raw": 1601966585000.0,
        "type": "value",
        "location": {
            "lat": 54.9264033,
            "lng": 37.3336633
        }
    },
    "time": {
        "v": "00:10",
        "raw": 605.0,
        "type": "value"
    },
    "length": {
        "v": "0.95",
        "raw": 0.95,
        "type": "value"
    },
    "avg_speed": {
        "v": "6",
        "raw": 6.0,
        "type": "value"
    },
    "max_speed": {
        "v": "13",
        "raw": 13.0,
        "type": "value"
    }
},
{

```

```

        "to": {
          "v": "12:36 - Selyatino, Naro-
Fominskii gor. okrug, Moscow Oblast, Russia, 143370",
          "raw": 1601977017000.0,
          "type": "value",
          "location": {
            "lat": 55.5309666,
            "lng": 36.9674183
          }
        },
        "from": {
          "v": "10:27 - Serpukhov,
Moscow Oblast, Russia, 142253",
          "raw": 1601969226000.0,
          "type": "value",
          "location": {
            "lat": 54.9219933,
            "lng": 37.335495
          }
        },
        "time": {
          "v": "02:09",
          "raw": 7791.0,
          "type": "value"
        },
        "length": {
          "v": "108.48",
          "raw": 108.48,
          "type": "value"
        },
        "avg_speed": {
          "v": "50",
          "raw": 50.0,
          "type": "value"
        },
        "max_speed": {
          "v": "89",
          "raw": 89.0,
          "type": "value"
        }
      },
      {
        "to": {
          "v": "16:01 - KhP \"Lesnoe
ozero\", Dernopol'e, gor. okrug Serpukhov, Moscow Oblast, Russia,
142279",
          "raw": 1601989300000.0,
          "type": "value",
          "location": {
            "lat": 54.9875133,
            "lng": 37.3093183
          }
        },
        "from": {
          "v": "13:34 - Selyatino, Naro-
Fominskii gor. okrug, Moscow Oblast, Russia, 143370",
          "raw": 1601980444000.0,
          "type": "value",
          "location": {
            "lat": 55.5309966,
            "lng": 36.96738
          }
        }
      }
    ]
  }
}

```

```

        },
        "time": {
            "v": "02:27",
            "raw": 8856.0,
            "type": "value"
        },
        "length": {
            "v": "95.79",
            "raw": 95.79,
            "type": "value"
        },
        "avg_speed": {
            "v": "39",
            "raw": 39.0,
            "type": "value"
        },
        "max_speed": {
            "v": "88",
            "raw": 88.0,
            "type": "value"
        }
    },
    ],
    "total": {
        "text": "In total:",
        "time": {
            "v": "10:33",
            "raw": 37980.0,
            "type": "value"
        },
        "length": {
            "v": "523.8",
            "raw": 523.8,
            "type": "value"
        },
        "avg_speed": {
            "v": "50",
            "raw": 50.0,
            "type": "value"
        },
        "max_speed": {
            "v": "94",
            "raw": 94.0,
            "type": "value"
        }
    },
    "header": "Oct 6, 2020 (Tue) : 7"
}
],
"type": "table",
"header": "Trips",
"columns": [
    {
        "align": "left",
        "field": "from",
        "title": "Movement start",
        "width": 4,
        "weight": 3,
        "highlight_min_max": false
    },

```

```

        {
            "align": "left",
            "field": "to",
            "title": "Movement end",
            "width": 4,
            "weight": 3,
            "highlight_min_max": false
        },
        {
            "align": "right",
            "field": "length",
            "title": "Total trips length,\nkm",
            "width": 1,
            "weight": 0,
            "highlight_min_max": false
        },
        {
            "align": "right",
            "field": "time",
            "title": "Travel time",
            "width": 1,
            "weight": 0,
            "highlight_min_max": false
        },
        {
            "align": "right",
            "field": "avg_speed",
            "title": "Average speed,\nkm/h",
            "width": 1,
            "weight": 0,
            "highlight_min_max": false
        },
        {
            "align": "right",
            "field": "max_speed",
            "title": "Max. speed,\nkm/h",
            "width": 1,
            "weight": 0,
            "highlight_min_max": false
        }
    ],
    "column_groups": []
},
{
    "rows": [
        {
            "v": "7",
            "raw": 7.0,
            "name": "Trips",
            "highlight": false
        },
        {
            "v": "523.8",
            "raw": 523.8,
            "name": "Total trips length, km",
            "highlight": false
        },
        {
            "v": "10:33",
            "raw": 633.0,
            "name": "Travel time",

```



```

        "highlight": false
      },
      {
        "v": "50",
        "raw": 50.0,
        "name": "Average speed, km/h",
        "highlight": false
      },
      {
        "v": "94",
        "raw": 94.0,
        "name": "Max. speed, km/h",
        "highlight": false
      },
      {
        "v": "515855",
        "raw": 515855.0,
        "name": "Odometer value *, km",
        "highlight": false
      }
    ],
    "type": "map_table",
    "header": "Summary"
  },
  {
    "text": "Odometer value at the end of the
selected period.",
    "type": "text",
    "style": "small_print"
  }
],
"entity_ids": [
  311852
],
"additional_field": ""
}
],
"from": "2020-10-06 00:00:00",
"to": "2020-10-06 23:59:59"
}

```

- `report` - object. Body of the generated report. Its contents are plugin-dependent.

## errors

- 204 - Entity not found - if report with the specified ID not found.
- 229 - Requested data is not ready yet - if report exists, but its generation is still in progress.

## status

Returns a report generation status for the specified report id.

**required sub-user rights:** reports.

#### parameters

| name      | description                            | type |
|-----------|--|------|
| report_id | ID of a report that should be deleted. | int  |

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/report/tracker/status' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "report_id": 1234567}'
```

##### HTTP GET

```
https://api.navixy.com/v2/report/tracker/status?
hash=a6aa75587e5c59c32d347da438505fc3&report_id=1234567
```

#### response

```
{
  "success": true,
  "percent_ready": 75
}
```

- percent\_ready - int. Report readiness in percent.

#### errors

- 204 - Entity not found - if report with the specified ID not found.

Last update: August 1, 2023





# Subuser

Contains API calls related to sub-users, that is, additional users who have access to your account and monitoring assets. Sub-users is a convenient way for corporate clients to provide multiple employees, who have different roles and privileges, with access to the monitoring system.

"Usual" user account called "master account" in relation to sub-users.

Every sub-user can operate on a subset of trackers from your "master account". Every entity, which is associated with unavailable trackers, also becomes hidden from sub-user. This is called "scoping". Sub-users' rights can also be limited to prevent unauthorized changes to your data and application setting.

NOTE: Sub-users cannot have any "exclusive" objects. Every tracker, rule, task, etc., even created or edited by sub-user, still belongs to your account. The only exception is reporting system: every sub-user has its own reports pool and reports schedule.

## Sub-user object structure

Sub-user object is almost identical to usual user.

```
{
  "id": 103,
  "activated": true,
  "login": "user@test.com",
  "first_name": "Charles",
  "middle_name": "Henry",
  "last_name": "Pearson",
  "legal_type": "legal_entity",
  "phone": "491761234567",
  "post_country": "Germany",
  "post_index": "61169",
  "post_region": "Hessen",
  "post_city": "Wiesbaden",
  "post_street_address": "Marienplatz 2",
  "registered_country": "Germany",
  "registered_index": "61169",
  "registered_region": "Hessen",
  "registered_city": "Wiesbaden",
  "registered_street_address": "Marienplatz 2",
  "state_reg_num": "12-3456789",
  "tin": "1131145180",
  "legal_name": "E. Biasi GmbH",
  "iec": "",
  "security_group_id": 333,
```

```
    "creation_date": "2016-05-20 00:00:00"  
  }
```

- `id` - int. Sub-user's ID, can be null (when creating new sub-user).
- `activated` - boolean. `true` if sub-user activated (allowed to log in).
- `login` - string. Sub-user email as login. Must be valid unique email address.
- `first_name` - string. Sub-user's or contact person first name.
- `middle_name` - string. Sub-user's or contact person middle name.
- `last_name` - string. Sub-user's or contact person last name.
- `legal_type` - [enum](#). Can be "legal\_entity", "individual" or "sole\_trader".
- `phone` - string. Sub-user's or contact phone (10-15 digits).
- `post_country` - string. Country part of sub-user's post address.
- `post_index` - string. Index part of sub-user's post address.
- `post_region` - string. Region part of sub-user's post address.
- `post_city` - string. City from postal address.
- `post_street_address` - string. Street address.
- `registered_country` - string. Country part of sub-user's registered address.
- `registered_index` - string. Index part of sub-user's registered address.
- `registered_region` - string. Region part of sub-user's registered address.
- `registered_city` - string. City from registered address.
- `registered_street_address` - string. Sub-user's registered address.
- `state_reg_num` - string. State registration number. E.g. EIN in the USA, OGRN in Russia. 15 characters max.
- `tin` - string. Taxpayer identification number aka "VATIN" or "INN".
- `legal_name` - string. Sub-user's legal name (for "legal\_entity" only).
- `iec` - optional string. Industrial Enterprises Classifier aka "KPP" (used in Russia. For "legal\_entity" only).
- `security_group_id` - int. An ID of the security group to which sub-user belongs to. Can be null, which means default group with no privileges.
- `creation_date` - [date/time](#). Date and time when sub-user was created. This field is read-only, it should not be used in subuser/update.

## API actions

API path: `/subuser`.

### delete

Deletes sub-user. This operation cannot be reversed.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

### parameters

| name                    | description                                      | type |
|-------------------------|--|------|
| <code>subuser_id</code> | ID of the sub-user belonging to current account. | int  |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/delete' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "subuser_id": 123567}'
```

#### HTTP GET

```
https://api.navixy.com/v2/subuser/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&subuser_id=123567
```

### response

```
{  
  "success": true  
}
```

### errors

- 13 – Operation not permitted – if user has insufficient rights.
- 236 – Feature unavailable due to tariff restrictions - if there is at least one tracker without `multilevel_access` tariff feature.
- 201 – Not found in the database – if sub-user with such an ID does not exist or does not belong to current master user.

## list

List all sub-users belonging to current user.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

### parameters

Only API key `hash`.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/list' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/subuser/list?  
hash=a6aa75587e5c59c32d347da438505fc3
```

### response

```
{  
  "success": true,  
  "list": [{  
    "id": 103,  
    "activated": true,  
    "login": "user@test.com",  
    "first_name": "Charles",  
    "middle_name": "Henry",  
    "last_name": "Pearson",  
    "legal_type": "legal_entity",  
    "phone": "491761234567",  
    "post_country": "Germany",  
    "post_index": "61169",  
    "post_region": "Hessen",  
    "post_city": "Wiesbaden",  
    "post_street_address": "Marienplatz 2",  
    "registered_country": "Germany",  
    "registered_index": "61169",  
    "registered_region": "Hessen",  
    "registered_city": "Wiesbaden",  
    "registered_street_address": "Marienplatz 2",  
    "state_reg_num": "12-3456789",  
    "tin": "1131145180",  
    "legal_name": "E. Biasi GmbH",  
    "iec": "",  
    "security_group_id": 333,  
    "creation_date": "2016-05-20 00:00:00"
```



```
}]
}
```

- `list` - array of objects. List of all sub-users belonging to this master account.

Sub-user object described [here](#).

### errors

- 13 – Operation not permitted – if user has insufficient rights.
- 236 – Feature unavailable due to tariff restrictions - if there is at least one tracker without `multilevel_access` tariff feature.

## register

Allows you to create sub-users associated to your master account.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

### parameters

| name     | description  | type        |
|----------|--|-------------|
| user     | <code>subuser</code> object without <code>id</code> field. | JSON object |
| password | New sub-user's password. 6 to 20 characters.               | string      |

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/register' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "password":
123456, "user": {"activated": true, "login": "user@test.com",
"first_name": "Charles", "middle_name": "Henry", "last_name":
"Pearson", "legal_type": "legal_entity", "phone": "491761234567",
"post_country": "Germany", "post_index": "61169", "post_region":
"Hessen", "post_city": "Wiesbaden", "post_street_address":
"Marienplatz 2", "registered_country": "Germany",
"registered_index": "61169", "registered_region": "Hessen",
"registered_city": "Wiesbaden", "registered_street_address":
"Marienplatz 2", "state_reg_num": "12-3456789", "tin":
"1131145180", "legal_name": "E. Biasi GmbH", "iec": "",
"security_group_id": 333}}'
```

## response

```
{
  "success": true,
  "id": 121458
}
```

- `id` - int. An ID of the created sub-user.

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 236 – Feature unavailable due to tariff restrictions - if there is at least one tracker without `multilevel_access` tariff feature.
- 201 – Not found in the database – when specified `security_group_id` does not exist.
- 206 – login already in use - if this login email already registered.

## update

Updates sub-user data.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

## parameters

| name | description   | type        |
|------|---|-------------|
| user | <code>subuser</code> object with <code>id</code> field. | JSON object |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "user":
{"id": 123451, "activated": true, "login": "user@test.com",
"first_name": "Charles", "middle_name": "Henry", "last_name":
"Pearson", "legal_type": "legal_entity", "phone": "491761234567",
"post_country": "Germany", "post_index": "61169", "post_region":
"Hessen", "post_city": "Wiesbaden", "post_street_address":
"Marienplatz 2", "registered_country": "Germany",
"registered_index": "61169", "registered_region": "Hessen",
"registered_city": "Wiesbaden", "registered_street_address":
"Marienplatz 2", "state_reg_num": "12-3456789", "tin":
"1131145180", "legal_name": "E. Biasi GmbH", "iec": "",
"security_group_id": 333}}'
```

## response

```
{
  "success": true
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 236 – Feature unavailable due to tariff restrictions - if there is at least one tracker without `multilevel_access` tariff feature.
- 201 – Not found in the database – if sub-user with such an ID does not exist or does not belong to current master user. Also, when specified `security_group_id` does not exist.

Last update: June 28, 2023





# Subuser places

Contains API calls to control which places is available to which sub-user.

## API actions

API path: `/subuser/places`.

### bind

Gives access for sub-user to specified places.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

### parameters

| name                       | description  | type      |
|----------------------------|--|-----------|
| <code>subuser_id</code>    | ID of a sub-user belonging to current account.   | int       |
| <code>access_to_all</code> | Optional. If <code>true</code> then sub-user will have access to all places of master user.                        | boolean   |
| <code>place_ids</code>     | Optional. List of place IDs to associate with a specified sub-user. All places must belong to current master user. | int array |

■ At least one of `access_to_all` and `place_ids` parameters must be not null.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/places/bind' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "subuser_id": 204951, "access_to_all": false, "place_ids": [7548]}'
```

## response

```
{
  "success": true
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 201 – Not found in the database – if sub-user/place does not exist or does not belong to current master user.
- 236 – Feature unavailable due to tariff restrictions (if there is at least one tracker without `multilevel_access` tariff feature).

## unbind

Disables access for a sub-user to specified places.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

## parameters

| name       | description  | type      |
|------------|--|-----------|
| subuser_id | ID of a sub-user belonging to current account.   | int       |
| place_ids  | List of place IDs to associate with a specified sub-user. All places must belong to current master user. | int array |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/places/unbind' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "subuser_id": 204951, "place_ids": [7548]}'
```

## response

```
{
  "success": true
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 201 – Not found in the database – if sub-user/place not exist or does not belong to current master user.
- 236 – Feature unavailable due to tariff restrictions (if there is at least one tracker without `multilevel_access` tariff feature).

## list\_ids

Gets a list of place IDs to which this sub-user has access.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

## parameters

| name       | description                                    | type |
|------------|--|------|
| subuser_id | ID of a sub-user belonging to current account. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/places/list_ids' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "subuser_id": 204951}'
```

## response

```
{
  "success": true,
  "access_to_all": true,
  "list" : [7548]
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 201 – Not found in the database – if sub-user with such an ID does not exist or does not belong to current master user.
- 236 – Feature unavailable due to tariff restrictions (if there is at least one tracker without `multilevel_access` tariff feature).



## list

Gets a list of places to which this sub-user has access.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

### parameters

| name       | description   | type      |
|------------|---|-----------|
| subuser_id | ID of a sub-user belonging to current account.  | int       |
| filter     | Optional. Filter for place label, description, address, external ID and custom fields.  | string    |
| tag_ids    | Optional. Tag IDs assigned to places. Places found must include all tags from a list.   | int array |
| offset     | Optional. Offset from start of found places for pagination.   | int       |
| limit      | Optional. Limit of found places for pagination.   | int       |
| order      | Optional. Specify list ordering. Can be any of <code>id</code> , <code>label</code> , <code>description</code> , <code>location</code> , <code>external_id</code> , <code>assigned_date</code> . Default order by <code>id</code> . | enum      |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/places/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "subuser_id": 204951, "offset": 0, "limit": 1000}'
```

### response

```
{
  "success": true,
  "access_to_all": false,
  "list" : [<place>, ...],
  "count": 12
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 201 – Not found in the database – if sub-user with such an ID does not exist or does not belong to current master user.
- 236 – Feature unavailable due to tariff restrictions (if there is at least one tracker without `multilevel_access` tariff feature).

Last update: December 26, 2023





# Subuser security group

Contains security group object structure and API calls related to security groups, that is, groups of sub-users with the specified set of rights and privileges.

## Security group object structure

```
{
  "id": 103,
  "label": "Managers",
  "privileges": {
    "rights": ["tag_update", "tracker_register"],
    "store_period": "1d"
  }
}
```

- `id` - int. Group id, can be null (when creating new security group).
- `label` - string. Group label.
- `privileges` - object containing privileges of group.
  - `rights` - string array. A set of rights granted to security group (see below).
  - `store_period` - optional string. Period of viewing history in legacy duration format, e.g. "2h" (2 hours), "3d" (3 days), "5m" (5 months), "1y" (one year).

## Default security group

Default (or empty) security group is the group which is effective when sub-users' `security_group_id` is null. It has empty `rights` array.

## Master user's rights

Master user always has all rights, including exclusive "admin" right.

## Security group rights

Absolute majority of read operations does not require any rights (that is, they are available to all sub-users, even with "null" security group). However, some entities may

be hidden because they are associated with the trackers unavailable to sub-user. Most of data-modifying operations, on the contrary, require some rights to be present.

Possible rights are:

- admin – master user-only. Can't be assigned to security groups,
- tracker\_update,
- tracker\_register,
- tracker\_rule\_update,
- tracker\_configure,
- tracker\_set\_output,
- tag\_update,
- task\_update,
- zone\_update,
- place\_update,
- employee\_update,
- vehicle\_update,
- payment\_create
- form\_template\_update,
- reports,
- checkin\_update.

## API actions

API path: `/subuser/security_group/`.

### create

Creates new security group.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

## parameters

| name  | description  | type        |
|-------|--|-------------|
| group | <code>security_group</code> object without "id" field. | JSON object |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/security_group/create' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "group":   
{"label": "Managers", "privileges": {"rights": ["tag_update",   
"tracker_register"], "store_period": "1d"}}}'
```

## response

```
{  
  "success": true,  
  "id": 103  
}
```

- `id` - int. An ID of the created security group.

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 236 – Feature unavailable due to tariff restrictions - if there is at least one tracker without `multilevel_access` tariff feature.

## delete

Deletes existing security group. All sub-users belonging to this group will be assigned to default (null) security group.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

## parameters

| name              | description                                  | type |
|-------------------|--|------|
| security_group_id | ID of security group, which must be deleted. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/security_group/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id": 103}'
```

### HTTP GET

```
https://api.navixy.com/v2/subuser/security_group/delete?
hash=a6aa75587e5c59c32d347da438505fc3&id=103
```

## response

```
{
  "success": true
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 201 – Not found in the database – when group with the specified security\_group\_id does not exist.
- 236 – Feature unavailable due to tariff restrictions - if there is at least one tracker without multilevel\_access tariff feature.

## list

List all security groups belonging to current user.

**required tariff features:** multilevel\_access – for ALL trackers. **required sub-user rights:** admin (available only to master users).

## parameters

Only API key hash.



## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/security_group/
list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/subuser/security_group/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [{
    "id": 103,
    "label": "Managers",
    "privileges": {
      "rights": ["tag_update", "tracker_register"],
      "store_period": "1d"
    }
  }]
}
```

- `list` - array of objects. List of all security groups belonging to this master account.

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 236 – Feature unavailable due to tariff restrictions (if there is at least one tracker without `multilevel_access` tariff feature).

## update

Updates existing security group.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

## parameters

| name  | description                                  | type        |
|-------|--|-------------|
| group | <code>security_group</code> with "id" field. | JSON object |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/security_group/
update' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "group":
{"id": 103, "label": "Managers", "privileges": {"rights":
["tag_update", "tracker_register"], "store_period": "1d"}}}'
```

## response

```
{
  "success": true
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 201 – Not found in the database – when security group with the specified ID does not exist.
- 236 – Feature unavailable due to tariff restrictions - if there is at least one tracker without `multilevel_access` tariff feature.

## assign

Assigns (removes) a security group to sub-users.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

## parameters

| name        | description                      | type      |
|-------------|----------------------------------|-----------|
| group_id    | Nullable, ID of a security group | int       |
| subuser_ids | IDs of sub-users                 | int array |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/security_group/assign' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "group_id": 3, subuser_ids: [12, 34]}'
```

## response

```
{
  "success": true
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 201 – Not found in the database – when security group with the specified ID does not exist.
- 236 – Feature unavailable due to tariff restrictions - if there is at least one tracker without `multilevel_access` tariff feature.

Last update: December 26, 2023





# Subuser session

Sub-user session actions to obtain its hash.

## API actions

API path: `/subuser/session/`.

### create

Creates a new session for the specified sub-user and obtain its hash. Can be used to log in to sub-user's accounts.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

### parameters

| name       | description                                      | type |
|------------|--|------|
| subuser_id | ID of the sub-user belonging to current account. | int  |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/session/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "subuser_id": 204951}'
```

#### HTTP GET

```
https://api.navixy.com/v2/subuser/session/create?
hash=a6aa75587e5c59c32d347da438505fc3&subuser_id=204951
```

### response

```
{
  "success": true,
```

```
"hash" : "22eac1c27af4be7b9d04da2ce1af111b"
}
```

- `hash` - string. Hash of the created sub-user session.

#### errors

- 13 – Operation not permitted – if user has insufficient rights.
- 236 – Feature unavailable due to tariff restrictions - if there is at least one tracker without `multilevel_access` tariff feature.
- 201 – Not found in the database – if sub-user with such an ID does not exist or does not belong to current master user.

Last update: December 26, 2022







# Subuser tracker

Contains API calls to control which tracker is available to which sub-user.

## API actions

API path: `/subuser/tracker`.

### bind

Gives access for sub-user to the specified trackers.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

### parameters

| name       | description   | type         |
|------------|---|--------------|
| subuser_id | ID of the sub-user belonging to current account.  | int          |
| trackers   | List of tracker IDs to associate with the specified sub-user.<br>All trackers must belong to current master user. | int<br>array |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/tracker/bind' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "subuser_id":  
204951, "trackers": [127830]}'
```

#### HTTP GET

```
https://api.navixy.com/v2/subuser/tracker/bind?  
hash=a6aa75587e5c59c32d347da438505fc3&subuser_id=204951&trackers=[1278
```

### response

```
{  
  "success": true  
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 236 – Feature unavailable due to tariff restrictions - if there is at least one tracker without `multilevel_access` tariff feature.
- 201 – Not found in the database – if sub-user with such an ID does not exist or does not belong to current master user.
- 262 – Entries list is missing some entries or contains nonexistent entries – if one or more of specified tracker IDs don't exist.

## list

Gets a list of tracker IDs to which this sub-user has access.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

## parameters

| name       | description                                      | type |
|------------|--|------|
| subuser_id | ID of the sub-user belonging to current account. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/tracker/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "subuser_id": 204951}'
```

### HTTP GET

```
https://api.navixy.com/v2/subuser/tracker/list?  
hash=a6aa75587e5c59c32d347da438505fc3&subuser_id=204951
```

## response

```
{  
  "success": true,  
  "list" : [124588]  
}
```

- `list` - int array. List of tracker IDs to which this sub-user has access.

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 236 – Feature unavailable due to tariff restrictions - if there is at least one tracker without `multilevel_access` tariff feature.
- 201 – Not found in the database – if sub-user with such an ID does not exist or does not belong to current master user.

## unbind

Disables access for sub-user to the specified trackers.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

## parameters

| name       | description   | type         |
|------------|---|--------------|
| subuser_id | ID of the sub-user belonging to current account.  | int          |
| trackers   | List of tracker IDs to associate with the specified sub-user.<br>All trackers must belong to current master user. | int<br>array |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/tracker/unbind' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "subuser_id":  
204951, "trackers": [127830]}'
```

### HTTP GET

```
https://api.navixy.com/v2/subuser/tracker/unbind?  
hash=a6aa75587e5c59c32d347da438505fc3&subuser_id=204951&trackers=[1278
```

## response

```
{  
  "success": true  
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 236 – Feature unavailable due to tariff restrictions (if there is at least one tracker without `multilevel_access` tariff feature).
- 201 – Not found in the database – if sub-user with such an ID does not exist or does not belong to current master user.
- 262 – Entries list is missing some entries or contains nonexistent entries – if one or more of specified tracker IDs don't exist.

Last update: January 15, 2023





# Subuser geofences

Contains API calls to control which geofences is available to which sub-user.

## API actions

API path: `/subuser/zones`.

### bind

Gives access for sub-user to specified geofences.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

### parameters

| name                       | description  | type         |
|----------------------------|--|--------------|
| <code>subuser_id</code>    | ID of a sub-user belonging to current account.   | int          |
| <code>access_to_all</code> | Optional. If <code>true</code> then sub-user will have access to all geofences of master user.                           | boolean      |
| <code>zone_ids</code>      | Optional. List of geofence IDs to associate with a specified sub-user. All geofences must belong to current master user. | int<br>array |

■ At least one of `access_to_all` and `zone_ids` parameters must be not null.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/zones/bind' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "subuser_id":  
204951, "access_to_all": false, "zone_ids": [7548]}'
```



## response

```
{
  "success": true
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 201 – Not found in the database – if sub-user/geofence does not exist or does not belong to current master user.
- 236 – Feature unavailable due to tariff restrictions (if there is at least one tracker without `multilevel_access` tariff feature).

## unbind

Disables access for sub-user to specified geofences.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

## parameters

| name       | description   | type         |
|------------|---|--------------|
| subuser_id | ID of a sub-user belonging to current account.  | int          |
| zone_ids   | List of geofence IDs to associate with a specified sub-user.<br>All geofences must belong to current master user. | int<br>array |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/zones/unbind' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "subuser_id": 204951, "zone_ids": [7548]}'
```

## response

```
{
  "success": true
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 201 – Not found in the database – if sub-user/geofence not exist or does not belong to current master user.
- 236 – Feature unavailable due to tariff restrictions (if there is at least one tracker without `multilevel_access` tariff feature).

## list\_ids

Gets a list of geofence IDs to which this sub-user has access.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

## parameters

| name       | description                                    | type |
|------------|--|------|
| subuser_id | ID of a sub-user belonging to current account. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/zones/list_ids' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "subuser_id": 204951}'
```

## response

```
{
  "success": true,
  "access_to_all": true,
  "list" : [7548]
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 201 – Not found in the database – if sub-user with such an ID does not exist or does not belong to current master user.
- 236 – Feature unavailable due to tariff restrictions (if there is at least one tracker without `multilevel_access` tariff feature).

## list

Gets a list of geofences to which this sub-user has access.

**required tariff features:** `multilevel_access` – for ALL trackers. **required sub-user rights:** `admin` (available only to master users).

### parameters

| name       | description  | type      |
|------------|--|-----------|
| subuser_id | ID of a sub-user belonging to current account.   | int       |
| filter     | Optional. Filter for geofence label.   | string    |
| tag_ids    | Optional. Tag IDs assigned to geofences. Geofences found must include all tags from a list.                              | int array |
| offset     | Optional. Offset from start of found geofences for pagination.   | int       |
| limit      | Optional. Limit of found geofences for pagination.   | int       |
| order      | Optional. Specify list ordering. Can be any of <code>id</code> , <code>label</code> . Default order by <code>id</code> . | enum      |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/subuser/zones/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "subuser_id": 204951, "offset": 0, "limit": 1000}'
```

### response

```
{  
  "success": true,  
  "access_to_all": false,  
  "list" : [<zone>, ...],  
  "count": 12  
}
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 201 – Not found in the database – if sub-user with such an ID does not exist or does not belong to current master user.
- 236 – Feature unavailable due to tariff restrictions (if there is at least one tracker without `multilevel_access` tariff feature).

Last update: December 26, 2023





# Tag

Tag is a label, or a key word that is used for a quick and easy search. They help find the needed places, geofences, employees, tasks, trackers, and vehicles. Contains tag object and API calls to interact with it.

Find more information about tags API usage in our [how-tos](#).

## Tag object

```
{
  "id": 3,
  "avatar_file_name": "avatar.jpg",
  "name": "hop",
  "color": "FF0000"
}
```

- `id` - int. Tag ID.
- `avatar_file_name` - optional string. File name with extension.
- `name` - string. Tag's name.
- `color` - string. Tag color in 3-byte RGB hex format.

### tagged entity types

- place
- task
- task\_schedule
- employee
- vehicle
- zone
- tracker

## API actions

API path: `/tag`.

## create

Creates a new tag.

**required sub-user rights:** `tag_update`.

### parameters

| name | description                               | type        |
|------|---|-------------|
| tag  | Tag object without <code>id</code> field. | JSON object |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tag/create' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tag":  
{"name": "hop", "color": "FF0000"}}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tag/create?  
hash=a6aa75587e5c59c32d347da438505fc3&tag={"name": "hop", "color":  
"FF0000"}
```

### response

```
{  
  "success": true,  
  "id": 111  
}
```

- `id` - int. An ID of the created tag.

### errors

[General](#) types only.

## delete

Deletes tag with the specified ID.

**required sub-user rights:** `tag_update`.



## parameters

To delete tags, only one of the following parameters must be specified.

| name    | description                    | type      |
|---------|--------------------------------|-----------|
| tag_id  | ID of the tag to delete.       | int       |
| tag_ids | An array of tag IDs to delete. | int array |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tag/delete' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tag_id": 1}'
```

### HTTP GET

```
https://api.navixy.com/v2/tag/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&tag_id=1
```

## response

```
{  
  "success": true  
}
```

## errors

- 201 – Not found in the database - if there is no tag with such an ID. This error will not occur if the tag\_ids parameter is specified, deletion is silent in this case.

## list

Gets all tags belonging to user with optional filtering.

## parameters

| name   | description  | type   |
|--------|--|--------|
| filter | Optional filter for tag name. 3-60 characters or null. | string |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tag/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tag/list?  
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{  
  "success": true,  
  "list": [{  
    "id": 3,  
    "avatar_file_name": "avatar.jpg",  
    "name": "hop",  
    "color": "FF0000"  
  }]  
}
```

## errors

[General](#) types only.

## search

Search entities that bound with all of specified tags.

### parameters

| name         | description  | type         |
|--------------|--|--------------|
| tag_ids      | List of tag IDs to search.                                       | int array    |
| entity_types | Optional. List of <a href="#">tagged entity types</a> to filter. | string array |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tag/search' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tag_ids":  
[1, 2, 3]}'
```

### HTTP GET

```
https://api.navixy.com/v2/tag/search?  
hash=a6aa75587e5c59c32d347da438505fc3&tag_ids=[1, 2, 3]
```

## response

```
{  
  "success": true,  
  "result": {  
    "place": [<place>],  
    "task": [<task>],  
    "task_schedule": [<task_schedule>],  
    "employee": [<employee>],  
    "vehicle": [<vehicle>],  
    "zone": [<zone>],  
    "tracker": [<tracker>]  
  }  
}
```

- `place` - array of objects. List of place objects.
- `task` - array of objects. List of task objects.
- `task_schedule` - array of objects. List of task\_schedule objects.
- `employee` - array of objects. List of employee objects.
- `vehicle` - array of objects. List of vehicle objects.
- `zone` - array of objects. List of zone objects.
- `tracker` - array of objects. List of tracker objects.

## errors

[General](#) types only.

## update

Updates existing tag.

**required sub-user rights:** `tag_update`.

## parameters

| name | description                            | type        |
|------|--|-------------|
| tag  | Tag object with <code>id</code> field. | JSON object |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tag/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tag": {"id": 3, "name": "hop", "color": "FF0000"}}'
```

### HTTP GET

```
https://api.navixy.com/v2/tag/update?
hash=a6aa75587e5c59c32d347da438505fc3&tag={"id": 3, "name": "hop",
"color": "FF0000"}
```

## response

```
{
  "success": true
}
```

## errors

- 201 – Not found in the database - if there is no tag with such an ID.

Last update: August 1, 2023





# Tag avatar

Contains API calls to interact with tag avatars.

## API actions

API path: `/tag/avatar`.

### assign

Assigns icon\_id (from standard icon set) to specified tag.

**required sub-user rights:** `tag_update`.

### parameters

| name    | description  | type |
|---------|--|------|
| tag_id  | ID of the tag to assign.   | int  |
| icon_id | Icon to assign to tag. Can be null – this means that uploaded avatar should be used instead of icon. | int  |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tag/avatar/assign' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tag_id": 1, \  
    "icon_id": 14}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tag/avatar/assign? \  
hash=a6aa75587e5c59c32d347da438505fc3&tag_id=1&icon_id=14
```

### response

```
{  
  "success": true  
}
```

## errors

- 201 – Not found in the database - when vehicle with specified `tag_id` not found.

## upload

Uploads avatar image for specified tag.

Then it will be available from `[api_base_url]/[api_static_path]/tag/avatars/<file_name>`

e.g. `https://api.navixy.com/v2/static/tag/avatars/abcdef123456789.png`.

**required sub-user rights:** `tag_update`.

**avatar\_file\_name** returned in response and will be returned from [/tag/list](#).

**MUST** be a POST multipart request (multipart/form-data), with one of the parts being an image file upload (with the name `file`).

File part **mime** type must be one of:

- `image/jpeg`
- `image/pjpeg`
- `image/png`
- `image/gif`
- `image/webp`

## parameters

| name            | description   |
|-----------------|---|
| tag_id          | ID of the tag to upload.  |
| file            | Image file.   |
| redirect_target | Optional. URL to redirect. If <code>redirect_target</code> passed return redirect to <code>&lt;redirect_target&gt;?response=&lt;urlencoded response json&gt;</code> |

## response

```
{  
  "success": true,  
}
```



```
"value": "avatar.jpg"
}
```

- `value` - string. Avatar file name.

#### errors

- 201 – Not found in the database - when tag with specified `tag_id` not found.
- 233 – No data file - if `file` part not passed.
- 234 – Invalid data format - if passed `file` with unexpected `mime` type.
- 254 – Cannot save file - on some file system errors.

Last update: December 26, 2022





# User

A user account lets you start working with the platform as well as customize your experience within it. Contains user object structure and API calls to interact with users.

## User object structure

```
{
  "success": true,
  "paas_id": 7,
  "paas_settings": <paas_settings>,
  "user_info": {
    "id": 43568,
    "login": "demo@navixy.com",
    "title": "John Smith",
    "phone": "79123456789",
    "creation_date": "2016-05-20 01:10:34",
    "balance": 74.31,
    "bonus": 0,
    "locale": "en-US",
    "demo": true,
    "verified": true,
    "legal_type": "individual",
    "default_geocoder": "google",
    "route_provider": "google",
    "time_zone": "America/New_York",
    "measurement_system": "metric",
    "tin": "2345678239",
    "iec": "",
    "post_country": "USA",
    "post_region": "NY",
    "post_index": "10120",
    "post_city": "New York",
    "post_street_address": "1556 Broadway, suite 416",
    "registered_country": "USA",
    "registered_region": "NY",
    "registered_index": "10120",
    "registered_city": "New York",
    "registered_street_address": "1556 Broadway, suite 416",
    "first_name": "John",
    "middle_name": "Walker",
    "last_name": "Smith",
    "legal_name": "QWER Inc."
  },
  "master": {
    "id": 1234,
    "demo": false,
    "legal_type": "individual",
    "first_name": "David",
```

```

        "middle_name": "Middle",
        "last_name": "Blane",
        "legal_name": "Blah LLC",
        "title": "David Blane",
        "balance": 0.0,
        "bonus": 89.78
    },
    "tariff_restrictions": {
        "allowed_maps": ["roadmap", "osm"]
    },
    "premium_gis": true,
    "features": ["branding_web"],
    "privileges": {
        "rights": ["tag_update"]
    }
}

```

- `paas_id` - int. Dealer ID.
- `paas_settings` - object. The same as `settings` in [/dealer/get\\_ui\\_config](#) response.
- `user_info` - object. Info about user.
  - `id` - int. User ID.
  - `login` - string. User's login (in most cases it's an email address).
  - `title` - string. User first and last name or organization title.
  - `phone` - string. User phone (if not empty).
  - `creation_date` - [date/time](#). User registration date/time.
  - `balance` - float. User balance, max. 2 digits after dot. For sub-users, this field should be ignored.
  - `bonus` - float. User bonus, max. 2 digits after dot. For sub-users, this field should be ignored.
  - `locale` - [enum](#). User locale, for example "en\_EN".
  - `demo` - boolean. `true` if this is a demo user, `false` otherwise.
  - `verified` - boolean. `true` if user email already verified.
  - `legal_type` - [enum](#). Can be "legal\_entity", "individual" or "sole\_trader".
  - `default_geocoder` - [enum](#). User's default geocoder. Can be "google", "yandex", "progorod", "osm", or "locationiq".
  - `route_provider` - [enum](#). User's route provider. Can be "progorod", "google" or "osrm".
  - `time_zone` - [enum](#). User timezone name.
  - `measurement_system` - [enum](#). User's measurement system "metric", "imperial", "us", "metric\_gal\_us" or "nautical".
  - `tin` - string. Taxpayer identification number aka "VATIN" or "INN".

- `iec` - optional string. Industrial Enterprises Classifier aka "KPP". Used in Russia for legal entities.
- `post_country` - string. Country part of user's post address.
- `post_index` - string. Post index or ZIP code.
- `post_region` - string. Region part of post address (oblast, state, etc.).
- `post_city` - string. City from postal address.
- `post_street_address` - string. Street address.
- `registered_country` - string. Country part of user's registered address.
- `registered_index` - string. Index part of user's registered address.
- `registered_region` - string. Region part of user's registered address.
- `registered_city` - string. City from registered address.
- `registered_street_address` - string. User's registered address.
- `first_name` - string. User's or contact person first name.
- `middle_name` - string. User's or contact person middle name.
- `last_name` - string. User's or contact person last name.
- `legal_name` - optional string. A juridical name.
- `master` - object. Returned only if current user is sub-user. All fields have same meaning as in "user\_info", but for master user's account.
- `tariff_restrictions` - tariff restrictions object, for more info see [user/get\\_tariff\\_restrictions](#).
  - `allowed_maps` - string array. List of allowed maps.
- `premium_gis` - boolean. `true` if a dealer has premium GIS tariff.
- `features` - string array. Set of allowed [Dealer features](#).
- `privileges` - object only returned for sub-users. Describes effective sub-user privileges.
- `rights` - string array. A set of rights granted to sub-user. Described in [security group rights](#).

## API actions

API path: `/user`.

## activate

Activates previously registered user with the provided session hash (it is contained in activation link from email sent to user). Available only to master users.

### Attention

This call will receive only session hash from registration email. Any other hash will result in result error code 4 (User or API key not found or session ended). The only thing that API calls with a user session will work for is creating, reading, and deleting API keys.

## response

```
{ "success": true }
```

## auth

Tries to authenticate user and get hash.

It does not need authentication/hash and is available at `UNAUTHORIZED` access level.

 We recommend using [API keys](#) instead of user session hash.

## parameters

| name      | description  | type   | restrictions                            |
|-----------|--|--------|---|
| login     | User email as login (or demo login).   | string | not null.                               |
| password  | User password.   | string | not null, 1 to 40 printable characters. |
| dealer_id | If specified, API will check that user belongs to this dealer, and if not, error 102 will be returned. | int    | optional.                               |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/auth' \
-H 'Content-Type: application/json' \
-d '{"login": "user@email.com", "password": "12@14Y$"}'
```

## response

```
{
  "success": true,
  "hash": "22eac1c27af4be7b9d04da2ce1af111b"
}
```

- `hash` - string. Session hash.

## errors

- 11 – Access denied - if dealer blocked.
- 102 – Wrong login or password.
- 103 – User not activated.
- 104 – Logins limit exceeded, please reuse existing sessions instead (see also user/session/renew).
- 105 – Login attempts limit exceeded, try again later.

## get\_info

Gets user information and some settings.

## parameters

| name        | description  | type   | restrictions |
|-------------|--|--------|--------------|
| application | If specified, the response will contain a description of the application's custom menu | string | optional     |



## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/get_info' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/user/get_info?  
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{  
  "success": true,  
  "paas_id": 7,  
  "paas_settings": <paas_settings>,  
  "user_info": {  
    "id": 43568,  
    "login": "demo@navixy.com",  
    "title": "John Smith",  
    "phone": "79123456789",  
    "creation_date": "2016-05-20 01:10:34",  
    "balance": 74.31,  
    "bonus": 0,  
    "locale": "en-US",  
    "demo": true,  
    "verified": true,  
    "legal_type": "individual",  
    "default_geocoder": "google",  
    "route_provider": "google",  
    "time_zone": "America/New_York",  
    "measurement_system": "metric",  
    "tin": "2345678239",  
    "iec": "",  
    "post_country": "USA",  
    "post_region": "NY",  
    "post_index": "10120",  
    "post_city": "New York",  
    "post_street_address": "1556 Broadway, suite 416",  
    "registered_country": "USA",  
    "registered_region": "NY",  
    "registered_index": "10120",  
    "registered_city": "New York",  
    "registered_street_address": "1556 Broadway, suite 416",  
    "first_name": "John",  
    "middle_name": "Walker",  
    "last_name": "Smith",  
    "legal_name": "QWER Inc."  
  },  
  "master": {  
    "id": 1234,  
    "demo": false,  
    "legal_type": "individual",  
    "first_name": "David",  
  },  
}
```

```

        "middle_name": "Middle",
        "last_name": "Blane",
        "legal_name": "Blah LLC",
        "title": "David Blane",
        "balance": 0.0,
        "bonus": 89.78
    },
    "tariff_restrictions": {
        "allowed_maps": ["roadmap", "osm"]
    },
    "premium_gis": true,
    "features": ["branding_web"],
    "privileges": {
        "rights": ["tag_update"]
    },
    "menu": <customizable_user_menu>
}

```

- `user_object` - for more info see [user object structure](#).

## errors

- [General](#) types only.

## get\_tariff\_restrictions

Gets user tariff restrictions.

### parameters

Only API key `hash`.

### examples

#### cURL

```

curl -X POST 'https://api.navixy.com/v2/user/
get_tariff_restrictions' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'

```

#### HTTP GET

```

https://api.navixy.com/v2/user/get_tariff_restrictions?
hash=a6aa75587e5c59c32d347da438505fc3

```

### response

```

{
    "success": true,
    "value": {
        "allowed_maps": ["roadmap", "osm"]
    }
}

```

```
}  
}
```

- `allowed_maps` - string array. List of allowed maps.

#### errors

- [General](#) types only.

## logout

Destroys current user session. Works only with standard user session (not with API key).

#### parameters

Only session `hash`.

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/logout' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/user/logout?  
hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{ "success": true }
```

#### errors

- [General](#) types only.

## resend\_activation

Sends a new activation link to user.

It does not need authentication/hash and is available at `UNAUTHORIZED` access level.

## parameters

| name  | description         | type   | restrictions |
|-------|---------------------|--------|--------------|
| login | User login (email). | string | not null.    |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/resend_activation' \  
  -H 'Content-Type: application/json' \  
  -d '{"login": "user@login.com"}'
```

### HTTP GET

```
https://api.navixy.com/v2/user/resend_activation?  
login=user@login.com
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – user with a passed login not found.
- 209 - Failed sending email – can't send email.
- 264 - Timeout not reached – previous activation link generated less than 5 minutes ago (or other configured on server timeout).

```
{  
  "success": false,  
  "status": {  
    "code": 264,  
    "description": "Timeout not reached"  
  },  
  "timeout": "PT5M",  
  "remainder": "PT4M31.575S"  
}
```

- `timeout` - string. timeout between sending activation links in ISO-8601 duration format.
- `remainder` - string. remaining time to next try in ISO-8601 duration format
- 265 - Already done – user already activated and verified.





# User password


Contains API calls to change and set users' passwords.

## API actions

API path: `/user/password`.

### change

Changes password of user with the provided session hash (it is contained in a password restore link from email sent to user by `user/restore_password`).

 This call will receive only session hash from a password restore email. Any other hash will result in result error code 4 (User or API key not found or session ended).

### parameters

| name     | description  | type   |
|----------|--|--------|
| password | New password for the user. 6 to 20 printable characters. | string |

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/password/change' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "password":  
  "12@14Y$"}'
```

### response

```
{ "success": true }
```

### errors

- 101 – In demo mode this function disabled - if specified session hash belongs to demo user.

## set

Changes password for login user. Works only with standard user session (not with API key).

### parameters

| name         | description  | type   |
|--------------|--|--------|
| old_password | Current password of the user.                            | string |
| new_password | New password for the user. 6 to 20 printable characters. | string |

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/password/set' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",  
    "old_password": "qwert1", "new_password": "12@14Y$"}'
```

### response

```
{ "success": true }
```

### errors

- 101 – In demo mode this function disabled - if specified session hash belongs to demo user.
- 245 – New password must be different - if `old_password` = `new_password`.
- 248 – Wrong password - if `old_password` is wrong.

Last update: December 26, 2022







# User personal info

Contains user personal info update API call.

## API actions

API path: `/user/personal_info`.

### update

Updates user personal info.

Require a plugin with **id=45**.

#### parameters

- `legal_type` – string. Either "legal\_entity", "sole\_trader" or "individual".
- `first_name` – string. Contact person first name.
- `middle_name` – string. Contact person middle name.
- `last_name` – string. Contact person last name.
- `phone` – string. 0-15 digits. Optional. Contact phone. Not changes if not passed.
- `post_country` – string. Optional. Country part of user's post address.
- `post_index` – string. Optional. Index part of user's post address.
- `post_region` – string. Optional. Region part of user's post address.
- `post_city` – string. Optional. City from post address.
- `post_street_address` – string. Optional. User's post address,

and for `legal_entity` or `sole_trader`:

- `iec` – string. Industrial Enterprises Classifier aka "KPP". Used in Russia. For `legal_entity` only.
- `legal_name` – string. User legal (juridical) name. For `legal_entity` only.
- `okpo_code` – string, optional, 8 or 10 characters maximum. All-Russian Classifier of Enterprises and Organizations. Used in Russia.
- `registered_country` – string. Country part of user's registered address.
- `registered_index` – string. Index part of user's registered address.

- `registered_region` – string. Region part of user's registered address.
- `registered_city` – string. City from registered address.
- `registered_street_address` – string. User's registered address.
- `state_reg_num` - string, optional, 15 characters maximum. State registration number. E.g. EIN in the USA, OGRN in Russia.
- `tin` – string. Taxpayer identification number.

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/personal_info/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "first_name": "Charles", "middle_name": "Henry", "last_name": "Pearson", "legal_type": "legal_entity", "phone": "491761234567", "post_country": "Germany", "post_index": "61169", "post_region": "Hessen", "post_city": "Wiesbaden", "post_street_address": "Marienplatz 2", "registered_country": "Germany", "registered_index": "61169", "registered_region": "Hessen", "registered_city": "Wiesbaden", "registered_street_address": "Marienplatz 2", "state_reg_num": "12-3456789", "tin": "1131145180", "legal_name": "E. Biasi GmbH", "iec": "", "okpo_code": ""}'
```

## response

```
{ "success": true }
```

## errors

- 222 - Plugin not found – when plugin **45** not available for user.

Last update: December 26, 2022





# User audit

Contains user audit check-in method that calls when user opens UI or activates the UI tab in the browser after it hasn't been used for more than 2 hours.

## API actions

API path: `/user/audit`.

### checkin

This action occurs when a customer opens the UI or activates the UI tab in the browser after it hasn't been used for more than 2 hours. Works only with standard user session (not with API key). This action type may be in the [user audit log](#).

#### parameters

Only session `hash`.

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/audit/checkin' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/user/audit/checkin?  
hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{  
  "success": true  
}
```

#### errors

- [General](#) types only.

Last update: March 19, 2024







# User audit log

Using the audit log, account owner can track the activity of all users added through the "Access rights" section. Contains audit object and list method to get the audit log.

## Audit object

```
{
  "id": 44504790,
  "user_id": 3,
  "subuser_id": 184541,
  "entry_category": "custom_field",
  "entry_id": null,
  "action": "create",
  "payload": {
    "name": "Decimal number"
  },
  "host": "94.140.138.215",
  "user_agent": "Apache-HttpClient/4.1.1 (java 1.5)",
  "action_date": "2020-12-21 17:54:01"
}
```

- `id` - int. An ID of the audit record.
- `user_id` - int. Master user's ID.
- `subuser_id` - int. ID of the sub-user who made an action.
- `entry_category` - string. Category of the entry on which an action made.
- `entry_id` - int. ID of the entry on which an action made. Nullable.
- `action` - string. Action on entry.
- `payload` - Nullable JSON object. Additional information about action.
- `host` - string. Host from which an action made. IPv4 or IPv6.
- `user_agent` - string. User agent.
- `action_date` - [date/time](#). Date and time of the action.

## API actions

API path: `/user/audit/log`.

## list

Gets list of audit records available for current user.

**required sub-user rights:** `admin` (available only to master users).

### parameters

| name        | description  | type            |
|-------------|--|-----------------|
| from        | Include audit objects recorded after this date.  | date/<br>time   |
| to          | Include audits before this date.   | date/<br>time   |
| subuser_ids | Optional. Include audits for specific sub-users.   | int<br>array    |
| actions     | Optional. Include audits for specific actions only.  | string<br>array |
| limit       | Pagination. Maximum number of audit records to return.   | int             |
| offset      | Pagination. Get audits starting from.  | int             |
| sort        | Optional. Set of sort options. Each option is a pair of property name and sorting direction, e.g.<br><code>[ "action_date=asc", "user=desc" ]</code> . | string<br>array |
| grouping    | Optional. Group log by "user", "action_date", "action" or don't group "default".   | enum            |

Properties available for sorting by:

- `action`.
- `action_date` - sort only by date, not considering time part.
- `action_datetime` - sort by date including time.
- `user` - sort by user's (sub-user) last+first+middle name, not by ID.
- `host`. If no sort param is specified, then sorting equivalent to option `[ "action_date=asc" ]` will be applied.

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/audit/log/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "from": "2020-12-25 03:24:00", "to": "2020-12-28 06:24:00", "limit": 50, "offset": 0}'
```

## response

```
{
  "success": true,
  "list": [
    {
      "id": 44504790,
      "user_id": 3,
      "subuser_id": 184541,
      "entry_category": "custom_field",
      "entry_id": null,
      "action": "create",
      "payload": {
        "name": "Decimal number"
      },
      "host": "94.140.138.215",
      "user_agent": "Apache-HttpClient/4.1.1 (java 1.5)",
      "action_date": "2020-12-21 17:54:01"
    }
  ]
}
```

## errors

- [General](#) types only.

Last update: April 8, 2024



# User session

Contains a call to prolong user session.

## API actions

API path: `/user/session`.

### renew

Prolongs current user session. Works only with standard user session (not with API key).

#### parameters

Only session `hash`.

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/session/renew' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/user/session/renew?  
hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{ "success": true }
```

Last update: August 1, 2023



# Delivery

Calls to work with "delivery" type sessions. Those are special sessions to integrate order (task) tracking functionality into external systems.

## API actions

API path: `/user/session/delivery`.

### create

Creates new user delivery session. In demo session allowed to create a new session only if it not already exists.

**required sub-user rights:** `admin` (available only to master users).

### parameters

Only API key `hash`.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/session/delivery/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/user/session/delivery/create?
hash=a6aa75587e5c59c32d347da438505fc3
```

### response

```
{
  "success": true,
  "value": "42fc7d3068cb98d233c3af749dee4a8d"
}
```

- `value` - string. Created delivery session hash key.



## errors

- 101 - In demo mode this function disabled – current session is demo but weblocator session already exists.
- 236 – Feature unavailable due to tariff restrictions.

## read

Returns current user delivery session key.

## parameters

Only API key `hash`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/session/delivery/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/user/session/delivery/read?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "value": "42fc7d3068cb98d233c3af749dee4a8d"
}
```

- `value` - string. Delivery session hash.

## errors

- 201 – Not found in the database - if there is no delivery session.

## errors

- [General](#) types only.

Last update: July 19, 2022





# Push token

Contains API calls to interact with push token.

Find information about push token usage in our [instructions](#).

## API actions

API path: `/user/session/push_token`.

### bind

Binds Push token with a current session.

#### parameters

| name            | description   | type        |
|-----------------|---|-------------|
| application     | Application ID, "navixy_iphone_viewer" or "navixy_android_viewer" or "w3c_pushapi".   | enum        |
| token           | Push token or endpoint from pushSubscription, full URL like <a href="https://fcm.googleapis.com/fcm/send/f6kicrBn7S0:APA91b">https://fcm.googleapis.com/fcm/send/f6kicrBn7S0:APA91b</a> if your app ID is " " | string      |
| parameters      | Should be used only with object with "w3c_pushapi". Contain keys from pushSubscription {"p256dh": "...", "auth": "..."}.  | JSON object |
| category_filter | Optional. Push notifications category filter, default is <code>*</code> .   | string      |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/session/push_token/
bind' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"application": "navixy_android_viewer", "token":
"f4be7b9d04da2ce1af111b"}'
```

## response

```
{ "success": true }
```

Using `category_filter` you can filter out unwanted notifications categories.

If `category_filter` equals to `*` this means all categories allowed.

Delimited with comma list means that allowed only listed categories i.e.

`chat_message,history_rule`.

Prepended with minus and delimited with comma list means that all categories allowed except given i.e. `- history_task,history_rule`.

### POSSIBLE CATEGORIES:

- `chat_message` – notification about new chat message.
- `history_rule` – notifications related to rule actuation.
- `history_task` – notifications related to tasks.
- `history_info` – service information.
- `history_service_task` – service task notifications.
- `history_work_status` – work status notifications.

## errors

- [General](#) types only.

## delete

Deletes push token bound with the session.

## parameters

Only session `hash`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/session/push_token/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/user/session/push_token/delete?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{ "success": true }
```

## errors

[General](#) types only.

Last update: August 1, 2023







# User sessions weblocator

Calls to work with "weblocator" type sessions. Those are special sessions to integrate tracking device functionality into external systems.

## API actions

API path: `/user/sessions/weblocator`.

### create

Creates a new user weblocator session. In demo session allowed to create a new session only if it not already exists.

**required sub-user rights:** `admin` (available only to master users).

### parameters

Only API key `hash`.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/session/weblocator/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/user/session/weblocator/create?
hash=a6aa75587e5c59c32d347da438505fc3
```

### response

```
{
  "success": true,
  "value": "42fc7d3068cb98d233c3af749dee4a8d"
}
```

- `value` - string. Created session hash key.

## errors

- 101 - In demo mode this function disabled – current session is demo but weblocator session already exists.
- 236 – Feature unavailable due to tariff restrictions.

## read

Returns current user weblocator session key.

## parameters

Only API key `hash`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/session/weblocator/  
read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/user/session/weblocator/read?  
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{  
  "success": true,  
  "value": "42fc7d3068cb98d233c3af749dee4a8d"  
}
```

- `value` - string. Session hash key.

## errors

- 201 – Not found in the database - if there is no weblocator session.

Last update: July 19, 2022





# User settings

CRUD actions for user settings.

## settings object

```
{
  "time_zone": "Europe/Amsterdam",
  "locale": "nl_NL",
  "measurement_system": "metric",
  "date_format": "ddMMyyyy_dots",
  "hour_mode": "TWENTY_FOUR_HOURS",
  "geocoder": "osm",
  "route_provider": "google",
  "translit": false
}
```

- `time_zone` - [enum](#). ISO timezone ID.
- `locale` - [enum](#). Locale code.
- `measurement_system` - [enum](#). Measurement system. Can be "metric", "imperial", "us", "metric\_gal\_us" or "nautical".
- `date_format` - Optional [enum](#). Date representation. Can be "ddMMyyyy\_dots"("dd.MM.yyyy", "01.12.2021"), "ddMMyyyy\_slashes"("dd/MM/yyyy", "01/12/2021"), "MMddyyyy\_hyphens"("MM-dd-yyyy", "12-01-2021"), "yyyyMMdd\_hyphens"("yyyy-MM-dd", "2021-12-01"), "dMMMMy"("d MMM y", "1 Dec 2021") or "dMMMMMy"("d MMMM y", "1 December 2021")
- `hour_mode` - Optional [enum](#). Time representation. Can be "TWENTY\_FOUR\_HOURS" (24-hour clock, "HH:mm" or "HH:mm:ss", "17:45"/"17:45:46") or "TWELVE\_HOURS" (12-hour clock, "h:mm a" or "h:mm:ss a", "5:45 PM"/"5:45:46 PM")
- `geocoder` - [enum](#). Preferred geocoder type. Can be "google", "yandex", "progorod", "osm" or "locationiq".
- `route_provider` - [enum](#). Preferred route finding provider. Can be "google", "progorod" or "osrm".
- `translit` - boolean. `true` if sms notification should be transliterated, `false` otherwise.

`balance_alert_settings` type is JSON object:

```
{
  "emails": ["email1@example.com", "email2@example.com"]
}
```

- `emails` - string array. List of emails to send alert message about balance. Empty array means disclaimer of notifications.

`file_storage_settings` type is JSON object:

```
{
  "auto_overwrite": true
}
```

- `auto_overwrite` - boolean. If `true` new files will replace old ones when file storage is full. Default is `false`.

## API actions

API path: `/user/settings`.

### read

Reads current user's settings.

### parameters

Only API key `hash`.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/settings/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/user/settings/read?
hash=a6aa75587e5c59c32d347da438505fc3
```

### response

```
{
  "success": true,
  "settings": {
    "time_zone": "Europe/Amsterdam",
    "locale": "nl_NL",
  }
}
```

```

        "measurement_system": "metric",
        "geocoder": "osm",
        "route_provider": "google",
        "translit": false
    },
    "file_storage_settings": {
        "auto_overwrite": true
    },
    "balance_alert_settings": {
        "emails": ["email1@example.com", "email2@example.com"]
    },
    "first_user_balance_warning_period": "7d",
    "second_user_balance_warning_period": "2d"
}

```

- `first_user_balance_warning_period` - string. The first interval to send alert. "7d" means send the first alert warning 7 days before.
- `second_user_balance_warning_period` - string. The second interval to send alert. Send the second alert warning n days before.
- Where `settings`, `balance_alert_settings` and `file_storage_settings` described above.

**required sub-user rights** for `balance_alert_settings` and `file_storage_settings` fields: `admin` (available only to master users).

## update

Update current user's settings.

**required sub-user rights** for `balance_alert_settings` and `file_storage_settings`: `admin` (available only to master users).

## parameters

| name               | description  | type |
|--------------------|--|------|
| time_zone          | ISO timezone ID.   | enum |
| locale             | Locale code.   | enum |
| measurement_system | Measurement system. Can be "metric", "imperial", "us", "metric_gal_us" or "nautical".  | enum |
| geocoder           | Preferred geocoder type. Can be "google", "yandex", "progorod", "osm" or "locationiq". | enum |

| name                   | description   | type        |
|------------------------|---|-------------|
| route_provider         | Preferred route finding provider. Can be "google", "progorod" or "osrm".                      | enum        |
| translit               | <code>true</code> if sms notification should be transliterated, <code>false</code> otherwise. | boolean     |
| balance_alert_settings | Object containing array of emails.  | JSON object |
| file_storage_settings  | Object containing file storage settings.  | JSON object |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/settings/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "time_zone": "Europe/Amsterdam", "locale": "nl_NL", "measurement_system": "metric", "geocoder": "osm", "route_provider": "google", "translit": false, "balance_alert_settings": {"emails": ["email1@example.com", "email2@example.com"]}, "file_storage_settings": {"auto_overwrite": true}}'
```

## response

```
{ "success": true }
```

## errors

- [General](#) types only.

## file\_storage/update

Updates current user's file storage settings.

**required sub-user rights:** `admin` (available only to master users).



## parameters

| name                  | description                              | type        |
|-----------------------|--|-------------|
| file_storage_settings | Object containing file storage settings. | JSON object |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/settings/file_storage/
update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"file_storage_settings": {"auto_overwrite": true}}'
```

## errors

- 13 – Operation not permitted – if user has insufficient rights.

Last update: April 11, 2024





# User UI settings

The user interface settings intended for storing settings of client applications that use the API. One can imagine that this works similarly to the browser cache/local storage mechanism. The feature is that long-term storage of these settings provided but not guaranteed - when the quota exceeded, data could be deleted.

## API actions

API path: `/user/settings/ui.`

### read

Reads setting value by key.

#### parameters

| name | description   | type   |
|------|---|--------|
| key  | Length should be between 1 and 50 is 50 symbols, should only contain English letters, digits, <code>_</code> and <code>-</code> . | string |

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/user/settings/ui/read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "key":  
  "tracker-icons"}'
```

#### responses:

```
{  
  "success": true,  
  "value": "previously saved value"  
}
```

When nonexistent key provided:

```
{  
  "success": false,
```

```
"status": {
  "code": 201,
  "description": "Not found in database"
}
```

#### errors

- [General](#) types only.

## update

Sets setting value.

#### parameters

| name  | description   | type   |
|-------|---|--------|
| key   | Length should be between 1 and 50 is 50 symbols, should only contain English letters, digits, <code>_</code> and <code>-</code> . | string |
| value | A new UI config value. Length should be between 0 and 8192 symbols.   | string |

#### responses:

```
{ "success": true }
```

#### errors

- [General](#) types.
- 268 - over quota. The amount of storage available for the user for these settings has been exhausted. New settings cannot be added until the amount of stored data has been reduced.

Last update: December 26, 2022





# Check-ins

Check-ins are created using Mobile Tracker App ([Android](#) / [iOS](#)). They contain date/time, address, coordinates and additional information (comment, photo, filled form) which is provided by app user after pressing the "Check-in" in the tracker app. Using check-ins field personnel can provide information to their HQ while on site. For example, provide photo proof of the work done, or notify about a malfunction along with filled form describing the problem.

Check-ins cannot be created using web API ([create](#) is needed for exceptional cases and described in [how-tos](#)), so all actions are read-only.

## Check-in object

```
{
  "id": 1,
  "marker_time": "2017-03-15 12:36:27",
  "user_id": 111,
  "tracker_id": 222,
  "employee_id": 333,
  "location": {
    "lat": 53.787154,
    "lng": 9.757980,
    "address": "Moltkestrasse 32",
    "precision": 150
  },
  "comment": "houston, we have a problem",
  "files": [{
    "id": 16,
    "storage_id": 1,
    "user_id": 12203,
    "type": "image",
    "created": "2017-09-06 11:54:28",
    "uploaded": "2017-09-06 11:55:14",
    "name": "lala.jpg",
    "size": 72594,
    "mime_type": "image/png",
    "metadata": {
      "orientation": 1
    },
    "state": "uploaded",
    "download_url": "https://static.navixy.com/file/dl/1/0/1g/01gw2j5q7nm4r92dytolzd6kox9e38v.png/lala.jpg"
  }],
  "form_id": 23423,
```



```
"form_label": "Service request form"
}
```

- `id` - int. An ID of a check-in.
- `marker_time` - [date/time](#). Non-null. The time of check-in creation.
- `user_id` - int. Non-null. An ID of the master user.
- `tracker_id` - int. Non-null. An ID of the tracker which created this check-in.
- `employee_id` - optional int. An ID of the employee assigned to the tracker.
- `location` - non-null object. Location associated with this check-in marker.
  - `address` - string. Address of the location.
- `comment` - optional string. A comment provided by app user.
- `files` - list of objects. Non-null. May be empty.
  - `id` - int. File ID.
  - `storage_id` - int. Storage ID.
  - `user_id` - int. An ID of the user.
  - `type` - [enum](#). Can be "image" | "file".
  - `created` - [date/time](#). Date when file created.
  - `uploaded` - [date/time](#). Date when file uploaded, can be null if file not yet uploaded.
  - `name` - string. A name of the file.
  - `size` int. File size in bytes. If file not uploaded, show maximum allowed size for an upload.
  - `metadata` - metadata object.
    - `orientation` - int. Image exif orientation.
  - `state` - [enum](#). Can be "created" | "in\_progress" | "uploaded" | "deleted".
  - `download_url` - string. Actual URL at which file is available. Can be null if file not yet uploaded.
- `form_id` - int. An ID of the form which was sent along with a check-in, can be null.
- `form_label` - string. Label of the form which was sent along with a check-in, can be null.

## API actions

API path: `/checkin`.

## read

Get check-in which ID is equal to `checkin_id`.

**required sub-user rights:** `employee_update`.

### parameters

| name       | description               | type |
|------------|---------------------------|------|
| checkin_id | ID of the check-in entry. | int  |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/checkin/read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "checkin_id":  
1}'
```

#### HTTP GET

```
https://api.navixy.com/v2/checkin/read?  
hash=a6aa75587e5c59c32d347da438505fc3&checkin_id=1
```

### response

```
{  
  "success": true,  
  "value": {  
    "id": 1,  
    "marker_time": "2017-03-15 12:36:27",  
    "user_id": 111,  
    "tracker_id": 222,  
    "employee_id": 333,  
    "location": {  
      "lat": 53.787154,  
      "lng": 9.757980,  
      "address": "Moltkestrasse 32",  
      "precision": 150  
    },  
    "comment": "houston, we have a problem",  
    "files": [{  
      "id": 16,  
      "storage_id": 1,  
      "user_id": 12203,  
      "type": "image",  
      "created": "2017-09-06 11:54:28",  
      "uploaded": "2017-09-06 11:55:14",  
      "name": "lala.jpg",  
      "size": 72594,  
      "mime_type": "image/png",  
    }  
  ]  
}
```

```

        "metadata": {
            "orientation": 1
        },
        "state": "uploaded",
        "download_url": "https://static.navixy.com/file/dl/1/0/1g/01gw2j5q7nm4r92dytolzd6kox9e38v.png/lala.jpg"
    }],
    "form_id": 23423,
    "form_label": "Service request form"
}

```

## errors

- 7 – Invalid parameters.
- 204 – Entity not found – when the marker entry is not exists.

## list

Gets marker entries on a map for trackers and for the specified time interval.

**required sub-user rights:** `employee_update`.

## parameters

| name       | description  | type                        |
|------------|--|-----------------------------|
| trackers   | Optional. Array of tracker IDs. All trackers must not be deleted or blocked (if list_blocked=false). If not specified, all available trackers will be used as value.                                       | int array                   |
| from       | Optional. Start date/time for searching.   | <a href="#">date/time</a>   |
| to         | Optional. End date/time for searching. Must be after "from" date.  | <a href="#">date/time</a>   |
| conditions | Optional. Search conditions to apply to list. See <a href="#">Search conditions</a> . Allowed fields are <code>employee</code> , <code>location</code> , <code>marker_time</code> , <code>comment</code> . | string array                |
| sort       | Optional. List of sort expressions. See below.   | string array                |
| location   |  | Location JSON. For example, |

| name   | description  | type  |
|--------|--|---|
|        | Optional, location with radius, inside which check-ins must reside.  | <code>{ "lat": 53.787154, "lng": 9.757980, "radius": 350 }</code> |
| limit  | Optional. Max number of records to return.   | int   |
| offset | Optional, offset (starting index of first returned record), default is 0.  | int   |
| format | Optional. If empty, JSON will be returned. Otherwise server will return file download in specified format. Can be "pdf" or "xlsx". | string  |

#### CONDITION FIELDS

| Name        | Type     | Comment       |
|-------------|----------|---------------|
| employee    | number   | ID            |
| tracker_id  | number   |               |
| marker_time | DateTime |               |
| location    | string   | address       |
| comment     | string   |               |
| form        | number   | template's ID |

#### SORT

It's a set of sort options. Each option is a pair of field name and sorting direction, e.g. `["location=asc", "employee=desc", "marker_time=desc"]`.

#### SORT FIELDS

| Name     | Type   | Comment   |
|----------|--------|-----------|
| employee | string | full name |

| Name        | Type     | Comment |
|-------------|----------|---------|
| tracker_id  | number   |         |
| marker_time | DateTime |         |
| location    | string   | address |
| comment     | string   |         |
| form        | string   | label   |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/checkin/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "trackers": [616384,345623], "from": "2020-08-05 03:06:00", "to": "2020-09-05 03:00:00", "offset": 20, "limit": 100, "format": "xlsx"}'
```

## response

```
{
  "success": true,
  "list": [<checkin>],
  "count": 22
}
```

- `list` - list of check-in objects.
- `count` - int. Total number of check-ins (ignoring offset and limit).

## errors

- 7 – Invalid parameters.
- 211 – Requested time span is too big.
- 217 – The list contains non-existent entities – if one of the specified trackers does not exist, is blocked or doesn't have required tariff features.
- 221 – Device limit exceeded - if device limit set for the user's dealer has been exceeded.

## delete

Deletes check-ins with the specified IDs.

**required sub-user rights:** `checkin_update`.

### parameters

| name        | description           | type      |
|-------------|-----------------------|-----------|
| checkin_ids | List of check-in IDs. | int array |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/checkin/delete' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",  
  "checkin_ids": [2132,4533]}'
```

#### HTTP GET

```
https://api.navixy.com/v2/checkin/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&checkin_ids=[2132,4533]
```

### response

```
{  
  "success": true  
}
```

### errors

- 7 – Invalid parameters.
- 201 – Not found in the database - check-ins with the specified IDs don't exist, or their corresponding trackers are not available to current sub-user.

## create

Creates a new check-in. Needed for exceptional cases.

**required sub-user rights:** `checkin_update`.

## parameters

| name            | description   | type        |
|-----------------|---|-------------|
| tracker_id      | ID of the tracker. Tracker must belong to authorized user and not be blocked.   | int         |
| location        | Location coordinates (see: <a href="#">data types description section</a> section).   | JSON object |
| comment         | Optional.   | string      |
| file_ids        | Optional. IDs of files created by checkin/image/create).  | int array   |
| form_submission | Optional, only present when sending form along with check-in. If the form includes optional fields that should be left empty for your check-in, refrain from adding these fields to the form submission object. | JSON object |

where `form_submission` type is JSON object:

```
{
  "form_id": <int>, // id of the form previously created with
  checkin/form/create
  "values": {
    // map which contains values for form fields
  }
}
```

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/checkin/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
22, "location": { "lat": 9.861999, "lng": -83.948999 }, "comment":
"houston, we have a problem", "file_ids": [11, 22],
"form_submission": { "form_id": 23423, "values": {"111-aaa-
whatever": { "type": "text", "value": "John Doe" }} } }'
```

## response

```
{
  "success": true,
```

```
"id": 111
}
```

#### errors

- 7 – Invalid parameters.
- 201 – Not found in the database - form with the specified IDs don't exist, or their corresponding trackers are not available to current sub-user.
- 242 – There were errors during content validation, if given values are invalid for the form.

#### image/create

Creates an image for check-in. If you have multiple files to upload, be sure to add a brief delay between uploading each one to ensure a smooth process.

#### parameters

| name     | description  | type        |
|----------|--|-------------|
| size     | Maximum size in bytes for the file which will be uploaded.<br>This is needed to "reserve" the space for a file in user's disk space quota. | int         |
| filename | Optional. If specified, uploaded file will have the specified name. If not, name will be taken from actual file upload form.               | string      |
| metadata | Optional. Metadata object (for images only).   | JSON object |

#### response

when using internal storage:

```
{
  "success": true,
  "value": {
    "file_id": 111,
    "url": "http://bla.org/bla",
    "expires": "2020-02-03 03:04:00",
    "file_field_name": "file",
    "fields": {
      "token": "a43f43ed4340b86c808ac"
    }
  }
}
```



```
}
}
}
```

when using the Amazon S3:

```
{
  "success": true,
  "value": {
    "file_id": 111,
    "url": "https://bla.s3.amazonaws.com/",
    "expires": "2020-02-03 03:04:00",
    "file_field_name": "file",
    "fields": {
      "policy": "<Base64-encoded policy string>",
      "key": "user/user1/${filename}",
      "success_action_status": "200",
      "x-amz-algorithm": "AWS4-HMAC-SHA256",
      "x-amz-credential": "AKIAIOSFODNN7EXAMPLE/20151229/us-east-1/
s3/aws4_request",
      "x-amz-date": "20151229T000000Z",
      "x-amz-signature": "<signature-value>",
      "x-amz-server-side-encryption": "AES256",
      "content-type": "image/png"
    }
  }
}
```

- `file_id` - int. This value will be submitted as form's field value.
- `url` - string. A URL to which POST form-data with file contents should be executed.
- `expires` - date/time. After this date file record wil expire and upload requests will be rejected.
- `file_field_name` - string. Name for file field in POST upload request.
- `fields` - these fields should be passed as additional fields in POST multipart upload request, field with a file must be the last one.

### How to upload file data

Here's an example of upload you must make after receiving such response (assuming you uploading image named `actual_file_name.png`):

Internal storage example:

```
POST /bla HTTP/1.1
Host: bla.org
Content-Length: 1325
Origin: http://bla.org
... other headers ...
Content-Type: multipart/form-data; boundary=WebAppBoundary
```

```
--WebAppBoundary
Content-Disposition: form-data; name="token"

a43f43ed4340b86c808ac
--WebAppBoundary
Content-Disposition: form-data; name="file";
filename="actual_file_name.png"
Content-Type: image/png

... contents of file goes here ...
--WebAppBoundary--
```

### Amazon S3 example:

```
POST / HTTP/1.1
Host: https://bla.s3.amazonaws.com
Content-Length: 1972
Origin: https://bla.s3.amazonaws.com/
... other headers ...
Content-Type: multipart/form-data; boundary=WebAppBoundary

--WebAppBoundary
Content-Disposition: form-data; name="policy"
Content-Type: text/plain

eyJleHBpcmF0aW9uIjogIjIwMjMtMDMtMjdUMjE6MTU6MzYuMDczWiIsImNvbRpdGlvbni
--WebAppBoundary
Content-Disposition: form-data; name="key"
Content-Type: text/plain

nj9relv6m52qp01t0wv47wyk1ozd309g/${filename}
--WebAppBoundary
Content-Disposition: form-data; name="success_action_status"
Content-Type: text/plain

200
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-algorithm"
Content-Type: text/plain

AWS4-HMAC-SHA256
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-credential"
Content-Type: text/plain

AKIAIBQ6SRB65EVSSRMA/20230327/eu-central-1/s3/aws4_request
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-date"
Content-Type: text/plain

20230327T210036Z
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-signature"
Content-Type: text/plain

2df7efa0c0e0c5b97d0d9483acd77c9ec37360df921b019a4c4a93180a6136ad
```

```
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-server-side-encryption"
Content-Type: text/plain

AES256
--WebAppBoundary
Content-Disposition: form-data; name="file";
filename="actual_file_name.png"
Content-Type: image/png

... contents of file goes here ...
--WebAppBoundary--
```

## errors

- 268 – File cannot be created due to quota violation.
- 271 – File size is larger than the maximum allowed (by default 16 MB).

## form/create

Creates a new form that can be attached to a check-in. Form always created on the basis of form template.

## parameters

| name        | description   | type |
|-------------|---|------|
| tracker_id  | ID of the tracker. Tracker must belong to authorized user and not be blocked. | int  |
| template_id | ID of the form template.  | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/checkin/form/create' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":  
22, "template_id": 12548}'
```

### HTTP GET

```
https://api.navixy.com/v2/checkin/form/create?  
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=22&template_id=12548
```

## response

```
{
  "success": true,
  "id": 23423
}
```

## errors

- 201 – Not found in the database - if there is no template with such an ID.

## form/file/create

Creates a new file entry associated with form's field. If you have multiple files to upload, be sure to add a brief delay between uploading each one to ensure a smooth process.

## parameters

| name       | description   | type        |
|------------|---|-------------|
| checkin_id | ID of the check-in to which form attached.  | int         |
| form_id    | ID of the form.   | int         |
| field_id   | ID of the form's field to which a new file should be attached.  | string      |
| size       | Maximum size in bytes for the file which will be uploaded. This is needed to "reserve" the space for a file in user's disk space quota. | int         |
| filename   | Optional. If specified, uploaded file will have the specified name. If not, name will be taken from actual file upload form.            | string      |
| metadata   | Optional. Metadata object (for images only).  | JSON object |

- Use only one parameter `checkin_id` or `form_id`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/checkin/form/file/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "checkin_id": 1, "field_id": "111-aaa-whatever", "size": 101}'
```

## response

The response and update process are same to [image/create](#).

- `file_id` - int. This value will be submitted as form's field value.
- `url` - string. A URL to which POST form-data with file contents should be executed.
- `expires` - date/time. After this date file record will expire and upload requests will be rejected.
- `file_field_name` - string. Name for file field in POST upload request.
- `fields` - these fields should be passed as additional fields in POST multipart upload request, field with a file must be the last one.

## errors

- 201 – Not found in the database - if there is no check-in with such an ID, or check-in doesn't have form, or form has no field with such a `field_id`.
- 231 – Entity type mismatch - if form field is not file-based, i.e. doesn't use file ID as its value.
- 267 – Too many entities - if there are 6 or more unsubmitted files already associated with this form's field.
- 268 – File cannot be created due to quota violation.
- 271 – File size is larger than the maximum allowed (by default 16 MB).

Last update: April 8, 2024





# Departments

Department is essentially just a group of [employees](#). They can be assigned to departments by specifying non-null `department_id`.

## Department object

```
{
  "id": 222,
  "label": "Drivers",
  "location": {
    "lat": 46.9,
    "lng": 7.4,
    "address": "Rosenweg 3",
    "radius": 150
  }
}
```

- `id` - int. An ID of department.
- `label` - string. Name of department.
- `location` - optional object. Location associated with these departments. Should be valid or null.
  - `address` - string. Address of the location.
  - `radius` - int. Radius of location zone in meters.

## API actions

API base path: `/department`.

### list

Gets all departments belonging to user.



## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/department/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/department/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [{
    "id": 222,
    "label": "Drivers",
    "location": {
      "lat": 46.9,
      "lng": 7.4,
      "address": "Rosenweg 3",
      "radius": 150
    }
  }]
}
```

## errors

- 7 – Invalid parameters.
- 217 – The list contains non-existent entities – if one of the specified trackers does not exist, is blocked or doesn't have required tariff features.
- 221 – Device limit exceeded - if device limit set for the user's dealer has been exceeded.

## create

Creates a new department with specified parameters.

**required sub-user rights:** `employee_update`.

## parameters

| name       | description   | type        |
|------------|---|-------------|
| department | An <a href="#">department object</a> without <code>id</code> field. | JSON object |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/department/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "department":
{"label": "My Department", "location": {"lat": 46.9, "lng": 7.4,
"address": "Rosenweg 3", "radius": 50}}}'
```

## response

```
{
  "success": true,
  "id": 111
}
```

- `id` - int. An ID of the created department.

## errors

- 7 – Invalid parameters.
- 217 – The list contains non-existent entities – if one of the specified trackers does not exist, is blocked or doesn't have required tariff features.
- 221 – Device limit exceeded - if device limit set for the user's dealer has been exceeded.

## update

Updates existing department with a new specified parameters.

**required sub-user rights:** `employee_update`.

## parameters

| name       | description                            | type        |
|------------|--|-------------|
| department | An <a href="#">department object</a> . | JSON object |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/department/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "department": {"id": 111, "label": "My Department", "location": {"lat": 46.9, "lng": 7.4, "address": "Rosenweg 3", "radius": 50}}}'
```

## response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - if there is no department with specified ID.

## delete

Deletes department with the specified ID.

**required sub-user rights:** employee\_update .

## parameters

| name          | description              | type |
|---------------|--------------------------|------|
| department_id | An ID of the department. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/department/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "department_id": 111}'
```

### HTTP GET

```
https://api.navixy.com/v2/department/delete?
hash=a6aa75587e5c59c32d347da438505fc3&department_id=111
```

## response

```
{ "success": true }
```

**errors**

- 201 – Not found in the database - if there is no department with specified ID.

Last update: December 26, 2022





# Working with employees and drivers

Employees and drivers used to represent people working at one's organization. They can be linked with other entities such as trackers, vehicles, places, etc.

## Employee object

```
{
  "id": 222,
  "tracker_id": null,
  "first_name": "John",
  "middle_name": "Jane",
  "last_name": "Smith",
  "email": "smith@example.com",
  "phone": "442071111111",
  "driver_license_number": "SKIMP407952HJ9GK 06",
  "driver_license_cats": "C",
  "driver_license_issue_date": "2008-01-01",
  "driver_license_valid_till": "2018-01-01",
  "hardware_key": null,
  "icon_id": 55,
  "avatar_file_name": null,
  "department_id": null,
  "location": {
    "lat": 52.5,
    "lng": 13.4,
    "address": "Engeldamm 18"
  },
  "personnel_number": "1059236",
  "ssn": "123-45-6789",
  "tags": [1,2]
}
```

- `id` - int. Internal ID. Can be passed as null only for "create" action.
- `tracker_id` - int. An ID of the tracker currently assigned to this employee or driver. `null` means no tracker assigned.
- `first_name` - string. First name. Cannot be empty. Max 100 characters.
- `middle_name` - string. Middle name. Can be empty, cannot be null. Max 100 characters.
- `last_name` - string. Last name. Can be empty, cannot be null. Max 100 characters.
- `email` - string. Employee's email. Must be valid email address. Can be empty, cannot be null. Max 100 characters.

- `phone` - string. Employee's phone without "+" sign. Can be empty, cannot be null. Max 32 characters.
- `driver_license_number` - string. Driver license number. Can be empty, cannot be null. Max 32 characters.
- `driver_license_cats` - string. Driver license categories. Max 32 characters.
- `driver_license_issue_date` - string date (yyyy-MM-dd). Issue date of a driver license. Can be null.
- `driver_license_valid_till` - string date (yyyy-MM-dd). Date till a driver license valid. Can be null.
- `hardware_key` - string. A hardware key. Can be null. Max 64 characters.
- `icon_id` - int. An icon ID. Can be null, can only be updated via [avatar/assign](#).
- `avatar_file_name` - string. A name of the updated avatar file. Nullable, can only be updated via [avatar/upload](#).
- `department_id` - int. An ID of the department to which employee assigned. Can be null.
- `location` - optional object. Location associated with this employee, should be valid or null.
  - `address` - string. Address of the location.
- `personnel_number` - optional string. Max length is 15.
- `ssn` - optional string. Social Security number. Max length is 32.
- `tags` - int array. List of tag IDs.

## API actions

API base path: `/employee`.

### list

Gets all employees and drivers belonging to user.

#### parameters

| name  | description   | type |
|-------|---|------|
| limit | Pagination. Maximum number of employee records to return. | int  |



| name   | description   | type         |
|--------|---|--------------|
| offset | Pagination. Get employee records starting from.   | int          |
| sort   | Optional. Set of sort options. Each option is a pair of property name and sorting direction, e.g.<br><code>[{"first_name=desc", "object_label=asc"}]</code> . Maximum 2 options in request. Available properties:<br>- ID<br>- first_name<br>- object_label<br>- department_label<br>- personnel_number<br>- hardware_key<br>- phone<br>- email<br>- address<br>- driver_license_number<br>- driver_license_cats<br>- driver_license_valid_till<br>- driver_license_valid_till<br>- ssn | string array |
| filter | Get a list of employees filtered by properties, at least one property must contain the desired string. All properties from the sorting list are used in filtering. Maximum 100 characters or null.  | string       |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/employee/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/employee/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [<employee>],
```

```
    "count": 12
}
```

- `list` - a list of employees.
- `count` - int. Total number of employees (ignoring offset and limit).

## errors

General types only.

## create

Creates a new employee/driver.

**required sub-user rights:** `employee_update`.

## parameters

| name     | description   | type        |
|----------|---|-------------|
| employee | An <a href="#">employee object</a> without <code>id</code> field. Non-null. | JSON object |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/employee/create' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "employee": \  
{"tracker_id": 625987, "first_name": "John", "middle_name": \  
"Jane", "last_name": "Smith", "email": "smith@example.com", \  
"phone": "442071111111", "driver_license_number": \  
"SKIMP407952HJ9GK 06", "driver_license_cats": "C", \  
"driver_license_valid_till": "2018-01-01", "hardware_key": null, \  
"icon_id" : 55, "avatar_file_name": null, "department_id": null, \  
"location": {"lat": 52.5, "lng": 13.4, "address": "Engeldamm 18"}, \  
"personnel_number": "1059236", "tags": [1,2]}'
```

## response

```
{  
  "success": true,  
  "id": 111  
}
```

- `id` - int. An ID of the created employee (driver).

## errors

- 247 – Entity already exists, if `tracker_id != null` and exists an employee that already bound to this `tracker_id`.

## read

Gets employee/driver by his ID.

## parameters

| name        | description        | type |
|-------------|--------------------|------|
| employee_id | ID of an employee. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/employee/read' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",  
"employee_id": 111}'
```

### HTTP GET

```
https://api.navixy.com/v2/employee/read?  
hash=a6aa75587e5c59c32d347da438505fc3&employee_id=111
```

## response

```
{  
  "success": true,  
  "value": {  
    "id": 222,  
    "tracker_id": null,  
    "first_name": "John",  
    "middle_name": "Jane",  
    "last_name": "Smith",  
    "email": "smith@example.com",  
    "phone": "442071111111",  
    "driver_license_number": "SKIMP407952HJ9GK 06",  
    "driver_license_cats": "C",  
    "driver_license_issue_date": "2008-01-01",  
    "driver_license_valid_till": "2018-01-01",  
    "hardware_key": null,  
    "icon_id": 55,  
    "avatar_file_name": null,  
    "department_id": null,  
    "location": {
```

```

        "lat": 52.5,
        "lng": 13.4,
        "address": "Engeldamm 18"
    },
    "personnel_number": "1059236",
    "ssn": "123-45-6789",
    "tags": [1,2]
}

```

- `value` - an employee object.

## errors

- 201 – Not found in the database - if there is no employee/driver with such an ID.

## update

Updates existing employee/driver.

**required sub-user rights:** `employee_update`.

## parameters

| name     | description  | type        |
|----------|--|-------------|
| employee | An <a href="#">employee object</a> with <code>id</code> field. Non-null. | JSON object |

## example

### cURL

```

curl -X POST 'https://api.navixy.com/v2/employee/update' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "employee":
{"employee_id": 111, "tracker_id": 625987, "first_name": "John",
"middle_name": "Jane", "last_name": "Smith", "email":
"smith@example.com", "phone": "442071111111",
"driver_license_number": "SKIMP407952HJ9GK 06",
"driver_license_cats": "C", "driver_license_valid_till":
"2018-01-01", "hardware_key": null, "icon_id" : 55,
"avatar_file_name": null, "department_id": null, "location":
{"lat": 52.5, "lng": 13.4, "address": "Engeldamm 18"},
"personnel_number": "1059236", "tags": [1,2]}}'

```

## response

```

{ "success": true }

```

## errors

- 201 – Not found in the database - if there is no employee/driver with such an ID.
- 247 – Entity already exists, if `tracker_id != null` and exists an employee that already bound to this `tracker_id`.

## delete

Deletes an employee/driver with the specified ID.

**required sub-user rights:** `employee_update`.

## parameters

| name                     | description                           | type |
|--------------------------|---------------------------------------|------|
| <code>employee_id</code> | ID of an employee (driver) to delete. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/employee/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
    "employee_id": 111}'
```

### HTTP GET

```
https://api.navixy.com/v2/employee/delete?
hash=a6aa75587e5c59c32d347da438505fc3&employee_id=111
```

## response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - if there is no employee/driver with such an ID.

## batch\_convert

Converts batch of tab-delimited employees/drivers and returns list of checked employees/drivers with errors.

**Required sub-user rights:** `employee_update`.

#### parameters

| name           | description  | type         |
|----------------|--|--------------|
| batch          | Batch of tab-delimited employees/drivers.  | string       |
| file_id        | Preloaded file ID.   | string       |
| fields         | Optional. Array of field names. Default is <code>["first_name", "middle_name", "last_name", "email", "phone"]</code> . | string array |
| geocoder       | Geocoder type.   | string       |
| default_radius | Optional. Radius for point in meters. Default is 100.  | int          |

- If `file_id` is set – `batch` parameter will be ignored.

#### response

```
{
  "success": true,
  "list": [{
    "success": true,
    "value": {
      "id": 222,
      "tracker_id": null,
      "first_name": "John",
      "middle_name": "Jane",
      "last_name": "Smith",
      "email": "smith@example.com",
      "phone": "442071111111",
      "driver_license_number": "SKIMP407952HJ9GK 06",
      "driver_license_cats": "C",
      "driver_license_issue_date": "2008-01-01",
      "driver_license_valid_till": "2018-01-01",
      "hardware_key": null,
      "icon_id": 55,
      "avatar_file_name": null,
      "department_id": null,
      "location": {
        "lat": 52.5,
        "lng": 13.4,
        "address": "Engeldamm 18"
      },
      "personnel_number": "1059236",
      "ssn": "123-45-6789",
    }
  ]
}
```

```
    "tags": [
      1,
      2
    ],
    "errors": <array_of_objects>
  }
}],
"limit_exceeded": false
}
```

- `list` - list of checked employees/drivers.
  - `errors` - optional array of errors.
- `limit_exceeded` - boolean. `true` if given batch constrained by a limit.

#### **errors**

- 234 - Invalid data format.

Last update: April 8, 2024







# Changing avatar

Avatars can't be changed through `/employee/update`, you must use either `assign` (to set avatar to one of preset icons), or `upload` (to upload your own image).

## API actions

API path: `/employee/avatar`.

### assign

Assign `icon_id` (from standard icon set) to this employee/driver. The `icon_id` can be `null` – this means that uploaded avatar should be used instead of icon.

**required sub-user rights:** `employee_update`.

### parameters

| name        | description   | type |
|-------------|---|------|
| employee_id | ID of the employee/driver to whom the icon will assign. | int  |
| icon_id     | ID of the icon.   | int  |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/employee/avatar/assign' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
    "employee_id": 2132, "icon_id": 3654}'
```

#### HTTP GET

```
https://api.navixy.com/v2/employee/avatar/assign?
hash=a6aa75587e5c59c32d347da438505fc3&employee_id=2132&icon_id=3654
```

### response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - when employee/driver with `employee_id` not found.

## upload

Uploads avatar image for specified employee/driver. Then it will be available from `/employee/avatars/` e.g. `https://api.navixy.com/v2/static/employee/avatars/abcdef123456789.png`.

**required sub-user rights:** `employee_update`.

**avatar\_file\_name** returned in response and will be returned from </employee/list>.

**MUST** be a POST multipart request (multipart/form-data), with one of the parts being an image file upload (with the name `file`).

File part **mime** type must be one of:

- `image/jpeg`
- `image/pjpeg`
- `image/png`
- `image/gif`
- `image/webp`

## parameters

| name            | description  | type   |
|-----------------|--|--------|
| employee_id     | ID of the employee/driver to whom the icon will assign.                            | int    |
| file            | Image file.  | string |
| redirect_target | Optional. URL to redirect. If passed returns redirect to <code>?response=</code> . | string |

## response

```
{  
  "success": true,
```

```
"value": "picture.png"  
}
```

- `value` - string. Uploaded file name.

#### errors

- 201 – Not found in the database - when employee/driver with `employee_id` not found.
- 233 – No data file - if `file` part not passed.
- 234 – Invalid data format - if passed `file` with unexpected `mime` type.
- 254 – Cannot save file - on some file system errors.

Last update: December 26, 2022





# Employee import

## API actions

API calls to import employees.

## API actions

API path: `/employee/import/`.

### start

Starting the background process of importing employees.

### parameters

| name         | description   | type         |
|--------------|---|--------------|
| filename     | Name of file preloaded with <a href="#">/data/spreadsheet/parse</a> | string       |
| headers      | List of files' headers, see available fields above                  | string array |
| user_headers | Optional. List of user labels for headers                           | string array |

Available fields:

- `first_name`
- `middle_name`
- `last_name`
- `email`
- `phone`
- `driver_license_number`
- `driver_license_cats`
- `driver_license_issue_date`

- `driver_license_valid_till`
- `hardware_key`
- `address`
- `lat`
- `lng`
- `radius`
- `personnel_number`
- `ssn`
- `tracker_label`
- `tags`
- `undefined` (if a meaning of a field is not known)

## response

```
{
  "success": true,
  "id": <int>
}
```

## example

### cURL

```
curl -X POST "https://api.navixy.com/v2/employee/import/start" \
-H "Content-Type: application/json" \
--data-binary @- << EOF
{
  "hash": "a6aa75587e5c59c32d347da438505fc3",
  "filename": "tmp-sheet640571613016981796.tsv",
  "headers": ["label", "model", "max_speed", "type", "subtype",
"reg_number", "fuel_grade", "fuel_tank_volume",
"free_insurance_policy_number", "free_insurance_valid_till",
"tracker_label", "tags"],
  "user_headers": [ "Model", "Max speed", "Type", "Subtype",
"Reg. number", "Fuel grade", "Fuel tank volume", "Free insurance
policy number", "Free insurance valid till", "Object", "Tags"]
}
EOF
```

## errors

- 15 - Too many requests (rate limit exceeded) - if too many imports in progress
- 233 - No data file
- 234 - Invalid data format



- 247 - Entity already exists - there is another identical import with the same file

## read

Returns an import process with specified ID.

### parameters

| name       | description | type |
|------------|-------------|------|
| process_id | Process ID  | int  |

### response

```
{
  "success": true,
  "value": {
    "id": <int>,
    "user_id": <int>,
    "created": <date>,
    "type": "Employee",
    "params": {
      "headers": [<string>, <string>,...] // List of files' headers
    },
    "filename": <string>, // Name of preloaded TSV.
    "status": <string>, // created | in_progress | done | failed |
finished
    "status_change_date": <date>,
    "progress": {
      "imported": <int>,
      "failed": <int>,
      "percent": <int>, // approximate percentage of processed
      "processed_lines": <int>,
      "warnings": [{line:<int>, error: <string>}], // first 25
      "errors": [{line:<int>, error: <string>}], // first 25
    }
  }
}
```

### example

#### cURL

```
curl -X POST "https://api.navixy.com/v2/employee/import/read" \
-H "Content-Type: application/json" \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "process_id":
1}'
```

## errors

- 201 – Not found in database (if import is not found)

## list

Returns the list of the user's employee import processes.

## response

```
{
  "success" : true,
  "list" : [ {
    "id": <int>,
    "user_id": <int>,
    "created": <date>,
    "type": "Employee",
    "params": {
      "headers": [<string>, <string>,...] // List of files' headers
    },
    "filename": <string>, // Name of preloaded TSV.
    "status": <string>, // created | in_progress | done | failed
    "status_change_date": <date>,
    "progress": {
      "imported": <int>,
      "failed": <int>,
      "percent": <int>, // approximate percentage of processed
      "processed_lines": <int>,
      "warnings": [{line:<int>, error: <string>}], // first 25
      "errors": [{line:<int>, error: <string>}], // first 25
    }
  }, ...]
}
```

## example

### cURL

```
curl -X POST "https://api.navixy.com/v2/employee/import/list" \
-H "Content-Type: application/json" \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

## download\_failed

Retrieve a file with lines that contained errors and did not pass validation.

## parameters

| name       | description | type |
|------------|-------------|------|
| process_id | Process ID  | int  |

## response

File (standard file download).

## example

### cURL

```
curl -X POST "https://api.navixy.com/v2/employee/import/download_failed" \
  -H "Content-Type: application/json" \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "process_id": 7}'
```

## errors

- 201 – Not found in database (if import is not found)
- 204 – Entity not found (if file is not found)

Last update: October 6, 2023





# About forms

Forms used to provide additional information, such as user's name, phone, delivery date, etc. upon task completion or check-in from iOS/Android mobile tracker app. Forms can be attached to tasks. If form attached to task, this task cannot be completed without form submission.

- Each form must be created from template, read more at [Templates](#)
- For description of `<form_field>` and `<field_value>`, see [Form fields and values](#)
- Using web API, it's now possible to only attach/fill forms with tasks (checkin forms are created through Android/iOS tracker applications). See [Task form actions](#) to use forms with tasks.

Find comprehensive information on forms usage in our [instructions](#).

## Form object

```
{
  "id": 2,
  "label": "Order form",
  "fields": [
    {
      "id": "111-aaa-whatever",
      "label": "Name",
      "description": "Your full name",
      "required": true,
      "min_length": 5,
      "max_length": 255,
      "type": "text"
    }
  ],
  "created": "2017-03-15 12:36:27",
  "submit_in_zone": true,
  "task_id": 1,
  "template_id": 1,
  "values": {
    "111-aaa-whatever": {
      "type": "text",
      "value": "John Doe"
    }
  },
  "submitted": "2017-03-21 18:40:54",
  "submit_location": {
    "lat": 11.0,
    "lng": 22.0,
    "address": "Wall Street, NY"
  }
}
```

```
}
}
```

- `id` - int. Form unique ID.
- `label` - string. User-defined form label, from 1 to 100 characters.
- `fields` - array of multiple `form_field` objects.
- `created` - `date/time`. Date when this form created (or attached to the task). The read-only field.
- `submit_in_zone` - boolean. If `true`, form can be submitted only in task zone.
- `task_id` - int. An ID of the task to which this form attached.
- `template_id` - int. An ID of the form template on which this form based. Can be null if template deleted.
- `values` - a map with field IDs as keys and `field_value` objects as values. Can be null if form not filled.
  - `key` - string. Key used to link field and its corresponding value.
- `submitted` - `date/time`. Date when form values last submitted.
- `submit_location` - location at which form values last submitted.

## Form file object

```
{
  "id": 16,
  "storage_id": 1,
  "user_id": 12203,
  "type": "image",
  "created": "2017-09-06 11:54:28",
  "uploaded": "2017-09-06 11:55:14",
  "name": "lala.jpg",
  "size": 72594,
  "mime_type": "image/png",
  "metadata": <metadata_object>,
  "state": "uploaded",
  "download_url": "https://static.navixy.com/file/dl/1/0/1g/01gw2j5q7nm4r92dytolzd6koxy9e38v.png/lala.jpg"
}
```

- `id` - int. File ID.
- `type` - `enum`. Can be "image" or "file".
- `created` - `date/time`. Date when file created.
- `uploaded` - `date/time`. Date when file uploaded. Can be null if file not yet uploaded.

- `name` - string. A filename.
- `size` - int. Size in bytes. If file not uploaded, show maximum allowed size for the upload.
- `metadata` - nullable metadata object.
- `state` - [enum](#). Can be "created" | "in\_progress" | "uploaded" | "deleted".
- `download_url` - string. Actual URL at which file is available. Can be null if file not yet uploaded.

## API actions

API path: `/form`.

### read

Gets form by an ID.

#### parameters

| name            | description     | type |
|-----------------|-----------------|------|
| <code>id</code> | ID of the form. | int  |

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/form/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id": 2}'
```

##### HTTP GET

```
https://api.navixy.com/v2/form/read?
hash=a6aa75587e5c59c32d347da438505fc3&id=2
```

#### response

```
{
  "success": true,
  "value": {
    "id": 2,
    "label": "Order form",
    "fields": [
      {
        "id": "111-aaa-whatever",
```



```

        "label": "Name",
        "description": "Your full name",
        "required": true,
        "min_length": 5,
        "max_length": 255,
        "type": "text"
    }
],
"created": "2017-03-15 12:36:27",
"submit_in_zone": true,
"task_id": 1,
"template_id": 1,
"values": {
    "111-aaa-whatever": {
        "type": "text",
        "value": "John Doe"
    }
},
"submitted": "2017-03-21 18:40:54",
"submit_location": {
    "lat": 11.0,
    "lng": 22.0,
    "address": "Wall Street, NY"
}
},
"files": [{
    "id": 16,
    "storage_id": 1,
    "user_id": 12203,
    "type": "image",
    "created": "2017-09-06 11:54:28",
    "uploaded": "2017-09-06 11:55:14",
    "name": "lala.jpg",
    "size": 72594,
    "mime_type": "image/png",
    "metadata": {
        "orientation": 1
    },
    "state": "uploaded",
    "download_url": "https://static.navixy.com/file/dl/1/0/1g/01gw2j5q7nm4r92dytolzd6koxy9e38v.png/lala.jpg"
}]
}

```

- `value` - A [form object](#).
- `files` - list of [form\\_file objects](#). Files used in values of this form. Can be null or empty.

## errors

- 201 – Not found in the database - if there is no form with such an ID.

## download

Downloads form as a file by an ID.

### parameters

| name   | description                          | type |
|--------|--------------------------------------|------|
| id     | ID of the form.                      | int  |
| format | File format. Can be "pdf" or "xlsx". | enum |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/form/download' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id": 2,
    "format": "pdf"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/form/download?
hash=a6aa75587e5c59c32d347da438505fc3&id=2&format=pdf
```

### response

Regular file download, or JSON with an error.

### errors

- 201 – Not found in the database - if there is no form with such an ID.

Last update: August 1, 2023





# Form fields and values

Every form (and form template) contains an ordered list of fields of various types. Field type defines how user input elements will look like, and how user input will be validated.

Every field has a set of common parameters, which are the same for all field types, and type-specific parameters, which define specific style and validation constraints. Both common and type-specific parameters contained as fields in the JSON object.

Field values for submitted form stored separately as JSON objects. The contents of value JSON objects are entirely field type-specific.

## COMMON FIELD PARAMETERS:

```
{
  "id": "Text-1",
  "label": "Name",
  "description": "Your full name",
  "required": true,
  "type": "text"
}
```

- `id` - arbitrary alphanumeric string (1 to 19 characters). Unique across current form's fields, used to link with values and its "parent" in template form.
- `label` - string. User-defined label, shown as field header, 1 to 100 printable characters.
- `description` - string. Field description, shown in smaller text under the header, 1 to 512 printable characters.
- `required` - boolean. If `true`, form cannot be submitted without filling this field with valid value.
- `type` - string. Determines field type.

## Text field

**type:** `text`.

Multiline auto-expanding text field.

**Note 1:** when value contains empty string, it's considered empty, and thus valid when `required: false, min_length != 0`.

**Note 2:** combination `required: true, min_length: 0` is not allowed.

### type-specific parameters:

```
{
  "min_length": 5,
  "max_length": 255
}
```

- `min_length` - int. Minimum allowed length, from 0 to 1024.
- `max_length` - int. Maximum allowed length 1 to 1024.

### value object:

```
{
  "type": "text",
  "value": "text field value"
}
```

- `value` - string. What was entered the text field.

## Checkbox group

**type:** `checkbox_group`.

Group of checkboxes.

**Note 1:** when zero checkboxes selected, values considered empty, and thus valid when `required: false, min_checked != 0`.

**Note 2:** combination `required: true, min_checked: 0` is not allowed.

### TYPE-SPECIFIC PARAMETERS:

```
{
  "min_checked": 0,
  "max_checked": 3,
  "group": [{
    "label": "I agree to TOS"
  }]
}
```

- `min_checked` - int. Minimum allowed checked positions, 0 to "group".size - 1.
- `max_checked` - int. Maximum allowed checked positions, 1 to "group".size - 1.

### VALUE OBJECT:

```
{
  "type": "checkbox_group",
```

```
    "values": [true]
}
```

- `values` - array of boolean. They are in the same order as fields in `group`.

## Dropdown field

**type:** `dropdown`.

Dropdown menu for choosing one option.

### TYPE-SPECIFIC PARAMETERS:

```
{
  "options": [
    {
      "label" : "John"
    },
    {
      "label" : "Alice"
    }
  ]
}
```

### VALUE OBJECT:

```
{
  "type": "dropdown",
  "value_index": 1
}
```

- `value_index` - int. Zero-based index of value from "options".

## Radio button group

**type:** `radio_group`.

A group of radio buttons. Only one option is selectable.

### TYPE-SPECIFIC PARAMETERS:

```
{
  "options": [
    {
      "label" : "John"
    },
    {
      "label" : "Alice"
    }
  ]
}
```

```
}  
]  
}
```

#### VALUE OBJECT:

```
{  
  "type": "radio_group",  
  "value_index": 1  
}
```

- `value_index` - int. Zero-based index of value from "options".

## Date picker

**type:** `date`.

A date picker.

#### TYPE-SPECIFIC PARAMETERS:

```
{  
  "disable_future": false,  
  "disable_past": true  
}
```

- `disable_future` - boolean. If `true`, date from the future cannot be selected.
- `disable_past` - boolean. If `true`, date from the past cannot be selected.

#### VALUE OBJECT:

```
{  
  "type": "date",  
  "value": "2017-03-14"  
}
```

- `value` - [date/time](#).

## Rating

**type:** `rating`.

Rating with "stars". Zero stars not allowed.

#### TYPE-SPECIFIC PARAMETERS:



```
{
  "max_stars": 5
}
```

- `max_stars` - int. Max number of stars to select from.

#### VALUE OBJECT:

```
{
  "type": "rating",
  "value": 3
}
```

- `value` - int. Number of stars selected. Cannot be more than `max_stars`.

## File

**type:** `file`.

File attachment. For example, document or spreadsheet.

#### TYPE-SPECIFIC PARAMETERS:

```
{
  "max_file_size": 65536,
  "min_file_size": 128,
  "allowed_extensions": ["xls", "doc"]
}
```

- `max_file_size` - int. Max file size, bytes, no more than 16 Mb.
- `min_file_size` - int. Minimum file size, bytes.
- `allowed_extensions` - [enum](#) array. List of allowed file extensions, up to 16 items, cannot be empty, but can be null, which means "no extension limits".

#### VALUE OBJECT:

```
{
  "type": "file",
  "file_ids": [3345345]
}
```

- `file_ids` - int array. IDs of the file which should be attached to this form field as value. Files must be uploaded before form submission.

## Photo

**type:** photo.

Photograph attachment.

### TYPE-SPECIFIC PARAMETERS:

```
{
  "max_files": 2
}
```

- `max_files` - int. Maximum number of photos to attach, up to 6.

### VALUE OBJECT:

```
{
  "type": "photo",
  "file_ids": [3345345, 534534534]
}
```

- `file_ids` - int array. IDs of the files which should be attached to this form field as value. Files must be uploaded before form submission. Only image files allowed.

## Signature

**type:** signature.

A small image of customer's signature (usually obtained via writing on screen with a stylus).

### TYPE-SPECIFIC PARAMETERS:

- there are no type-specific parameters.

### VALUE OBJECT:

```
{
  "type": "file",
  "file_id": 3345345
}
```

- `file_id` - int. An ID of the file which should be attached to this form field as value. File must be uploaded before form submission.

## Separator

**type:** `separator`.

Cosmetic, just to show header. Doesn't contain any actual value. Always filled and valid.  
Cannot be required.

Last update: December 26, 2022





# Form templates

Form is a "one-shot" entity; after it was filled by someone, it cannot be reused. It's stored along with filled fields for future reference. Usually people need to fill forms with the same fields over and over again, so forms created on the basis of form templates. It's similar to paper forms: each paper form can be filled only once, but there's an electronic document, a template, on basis of which all paper forms printed.

The reason for such API design is that template fields can be changed over time (deleted, removed, reordered, etc.) and it should not affect already filled forms. By separating filled forms and templates, one can always view filled form in exactly same state regardless of how template changed.

User can assign form to the task or checkin by choosing template without the need to create all form fields every time.

## Form template object

```
{
  "id": 1,
  "label": "Order form",
  "fields": [{
    "id": "Text-1",
    "label": "Name",
    "description": "Your full name",
    "required": true,
    "type": "text",
    "min_length": 5,
    "max_length": 255
  }],
  "created": "2017-03-15 12:36:27",
  "submit_in_zone": true,
  "updated": "2017-03-16 15:22:53",
  "default": false
}
```

- `id` - int. An ID of a template.
- `label` - string. User-defined template label, from 1 to 100 characters.
- `fields` - array of multiple [form\\_field](#) objects.
- `created` - [date/time](#). Date when this template created. The read-only field.
- `submit_in_zone` - boolean. If `true`, form can be submitted only in task zone.

- `updated` - [date/time](#). Date when this template last modified. The read-only field.
- `default` - boolean. This form will be chosen default for all new tasks with form if `true`.

## API actions

API path: `/form/template`.

### list

Gets all form templates belonging to current master user.

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/form/template/list' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/form/template/list?  
hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{  
  "success": true,  
  "list": [{  
    "id": 1,  
    "label": "Order form",  
    "fields": [{  
      "id": "Text-1",  
      "label": "Name",  
      "description": "Your full name",  
      "required": true,  
      "type": "text",  
      "min_length": 5,  
      "max_length": 255  
    }],  
    "created": "2017-03-15 12:36:27",  
    "submit_in_zone": true,  
    "updated": "2017-03-16 15:22:53",  
    "default": false  
  }]  
}
```

- `list` - ordered array of [form\\_template](#) objects.

## errors

[General](#) types only.

## create

Creates new form template.

**required sub-user rights:** `form_template_update`.

## parameters

| name     | description   | type        |
|----------|---|-------------|
| template | Non-null form template object without <code>id</code> , <code>created</code> , <code>updated</code> fields. | JSON object |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/form/template/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "template":
{"label": "Order form", "fields": [{"id": "Text-1", "label":
"Name", "description": "Your full name", "required": true, "type":
"text", "min_length": 5, "max_length": 255}], "submit_in_zone":
true, "default": false}}'
```

## response

```
{
  "success": true,
  "id": 111
}
```

- `id` - int. An ID of the created form template.

## errors

- 101 – In demo mode this function disabled - if current user has "demo" flag.

## read

Gets form template belonging to current master user by specified ID.



## parameters

| name        | description              | type |
|-------------|--------------------------|------|
| template_id | ID of the form template. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/form/template/read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",  
    "template_id": 111}'
```

### HTTP GET

```
https://api.navixy.com/v2/form/template/read?  
hash=a6aa75587e5c59c32d347da438505fc3&template_id=111
```

## response

```
{  
  "success": true,  
  "list": [{  
    "id": 1,  
    "label": "Order form",  
    "fields": [{  
      "id": "Text-1",  
      "label": "Name",  
      "description": "Your full name",  
      "required": true,  
      "type": "text",  
      "min_length": 5,  
      "max_length": 255  
    }],  
    "created": "2017-03-15 12:36:27",  
    "submit_in_zone": true,  
    "updated": "2017-03-16 15:22:53",  
    "default": false  
  }]  
}
```

- `list` - ordered array of [form\\_template](#) objects.

## errors

- 201 – Not found in the database - if there is no template with such an ID.

## update

Updates existing form template.

**required sub-user rights:** `form_template_update`.

### parameters

| name     | description   | type        |
|----------|---|-------------|
| template | Non-null form template object without <code>created</code> , <code>updated</code> fields. | JSON object |

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/form/template/update' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "template":   
{"id": 111, "label": "Order form", "fields": [{ "id": "Text-1",   
"label": "Name", "description": "Your full name", "required":   
true, "type": "text", "min_length": 5, "max_length": 255}],   
"submit_in_zone": true, "default": false}}'
```

### response

```
{ "success": true }
```

### errors

- 201 – Not found in the database - if template with the specified ID does not exist.
- 101 – In demo mode this function disabled - if current user has "demo" flag.

## delete

Deletes form template.

**required sub-user rights:** `form_template_update`.

## parameters

| name        | description              | type |
|-------------|--------------------------|------|
| template_id | ID of the form template. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/form/template/delete' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",  
    "template_id": 111}'
```

### HTTP GET

```
https://api.navixy.com/v2/form/template/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&template_id=111
```

## response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - if template with the specified ID does not exist.
- 101 – In demo mode this function disabled - if current user has "demo" flag.

## stats/read

Returns template usage statistics.

**required sub-user rights:** none.

## parameters

| name        | description              | type |
|-------------|--------------------------|------|
| template_id | ID of the form template. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/form/template/stats/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
    "template_id": 111}'
```

### HTTP GET

```
https://api.navixy.com/v2/form/template/stats/read?
hash=a6aa75587e5c59c32d347da438505fc3&template_id=111
```

## response

```
{
  "success": true,
  "tasks": {
    "unassigned": 0,
    "assigned": 6,
    "done": 0,
    "failed": 0,
    "delayed": 9,
    "arrived": 0,
    "faulty": 0
  },
  "scheduled": 2
}
```

- **tasks** - maps task status to number of tasks with this status which use specified template.
- **scheduled** - int. Number of task schedules using this template.

## errors

- 201 – Not found in the database - if template with the specified ID does not exist.

Last update: January 22, 2023





# Working with points of interest (POI)

"Places" are business-specific points of interest (POI) like shops, delivery points, warehouses, etc. - which are visited by user's employees. Place entities can be extended with [custom fields](#) to make them even more useful.

In case an event happened at the POI, in various reports name of the POI will be specified after the address.

If there's an [employee assigned](#) to a Mobile Tracker App ([Android](#) / [iOS](#)), and a POI has a custom field of type "responsible employee", such point of interest will be available in the mobile app to view. Thus, field employee/driver can view all points of interest assigned to him to visit them, etc.

Working with POIs requires several actions so we described them in our [guides](#).

## Place object

```
{
  "id": 1,
  "icon_id": 55,
  "avatar_file_name": null,
  "location": {
    "lat": 52.366,
    "lng": 4.895,
    "address": "730 5th Ave, New York, NY 10019, Unites States",
    "radius": 500
  },
  "fields": {
    "131312": {
      "type": "text",
      "value": "I love text!"
    }
  },
  "label": "Crown Building",
  "description": "Here we buy our goods",
  "tags": [ 1, 2 ],
  "external_id": "1"
}
```

- `id` - int. An ID of a POI.
- `icon_id` - optional int. Can be 1 to 255. Can only be updated via [avatar/assign](#).
- `avatar_file_name` - optional string. Name of the avatar file. Can be null.

- `location` - required information about place location.
  - `lat` - required, float. The latitude.
  - `lng` - required, float. The longitude.
  - `address` - required, string, max length 255. The address of place.
  - `radius` - required, int, 1..300000. The radius of place in meters.
- `fields` - optional object. A map, each key of which is a custom field ID as a *string*. See [entity/fields](#)
- `label` - string. POI name.
- `description` - optional string. POI description.
- `tags` - optional int array. A list of tag\_ids. Non-empty.
- `external_id` - optional string. Max length 32.

## API actions

API path: `/place`.

### read

Gets POI by ID.

#### parameters

| name     | description    | type |
|----------|----------------|------|
| place_id | ID of the POI. | int  |

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/place/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "place_id": 122304}'
```

##### HTTP GET

```
https://api.navixy.com/v2/place/read?
hash=a6aa75587e5c59c32d347da438505fc3&place_id=122304
```

#### response

---



```

{
  "success": true,
  "value": {
    "id": 1,
    "icon_id": 55,
    "avatar_file_name": null,
    "location": {
      "lat": 40.773998,
      "lng": -73.66003,
      "address": "730 5th Ave, New York, NY 10019, Unites States",
      "radius": 50
    },
    "fields": {
      "131312": {
        "type": "text",
        "value": "I love text!"
      }
    },
    "label": "Crown Building",
    "description": "Here we buy our goods",
    "tags": [ 1, 2 ],
    "external_id": "1"
  }
}

```

## errors

- 201 - Not found in the database – if there is no POI with such ID.

## list

Get POIs belonging to user.

## parameters

| name       | description   | type             |
|------------|---|------------------|
| place_ids  | Optional. List of POI IDs.  | int array        |
| filter     | Optional. Filter for all built-in and custom fields. If used with conditions, both filter and conditions must match for every returned POI. | string           |
| conditions | Optional. Search conditions to apply to list. Array of search conditions, see <a href="#">Search conditions</a> .                           | array of objects |
| order_by   |   | string           |

| name      | description   | type      |
|-----------|---|-----------|
|           | Optional. Built-in or custom field according to which output should be sorted. Entity field name, e.g "label" (builtin) or "123" (field ID as string, see <a href="#">entity/</a> . |           |
| ascending | Optional. If <code>false</code> – descending order.   | boolean   |
| limit     | Optional. Limit.  | int       |
| offset    | Optional. offset, default is 0.   | int       |
| tag_ids   | Optional. Tag IDs assigned to the place. The places found must include all the tags from the list.  | int array |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/place/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/place/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [
    {
      "id": 1,
      "icon_id": 55,
      "avatar_file_name": null,
      "location": {
        "lat": 40.773998,
        "lng": -73.66003,
        "address": "730 5th Ave, New York, NY 10019, Unites
States",
        "radius": 50
      },
      "fields": {
        "131312": {
          "type": "text",
          "value": "I love text!"
        }
      }
    }
  ],
}
```

```
    "label": "Crown Building",
    "description": "Here we buy our goods",
    "tags": [ 1, 2 ],
    "external_id": "1"
  }
],
"count": 1
}
```

- `count` - int. Found POIs count.

## errors

General types only.

## create

Creates a new POI.

**required sub-user rights:** `place_update`.

## parameters

| name                  | description   | type        |
|-----------------------|---|-------------|
| place                 | A place object without <code>id</code> field.   | JSON object |
| ignore_missing_fields | Optional (default is false). If <code>true</code> , POI can be created even without all required custom fields. | boolean     |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/place/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "place":
{"icon_id" : 55, "avatar_file_name": null, "location": {"lat":
40.773998, "lng": -73.66003, "address": "730 5th Ave, New York, NY
10019, Unites States", "radius": 50}, "fields": {"131312":
{"type": "text", "value": "I love text!"}}, "label": "Crown
Building", "description": "Here we buy our goods", "tags": [1, 2],
"external_id": "1"}'
```

## response

```
{
  "success": true,
  "id": 111
}
```

- `id` - int. An ID of the created POI.

## errors

- 268 - Over quota – if the user's quota for POIs exceeded.

## search\_location

Gets all POI IDs and names within which a specified coordinates are located inside.

## parameters

| name     | description   | type        |
|----------|---|-------------|
| location | Location coordinates (see: <a href="#">data types description section</a> section). | JSON object |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/place/search_location' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "location": {"lat": 34.178868, "lng": -118.599672}}'
```

## response

```
{
  "list": [
    {
      "id": 1201,
      "label": "place 1"
    },
    {
      "id": 3574,
      "label": "place 2"
    }
  ],
  "success": true
}
```

- `id` - int. Place ID that containing a searched location.

- `label` - string. Place name.

## update

Updates existing POI.

**required sub-user rights:** `place_update`.

### parameters

| name  | description     | type        |
|-------|-----------------|-------------|
| place | A place object. | JSON object |

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/place/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "place":
{"id": 111, "icon_id" : 55, "avatar_file_name": null, "location":
{"lat": 40.773998, "lng": -73.66003, "address": "730 5th Ave, New
York, NY 10019, Unites States", "radius": 50}, "fields":
{"131312": {"type": "text", "value": "I love text!"}}, "label":
"Crown Building", "description": "Here we buy our goods", "tags":
[1, 2], "external_id": "1"}'
```

### response

```
{ "success": true }
```

### errors

- 201 - Not found in the database – if there is no POI with such ID.

## delete

Deletes POI with the specified ID.

**required sub-user rights:** `place_update`.

## parameters

| name     | description              | type |
|----------|--------------------------|------|
| place_id | ID of the POI to delete. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/place/delete' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "place_id":  
122304}'
```

### HTTP GET

```
https://api.navixy.com/v2/place/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&place_id=122304
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if there is no POI with such ID.

## batch\_convert

Converts batch of tab-delimited POIs and return list of checked POIs with errors.

**Required sub-user rights:** place\_update.

## parameters

| name    | description   | type            |
|---------|---|-----------------|
| batch   | Batch of tab-delimited POIs.  | string          |
| file_id | Preloaded file ID.  | string          |
| fields  | Optional. Array of field names, default is ["label", "address", "lat", "lng", "radius", "description", "tags"]. | string<br>array |

| name           | description   | type   |
|----------------|---|--------|
| geocoder       | Geocoder type.  | string |
| default_radius | Optional. Radius for point in meters. Default is 100. | int    |

If `file_id` is set – `batch` parameter will be ignored.

### response

```
{
  "success": true,
  "list": [
    {
      "id": 1,
      "icon_id": 55,
      "avatar_file_name": null,
      "location": {
        "lat": 40.773998,
        "lng": -73.66003,
        "address": "730 5th Ave, New York, NY 10019, Unites
States",
        "radius": 50
      },
      "fields": {
        "131312": {
          "type": "text",
          "value": "I love text!"
        }
      },
      "label": "Crown Building",
      "description": "Here we buy our goods",
      "tags": [ 1, 2 ],
      "external_id": "1",
      "errors": <array_of_objects>,
      "tag_names": <array_of_strings>
    }
  ],
  "limit_exceeded": false
}
```

- `list` - a list of objects.
  - `errors` - optional array of objects. Errors found during check.
  - `tag_names` - optional string array. Tag names of the POI.
- `limit_exceeded` - boolean. `true` if given batch constrained by a limit.

### errors

- 234 - Invalid data format.

## upload

Upload POIs.

**Required sub-user rights:** `place_update`.

**MUST** be a POST multipart request (multipart/form-data), with one of the parts being a CSV file upload (with the name "file").

CSV column separator is `;`, columns header required –  
`label;address;lat;lng;radius;external_id;description`.

### parameters

| name             | description  | type        |
|------------------|--|-------------|
| file             | A CSV file upload containing POIs data.  | File upload |
| error_policy     | <code>ignore</code> or <code>fail</code> .   | string      |
| duplicate_policy | <code>skip</code> or <code>update</code> or <code>fail</code> , <b>belongs only to external_id duplicates.</b>   | string      |
| default_radius   | Optional, radius for point, meters, default is 100.  | int         |
| geocoder         | Geocoder type.   | string      |
| redirect_target  | Optional URL to redirect. If <code>redirect_target</code> passed return redirect to <code>&lt;redirect_target&gt;?response=&lt;urlencoded_response_json&gt;</code> . | string      |

### response

```
{
  "success": true,
  "total": 1,
  "errors": 0
}
```

### errors

- 233 - No data file – if file part is missing.



- 234 - Invalid data format.
- 247 - Entity already exists – if uploaded POI contains `external_id` and POI with this ID already exists and `duplicate_policy=fail`.
- 268 - Over quota – if the user's quota for POIs exceeded.

Last update: January 15, 2024





# Changing POI avatar

Avatars don't change through `/place/update`, you must use either `assign` (to set avatar to one of preset icons), or `upload` (to upload your own image).

## API actions

### upload

Uploads avatar image for specified POI.

**required sub-user rights:** `place_update`.

Then it will be available from `[api_base_url]/<api_static_uri>/place/avatars/<file_name>` e.g. `https://api.navixy.com/v2/static/place/avatars/abcdef123456789.png`.

**avatar\_file\_name** returned in response and will be returned from [place/list](#).

**MUST** be a POST multipart request (multipart/form-data), with one of the parts being an image file upload (with the name "file").

File part **mime** type must be one of:

- `image/jpeg`
- `image/pjpeg`
- `image/png`
- `image/gif`
- `image/webp`

### PARAMETERS

| name            | description    | type        |
|-----------------|----------------|-------------|
| place_id        | ID of the POI. | int         |
| file            | Image file.    | File upload |
| redirect_target |                | string      |

| name | description  | type |
|------|--|------|
|      | Optional URL to redirect. If <b>redirect_target</b> passed return redirect to <code>&lt;redirect_target&gt;?response=&lt;urlencoded_response_json&gt;</code> . |      |

#### RESPONSE

```
{
  "success": true,
  "value": "Avatar file name"
}
```

- `value` - string. Avatar file name.

#### errors

- 201 - Not found in the database – when POI with `place_id` not found.
- 233 - No data file – if file part not passed.
- 234 - Invalid data format – if passed file with unexpected mime type.
- 254 - Cannot save file – on some file system errors.

#### assign

Assigns `icon_id` (from standard icon set) to this POI. `icon_id` can be null – this means that uploaded avatar should be used instead of icon.

**required sub-user rights:** `place_update`.

#### parameters

| name                  | description                                      | type |
|-----------------------|--|------|
| <code>place_id</code> | ID of the POI.                                   | int  |
| <code>icon_id</code>  | Optional. ID of the icon from standard icon set. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/place/avatar/assign' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "place_id":  
122304, "icon_id": 1}'
```

### HTTP GET

```
https://api.navixy.com/v2/place/avatar/assign?  
hash=a6aa75587e5c59c32d347da438505fc3&place_id=122304&icon_id=1
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – when POI with `place_id` not found.

Last update: December 26, 2022







## Working with tasks

You can assign task to any tracked device. If specified tracker visits task checkpoint at the specified time and meets other conditions such as filling form or staying in the task zone for the specified time, the task completed. Otherwise, the task either failed completely or completed with warnings.

If task assigned to a Mobile Tracker App ([Android](#) / [iOS](#)), it's available for viewing by app user. User will also receive notifications of newly assigned tasks, task changes, etc.

Find more guides on working with tasks [there](#).

### Task object

```
{
  "id": 111,
  "user_id": 3,
  "tracker_id": 22,
  "location": {
    "lat": 51.283546,
    "lng": 7.301086,
    "address": "Fichtenstrasse 11",
    "radius": 150
  },
  "label": "Deliver parcels",
  "description": "Quickly",
  "creation_date": "2014-01-02 03:04:05",
  "from": "2014-02-03 04:05:06",
  "to": "2014-03-04 05:06:07",
  "external_id": null,
  "status": "assigned",
  "status_change_date": "2014-01-02 03:04:05",
  "max_delay": 5,
  "min_stay_duration": 0,
  "arrival_date": "2014-01-02 03:04:05",
  "stay_duration": 0,
  "origin": "imported",
  "tags": [1, 2],
  "type": "task",
  "form": <form_object>,
  "form_template_id": 13245,
  "fields": {
    "131312": {
      "type": "text",
      "value": "I love text!"
    }
  }
}
```

```
}  
}
```

- `id` - int. Primary key. Used in task/update, *IGNORED* in task/create.
- `user_id` - int. User ID. *IGNORED* in create/update.
- `tracker_id` - int. An ID of the tracker to which task assigned. Can be null. *IGNORED* in task/update.
- `location` - location associated with this task. Cannot be null.
  - `address` - string. Address of the location.
  - `radius` - int. Radius of location zone in meters.
- `creation_date` - [date/time](#). When task created. *IGNORED* in create/update.
- `from` - [date/time](#). Date AFTER which task zone must be visited.
- `to` - [date/time](#). Date BEFORE which task zone must be visited.
- `external_id` - string. Used if task imported from external system. Arbitrary text string. Can be null.
- `status` - [enum](#). Task status. *IGNORED* in create/update. Can have "unassigned" value (unassigned to any executor), "assigned", "done", "failed", "delayed", "arrived" (arrived to geofence but haven't done the task), "faulty" (with problems).
- `status_change_date` - [date/time](#). When task status changed. *IGNORED* in create/update.
- `max_delay` - int. Maximum allowed task completion delay in minutes.
- `min_stay_duration` - int. Minimum duration of stay in task zone for task completion, minutes.
- `arrival_date` - [date/time](#). When tracker has arrived to the task zone. *IGNORED* in create/update.
- `stay_duration` - int. Duration of stay in the task zone, seconds.
- `origin` - string. Task origin. *IGNORED* in create/update.
- `tags` - int array. List of tag IDs.
- `form` - [form object](#). If present.
- `form_template_id` - int. An ID of form template. Used in create and update actions only if `create_form` parameter is `true` in them.
- `fields` - optional object. A map, each key of which is a custom field ID as a *string*. See [entity/fields](#)

**To associate the task with an address - this field should be added to the location object.**

## API actions

API base path: `/task`.

### assign

(Re)assigns task to new tracker (or make it unassigned).

**required sub-user rights:** `task_update`.

#### parameters

| name       | description   | type |
|------------|---|------|
| task_id    | ID of the task to assign.   | int  |
| tracker_id | ID of the tracker. Tracker must belong to authorized user and not be blocked. If null, task will be assigned to no one. | int  |

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/assign' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id": 23144, "tracker_id": 132421}'
```

##### HTTP GET

```
https://api.navixy.com/v2/task/assign?
hash=a6aa75587e5c59c32d347da438505fc3&task_id=23144&tracker_id=132421
```

#### response

```
{ "success": true }
```

#### errors

- 201 – Not found in the database (if there is no task with such an ID).
- 204 – Entity not found (if there is no tracker with such ID belonging to authorized user).
- 208 – Device blocked (if tracker exists but was blocked due to tariff restrictions or some other reason).

- 255 – Invalid task state (if current task state is not "unassigned" or "assigned").
- 236 – Feature unavailable due to tariff restrictions (if device's tariff does not allow usage of tasks).

## batch\_convert

Converts batch of tab-delimited tasks and return list of checked tasks with errors.

**required sub-user rights:** `task_update`.

### parameters

| name                      | description   | type         |
|---------------------------|---|--------------|
| batch                     | Batch of tab-delimited tasks.   | string       |
| fields                    | Optional. Array of field names, default is <code>["label", "from", "to", "address", "lat", "lng", "description"]</code> . | string array |
| geocoder                  | Geocoder type.  | enum         |
| default_radius            | Optional. Radius for point, default is 100.   | int          |
| default_max_delay         | Optional. Max delay for tasks, default is 0.  | int          |
| default_duration          | Optional. Duration for task in minutes, default is 60.  | int          |
| default_min_stay_duration | Optional. Minimal stay duration for task in minutes, default is 0.  | int          |
| location_check_mode       | Optional. One of "no_check", "entity_location", "parent_location"   | enum         |
| employee_ids              | Optional. List of employee IDs to automatic assign  | int array    |
| vehicle_ids               | Optional. List of vehicle IDs to automatic assign   | int array    |

In case of `location_check_mode==entity_location` – `vehicle_ids` will be ignored.

## response

```
{
  "success": true,
  "list": [{
    "id": 111,
    "user_id": 3,
    "tracker_id": 22,
    "location": {
      "lat": 51.283546,
      "lng": 7.301086,
      "address": "Fichtenstrasse 11",
      "radius": 150
    },
    "label": "Deliver parcels",
    "description": "Quickly",
    "creation_date": "2014-01-02 03:04:05",
    "from": "2014-02-03 04:05:06",
    "to": "2014-03-04 05:06:07",
    "external_id": null,
    "status": "assigned",
    "status_change_date": "2014-01-02 03:04:05",
    "max_delay": 5,
    "min_stay_duration": 0,
    "arrival_date": "2014-01-02 03:04:05",
    "stay_duration": 0,
    "origin": "imported",
    "tags": [1, 2],
    "type": "task",
    "form": <form_object>,
    "errors": [<error_object>]
  }],
  "limit_exceeded": false
}
```

- `list` - list of checked task objects that contain all fields from task and field `errors`.
- `errors` - array of objects. Optional. List of errors.
- `limit_exceeded` - boolean. `true` if given batch constrained by a limit.

## errors

[General](#) types only.

## count

Returns total number of tasks belonging to current user.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/count' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/count?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "count": 111
}
```

- `count` - int. Number of tasks.

## create

Creates a new task.

**required sub-user rights:** `task_update`.

### parameters

| name        | description   | type        |
|-------------|---|-------------|
| task        | <code>task</code> object without fields which are <b>IGNORED</b>  | JSON object |
| create_form | If <code>true</code> then check additional <code>form_template_id</code> field in <code>task</code> object and create form if it is not null. Default value is <code>false</code> for backward compatibility. | boolean     |

Minimal JSON object to create a new task must contain:

```
{
  "tracker_id": 22,
  "location": {
    "lat": 34.178868,
    "lng": -118.599672,
    "radius": 150
  },
}
```

```

    "label": "Name",
    "description": "Description example",
    "from": "2020-02-03 04:05:06",
    "to": "2020-03-04 05:06:07"
  }

```

- `tracker_id` - int. Optional. if the field specified then the task will be assigned to the employee associated with the tracker, otherwise it won't be assigned to anybody.
- `location` - area (circle geofence), entering and leaving of geofence will be controlled.
  - `lat` - float. Latitude.
  - `lng` - float. Longitude.
  - `radius` - int. Radius in meters.
- `label` - string. Task name, length 1-200 characters.
- `description` - string. Task description, length 0-1024 characters.
- `from` - [date/time](#). Start date of the interval - when the specified location has to be visited (in the user's time zone).
- `to` - [date/time](#). End date of the interval - when the specified location has to be visited (in the user's time zone).

## example

### cURL

```

curl -X POST 'https://api.navixy.com/v2/task/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task": {"tracker_id": 22, "location": {"lat": 34.178868, "lng": -118.599672, "radius": 150}, "label": "Name", "description": "Description example", "from": "2020-02-03 04:05:06", "to": "2020-03-04 05:06:07"}, "create_form": false}'

```

task/create call returns the identifier of the created task. A returned object also can include "external\_id\_counts" field see task/route/create [method description](#).

## response

```

{
  "success": true,
  "id": 111,
  "external_id_counts": [{
    "external_id": "456",
    "count": 2
  }]
}

```

```
} ]  
}
```

- `id` - int. An ID of the created task.

**Note:** The "id" parameter is unique, it is automatically generated by the server when you create a task. Therefore, if you call task/create two times with the same parameters, every time the new task will be created. These two tasks will differ only by an ID. Respectively, if the created task has to be connected to a certain record in external system, you have to remember the ID of this record to use it in future when you want to change/delete the associated task in our system.

#### errors

- 201 – Not found in the database (if task.tracker\_id is not null and belongs to nonexistent tracker).
- 236 – Feature unavailable due to tariff restrictions (if device's tariff does not allow usage of tasks).

## delete

Deletes the task with the specified ID.

**required sub-user rights:** `task_update`.

#### parameters

| name    | description               | type |
|---------|---------------------------|------|
| task_id | ID of the task to delete. | int  |

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/delete' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id":  
23144}'
```

##### HTTP GET

```
https://api.navixy.com/v2/task/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&task_id=23144
```

#### response



```
{ "success": true }
```

## errors

- 201 – Not found in the database (if there is no task with such an ID).

## list

Gets all task belonging to user with optional filtering.

## parameters

| name        | description  | type                                     |
|-------------|--|--|
| external_id | Optional. External task ID for search.   | string                                   |
| statuses    | Optional. Default all. List of task statuses, e.g. <code>["unassigned", "failed"]</code> .   | string array                             |
| trackers    | Optional. IDs of the trackers to which task assigned.  | int array                                |
| from        | Optional. Show tasks which are actual AFTER this date, e.g. "2020-07-01 00:00:00".   | <a href="#">date/time</a>                |
| to          | Optional. Show tasks which are actual BEFORE this date, e.g. "2020-07-01 00:00:00".  | <a href="#">date/time</a>                |
| conditions  | Optional. Search conditions to apply to list. Array of search conditions.  | array of <a href="#">SearchCondition</a> |
| filter      | Optional. Filter for all built-in and custom fields. If used with conditions, both filter and conditions must match for every returned task. | string                                   |
| filters     | Optional. Filters for task label, description or address.  | string array                             |
| tag_ids     | Optional. Tag IDs assigned to the task.  | int array                                |

| name     | description  | type          |
|----------|--|---------------|
| location | Optional. Location with radius, inside which task zone centers must reside. Example:<br><pre>{ "lat": 34.178868, "lng": -118.599672, "radius": 350 }</pre> | Location JSON |
| offset   | Optional. Offset from start of the found tasks for pagination.   | int           |
| limit    | Optional. Limit of the found tasks for pagination.   | int           |

#### CONDITION FIELDS

| Name               | Type      | Comment |
|--------------------|-----------|---------|
| id                 | int       |         |
| employee           | int       | ID      |
| status             | string    |         |
| label              | string    |         |
| location           | string    | address |
| from               | date/time |         |
| to                 | date/time |         |
| status_change_date | date/time |         |
| arrival_date       | date/time |         |
| stay_duration      | Seconds   |         |
| description        | string    |         |
| external_id        | string    |         |

| Name | Type   | Comment       |
|------|--------|---------------|
| form | number | template's ID |

If **external\_id**, **trackers**, **filters**, **from**, **to** or **tag\_ids** is not passed or *null* then appropriate condition not used to filter results.

If **offset** or **limit** is null then restrictions for pagination will not be applied.

**SORT: STRING[]?**

set of sort options. Each option is a pair of column name and sorting direction, e.g. ["label=asc", "address=desc", "employee=desc"].

**SORT FIELDS**

| Name               | Type      | Comment                    |
|--------------------|-----------|----------------------------|
| id                 | int       |                            |
| employee           | string    | full name or tracker label |
| status             | string    |                            |
| label              | string    |                            |
| location           | string    | address                    |
| from               | date/time |                            |
| to                 | date/time |                            |
| status_change_date | date/time |                            |
| arrival_date       | date/time |                            |
| stay_duration      | Seconds   |                            |
| description        | string    |                            |
| external_id        | string    |                            |

| Name | Type   | Comment |
|------|--------|---------|
| form | string | label   |

If **external\_id**, **trackers**, **filters**, **from**, **to** or **tag\_ids** is not passed or *null* then appropriate condition not used to filter results.

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/task/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{
  "success": true,
  "list": [{
    "id": 111,
    "user_id": 3,
    "tracker_id": 22,
    "location": {
      "lat": 48.210857,
      "lng": 16.369329,
      "address": "Schulhof 2, Wien, Austria",
      "radius": 150
    },
    "label": "Name",
    "description": "Description example",
    "creation_date": "2014-01-02 03:04:05",
    "from": "2020-02-03 04:05:06",
    "to": "2020-03-04 05:06:07",
    "external_id": "01234567",
    "status": "assigned",
    "status_change_date": "2020-01-02 03:04:05",
    "max_delay": 5,
    "min_stay_duration": 0,
    "arrival_date": "2020-01-02 03:04:05",
    "stay_duration": 10,
    "origin": "manual",
    "type": "task"
  }],
  "count": 1
}
```

- `list` - array of `task` objects.
- `id` - int. Task ID.

- `user_id` - int. User ID (office). An unchangeable parameter.
- `tracker_id` - int. Tracker ID. Indicator ID by which the implementation of this task will be controlled.
- `location` - area (circle geofence), entering and leaving of geofence will be controlled.
- `label` - string. Task name, length 1-200 characters.
- `description` - string. Task description, length 0-1024 characters.
- `creation_date` - [date/time](#). Date of creation of a task, unchangeable field.
- `from` - [date/time](#). Start date of the interval - when the specified location has to be visited (in the user's time zone).
- `to` - [date/time](#). End date of the interval - when the specified location has to be visited (in the user's time zone).
- `external_id` - string. Text field for tracking of communication of the task with certain external systems (for example, number of the order). Is for reference only.
- `status` - [enum](#). Current status of a task, can have "unassigned" value (unassigned to any executor), "assigned", "done", "failed", "delayed", "arrived" (arrived to geofence but haven't done the task), "faulty" (with problems).
- `status_change_date` - [date/time](#). Date of the last change of the status of a task.
- `max_delay` - int. The maximum time delay of the execution of the task, in minutes.
- `min_stay_duration` - int. The minimum stay time in the area of the task in which the task has to be done, in minutes.
- `arrival_date` - [date/time](#). Date and time of arrival in the area of the task. Can be null. If the executor has not visited it yet.
- `stay_duration` - int. Number of seconds spent inside task zone.
- `origin` - [enum](#). The way of creation of a task. Can be "manual", "scheduled" or "imported" (from excel).
- `type` - string. Reserved.
- `count` - int. count of the all found tasks.

## errors

[General](#) types only.

## read

Gets task, checkpoint, or route with checkpoints by specified ID.

### parameters

| name    | description                          | type |
|---------|--------------------------------------|------|
| task_id | ID of the task, route or checkpoint. | int  |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id":  
23144}'
```

#### HTTP GET

```
https://api.navixy.com/v2/task/read?  
hash=a6aa75587e5c59c32d347da438505fc3&task_id=23144
```

### response

```
{  
  "success": true,  
  "value": <task or checkpoint or route>,  
  "checkpoints": [  
    <checkpoint1>,  
    <checkpoint2>  
  ]  
}
```

- `value` - JSON object. `task` described [here](#).
- `checkpoints` - only returned if entity with specified ID is a route. Contains all checkpoints of this route. `checkpoint` object described [here](#).

### errors

- 201 – Not found in the database (if there is no task with such an ID).

## transmute

Converts task into a route checkpoint.

**required sub-user rights:** `task_update`.

#### parameters

| name     | description   | type |
|----------|---|------|
| task_id  | ID of the task to convert.  | int  |
| route_id | ID of the route to attach to.                                       | int  |
| order    | Zero-based index at which checkpoint should be inserted into route. | int  |

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/transmute' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id":  
23144, "route_id": 12334, "order": 0}'
```

##### HTTP GET

```
https://api.navixy.com/v2/task/transmute?  
hash=a6aa75587e5c59c32d347da438505fc3&task_id=23144&route_id=12334&ord
```

#### response

```
{ "success": true }
```

#### errors

- 201 – Not found in the database (if there is no task or route with such an ID, or tracker to which checkpoint assigned is unavailable to current sub-user).
- 255 – Invalid task state (if task or any of the checkpoints are not in unassigned or assigned state).

#### update

Updates existing task. Note that you cannot change task owner using this method.

**required sub-user rights:** `task_update`.

## parameters

| name        | description   | type        |
|-------------|---|-------------|
| task        | <code>task</code> object without fields which are <i>IGNORED</i> .  | JSON object |
| create_form | If <code>true</code> then check additional <code>form_template_id</code> field in <code>task</code> object and create, replace or delete task's form. Default value is <code>false</code> for backward compatibility. | boolean     |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/update' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task":  
{"id": 22379, "location": {"lat": 34.178868, "lng": -118.599672,  
"radius": 150}, "label": "Name", "description": "Description  
example", "from": "2020-02-03 04:05:06", "to": "2020-03-04  
05:06:07"}, "create_form": false}'
```

A returned object also can include "external\_id\_counts" field see [task/route/create method description](#).

## response

```
{  
  "success": true,  
  "external_id_counts": [{  
    "external_id": "456",  
    "count": 2  
  }]  
}
```

## errors

- 201 – Not found in the database (if there is no task with such an ID).
- 255 – Invalid task state (if current task state is not "unassigned" or "assigned").

Last update: April 8, 2024







# Routes

Routes basically named and ordered set of checkpoints. Each [checkpoint](#) is essentially a task with an additional link to the parent route.

Route completed if all the checkpoints completed and visited in the specified order. Otherwise, it is considered completed with warnings or failed.

## Route object

```
{
  "id": 111,
  "user_id": 3,
  "tracker_id": 222653,
  "label": "Deliver parcels",
  "description": "Quickly",
  "creation_date": "2014-01-02 03:04:05",
  "from": "2014-02-03 04:05:06",
  "to": "2014-03-04 05:06:07",
  "external_id": null,
  "status": "assigned",
  "status_change_date": "2014-01-02 03:04:05",
  "origin": "imported",
  "tags": [1, 2],
  "checkpoint_ids": [2977, 2978],
  "type": "route"
}
```

- `id` - int. Primary key used in route/update, *IGNORED* in route/create.
- `user_id` - int. User ID. *IGNORED* in route/create and route/update.
- `tracker_id` - int. An ID of the tracker to which route assigned. Can be null. *IGNORED* in route/update.
- `creation_date` - [date/time](#). When route created. *IGNORED* in route/create, route/update.
- `from` - [date/time](#). Date AFTER which first checkpoint zone must be visited, depends on first checkpoint `from`, *IGNORED* in route/create, route/update.
- `to` - [date/time](#). Date BEFORE which last checkpoint zone must be visited, depends on last checkpoint `to`, *IGNORED* in route/create, route/update.
- `external_id` - string. Used if route imported from external system. arbitrary text string. Can be null.
- `status` - string. A route status. *IGNORED* in route/create, route/update.

- `status_change_date` - [date/time](#). When route status changed. *IGNORED* in route/create, route/update.
- `origin` - string. A route origin. *IGNORED* in route/create, route/update.
- `tags` - int array. List of tag IDs.
- `checkpoint_ids` - int array. List of route checkpoint IDs in order of execution. *IGNORED* in route/create.

## API actions

API base path: `/task/route`.

### assign

(Re)assigns route to a new tracker (or make it unassigned).

**required sub-user rights:** `task_update`.

#### parameters

| name                    | description   | type |
|-------------------------|---|------|
| <code>route_id</code>   | ID of the route to assign.  | int  |
| <code>tracker_id</code> | ID of the tracker. Tracker must belong to authorized user and not be blocked. If null, task will be assigned to none. | int  |

#### examples

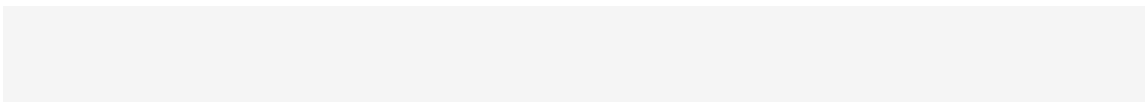
##### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/route/assign' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "route_id": 11231, "tracker_id": 223465}'
```

##### HTTP GET

```
https://api.navixy.com/v2/task/route/assign?
hash=a6aa75587e5c59c32d347da438505fc3&route_id=11231&tracker_id=223465
```

#### response



```
{
  "success": true
}
```

## errors

- 201 – Not found in the database - if there is no task with such an ID.
- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 255 – Invalid task state - if current task state is not "unassigned" or "assigned".
- 236 – Feature unavailable due to tariff restrictions - if device's tariff does not allow usage of tasks.

## create

Creates a new route. One of checkpoints can have ID (in this case it must be a task) - it will be transmuted from task to checkpoint.

**required sub-user rights:** `task_update`.

## parameters

| name        | description   | type                  |
|-------------|---|-----------------------|
| route       | Route object without fields which are <i>IGNORED</i> .  | JSON object           |
| checkpoints | Array of <a href="#">checkpoint objects</a> without fields which are <i>IGNORED</i> .   | array of JSON objects |
| create_form | If <code>true</code> then check additional <code>form_template_id</code> field in every <b>checkpoint</b> object and create form if it is not null. Default value is <code>false</code> for backward compatibility. | boolean               |

Minimal route object to create a new route must contain:

```
{
  "tracker_id": 223652,
  "label": "Name",
  "description": "Description example"
}
```

Also, need checkpoints list in order of execution, checkpoints `from` and `to` must be agreed with each other i.e. checkpoint `to` cannot be before 'from' of preceding items.

```
{
  "tracker_id": 223652,
  "location": {
    "lat": 34.178868,
    "lng": -118.599672,
    "radius": 150
  },
  "label": "Name",
  "description": "Description example",
  "from": "2014-02-03 04:05:06",
  "to": "2014-03-04 05:06:07"
}
```

- `tracker_id` - int. Optional. If the field specified then the task will be assigned to the employee associated with the tracker, otherwise it won't be assigned to anybody.
- `location` - area (circle geofence), entering and leaving of geofence will be controlled.
  - `lat` - float. Latitude.
  - `lng` - float. Longitude.
  - `radius` - int. Radius in meters.
- `label` - string. Task name, length 1-200 characters.
- `description` - string. Task description, length 0-1024 characters.
- `from` - [date/time](#). Start date of the interval - when the specified location has to be visited (in the user's time zone).
- `to` - [date/time](#). End date of the interval - when the specified location has to be visited (in the user's time zone).

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/route/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "route":
{"tracker_id": 669673, "label": "Products delivery",
"description": "12 trackers of model 1 and 37 trackers of model
2", "from": "2020-03-18 10:00:00", "to": "2020-03-18 16:00:00"},
"checkpoints": [{ "tracker_id": 669673, "location": { "lat":
34.178868, "lng": -118.599672, "radius": 100}, "label":
"Company1", "description": "5 trackers of model 1 and 15 trackers
of model 2", "from": "2021-03-18 10:00:00", "to": "2021-03-18
12:00:00", "external_id": "10100", "max_delay": 0,
"min_stay_duration": 10, "tags": [1, 4], "form_template_id":
132985}, { "tracker_id": 669673, "location": { "lat": 31.738386,
"lng": -106.453854, "radius": 100}, "label": "Company2",
"description": "4 trackers of model 1 and 12 trackers of model 2",
"from": "2021-03-18 10:00:00", "to": "2021-03-18 14:00:00",
"external_id": "10101", "max_delay": 0, "min_stay_duration": 10,
"tags": [2, 4], "form_template_id": 132985}], "create_form":
false}'
```

## response

Call returns JSON object of the created route. In response there will be external IDs which have count greater than zero. There can be multiple external IDs in response because you can specify different external IDs in a task's checkpoint. If there is nothing to return, then parameter "external\_id\_counts" will not be present in response.

```
{
  "success": true,
  "result": {
    "id": 111,
    "user_id": 3,
    "tracker_id": 22,
    "label": "Deliver parcels",
    "description": "Quickly",
    "creation_date": "2014-01-02 03:04:05",
    "from": "2014-02-03 04:05:06",
    "to": "2014-03-04 05:06:07",
    "external_id": null,
    "status": "assigned",
    "status_change_date": "2014-01-02 03:04:05",
    "origin": "manual",
    "checkpoint_ids": [2977, 2978],
    "type": "route"
  },
  "external_id_counts": [{"external_id": "456", "count": 2}]
}
```

- `checkpoint_ids` - int array. A list of route checkpoint IDs in order of execution.

- `external_id_counts` - optional object. Count of external IDs.

## errors

- 201 – Not found in the database - if `task.tracker_id` is not null and belongs to nonexistent tracker.
- 236 – Feature unavailable due to tariff restrictions - if device's tariff does not allow usage of tasks.

## delete

Deletes route (and its checkpoints) with the specified ID.

**required sub-user rights:** `task_update`.

## parameters

| name                  | description                | type |
|-----------------------|----------------------------|------|
| <code>route_id</code> | ID of the route to delete. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/route/delete' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "route_id": 23144}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/route/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&route_id=23144
```

## response

```
{  
  "success": true  
}
```

## errors

- 201 – Not found in the database - if there is no route with such an ID.



## list

Get all routes belonging to user with optional filtering.

### parameters

| name     | description  | type       |
|----------|--|------------|
| statuses | Optional. List of task statuses, e.g. <code>["unassigned", "failed"]</code> . Default all.   | enum array |
| trackers | Optional. List of <code>tracker_id</code> to which task assigned.  | int array  |
| from     | Optional. Show tasks which are actual AFTER this date, e.g. <code>"2020-06-01 00:00:00"</code> .   | date/time  |
| to       | Optional. Show tasks which are actual BEFORE this date, e.g. <code>"2020-07-01 00:00:00"</code> .  | date/time  |
| filter   | Optional. Filter for task label and description. If <b>trackers</b> , <b>filter</b> , <b>from</b> or <b>to</b> is not passed or <i>null</i> then appropriate condition not used to filter results. | string     |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/route/list' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/task/route/list?  
hash=a6aa75587e5c59c32d347da438505fc3
```

### response

```
{  
  "success": true,  
  "list": [{  
    "id": 111,  
    "user_id": 3,  
    "tracker_id": 222653,  
    "label": "Deliver parcels",  
    "description": "Quickly",  
    "creation_date": "2014-01-02 03:04:05",  
  }]
```

```

        "from": "2014-02-03 04:05:06",
        "to": "2014-03-04 05:06:07",
        "external_id": null,
        "status": "assigned",
        "status_change_date": "2014-01-02 03:04:05",
        "origin": "imported",
        "tags": [1, 2],
        "checkpoint_ids": [2977, 2978],
        "type": "route"
    }
}

```

## errors

General types only.

## read

Gets route by specified ID.

### parameters

| name     | description      | type |
|----------|------------------|------|
| route_id | ID of the route. | int  |

### response

```

{
    "success": true,
    "value": {
        "id": 111,
        "user_id": 3,
        "tracker_id": 222653,
        "label": "Deliver parcels",
        "description": "Quickly",
        "creation_date": "2014-01-02 03:04:05",
        "from": "2014-02-03 04:05:06",
        "to": "2014-03-04 05:06:07",
        "external_id": null,
        "status": "assigned",
        "status_change_date": "2014-01-02 03:04:05",
        "origin": "imported",
        "tags": [1, 2],
        "checkpoint_ids": [2977, 2978],
        "type": "route"
    }
}

```

```
}  
}
```

- `value` - route object described [here](#).

## errors

- 201 – Not found in the database - if there is no route with such an ID.

## update

Updates existing route. Note that you cannot change task owner using this method. Reordering checkpoint IDs in the `checkpoint_ids` array changes order of execution.

**required sub-user rights:** `task_update`.

## parameters

| name        | description  | type             |
|-------------|--|------------------|
| route       | Route object without fields which are <i>IGNORED</i> .   | JSON object      |
| checkpoints | List of <a href="#">checkpoint objects</a> objects. Should be null if <b>route</b> 's field <b>checkpoint_ids</b> is null, otherwise should be not null. If entry contains ID, then update existing checkpoint, else create a new one. Present route's checkpoints, which are not included in this array, will be deleted. | array of objects |
| create_form | If <code>true</code> then check additional <code>form_template_id</code> field in every <b>checkpoint</b> object and create, replace or delete checkpoint's form. Default value is <code>false</code> for backward compatibility.  | boolean          |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/route/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "route":
{"id": 23785, "label": "Products delivery", "description": "12
trackers of model 1 and 37 trackers of model 2", "from":
"2020-03-18 10:00:00", "to": "2020-03-18 16:00:00"},
"checkpoints": [{ "id": 123, "tracker_id": 669673, "location":
{"lat": 34.178868, "lng": -118.599672, "radius": 100}, "label":
"Company1", "description": "5 trackers of model 1 and 15 trackers
of model 2", "from": "2021-03-18 10:00:00", "to": "2021-03-18
12:00:00", "external_id": "10100", "max_delay": 0,
"min_stay_duration": 10, "tags": [1, 4], "form_template_id":
132985}, {"id": 124, "tracker_id": 669673, "location": {"lat":
31.738386, "lng": -106.453854, "radius": 100}, "label":
"Company2", "description": "4 trackers of model 1 and 12 trackers
of model 2", "from": "2021-03-18 10:00:00", "to": "2021-03-18
14:00:00", "external_id": "10101", "max_delay": 0,
"min_stay_duration": 10, "tags": [2, 4], "form_template_id":
132985}], "create_form": false}'
```

## response

JSON object of the updated route with `checkpoint_id`s

```
{
  "success": true,
  "result": {
    "id": 111,
    "user_id": 3,
    "tracker_id": 22,
    "label": "Deliver parcels",
    "description": "Quickly",
    "creation_date": "2014-01-02 03:04:05",
    "from": "2014-02-03 04:05:06",
    "to": "2014-03-04 05:06:07",
    "external_id": null,
    "status": "assigned",
    "status_change_date": "2014-01-02 03:04:05",
    "origin": "manual",
    "checkpoint_ids": [2977, 2978],
    "type": "route"
  }
}
```

## errors

- 201 – Not found in the database - if there is no task with such an ID.
- 255 – Invalid task state - if current task state is not "unassigned" or "assigned".

Last update: January 15, 2024





# Optimizing routes

To reduce transit time and costs, it may be helpful to rearrange route checkpoints so that the total travel time between them is minimized. Our platform offers a way to perform this optimization. You don't even need to create a route and checkpoints; you simply provide the necessary data for optimization, and the algorithm returns the order in which the points should be visited.

## API actions

API path: `/task/route/points/optimize`.

### optimize

The suggested order for the given route points will correspond to the time windows (from and to) of each point. Points with earlier time windows will have lower ordinal numbers. If time windows overlap, the order of such points may vary to maximize the overall efficiency of the route. The maximum distance per route optimization is 5000 kilometers. When using APIs, the maximum number of points per route optimization is 49 points to visit, plus 1 start point.

**required sub-user rights:** `task_update`.

### parameters

- **start\_point** - (object) the coordinates of the location from where the performer will depart. The departure time is optional parameter.

```
{
  "lat": 15.233,
  "lng": -5.554,
  "departure": "2024-03-19 13:30:00"
}
```

- **route\_points** - (array of objects) the points that the performer must visit, and the count of points must be within the range of 2 to 49. For example:

```
[
  {
    "location": {
      "lat": 11.111,
      "lng": 11.111
    },
    "from": "2024-03-19 00:00:00",
    "to": "2024-03-19 23:59:00"
  },
  {
    "location": {
      "lat": 22.222,
      "lng": -2.222
    },
    "from": "2024-03-19 00:00:00",
    "to": "2024-03-19 23:59:00"
  },
  {
    "location": {
      "lat": -3.333,
      "lng": 33.333
    },
    "from": "2024-03-19"
  }
]
```



```
00:00:00", "to": "2024-03-19 23:59:00"},
  {"location": {"lat": -4.444, "lng": -4.444}, "from": "2024-03-19
00:00:00", "to": "2024-03-19 23:59:00"},
  {"location": {"lat": 55.555, "lng": 55.555}, "from": "2024-03-19
00:00:00", "to": "2024-03-19 23:59:00"}
]
```

## response

```
{
  "success": true,
  "result": [2, 0, 1]
}
```

The `result` will return the order in which the points should be visited.

If for route points:

```
[
  {route_point_0}, // index in list = 0
  {route_point_1}, // index in list = 1
  {route_point_2} // index in list = 2
]
```

this action returns: `[2, 0, 1]`

it means "change points order as following":

```
point at index 2 move to index 0,
point at index 0 move to index 1,
point at index 1 move to index 0
```

or with a more tangible example with 5 points. You have the next points to be reordered

```
[
{"location": {"lat": 38.81673961922754, "lng": -77.15569496154785},
"from": "2024-03-19 00:00:00", "to": "2024-03-19 23:59:00"}, // it
has index 0
{"location": {"lat": 38.82767290746902, "lng": -77.1445369720459},
"from": "2024-03-19 00:00:00", "to": "2024-03-19 23:59:00"}, // it
has index 1
{"location": {"lat": 38.834760258479704, "lng":
-77.14093208312988}, "from": "2024-03-19 00:00:00", "to":
"2024-03-19 23:59:00"}, // this one with index 2
{"location": {"lat": 38.81583679562883, "lng": -77.14814186096191},
"from": "2024-03-19 00:00:00", "to": "2024-03-19 23:59:00"}, //
this with index 3
{"location": {"lat": 38.81031929163279, "lng": 7.15582370758057},
"from": "2024-03-19 00:00:00", "to": "2024-03-19 23:59:00"} // and
this one has index 4
]
```

The API request will be

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/route/points/optimize' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"start_point": {"lat": 38.81476676765485, "lng":
-77.1608018875122}, "route_points": [{"location": {"lat":
38.81673961922754,"lng": -77.15569496154785}, "from": "2024-03-19
00:00:00", "to": "2024-03-19 23:59:00"}, {"location": {"lat":
38.82767290746902,"lng": -77.1445369720459}, "from": "2024-03-19
00:00:00", "to": "2024-03-19 23:59:00"}, {"location": {"lat":
38.834760258479704,"lng": -77.14093208312988}, "from": "2024-03-19
00:00:00", "to": "2024-03-19 23:59:00"}, {"location": {"lat":
38.81583679562883,"lng": -77.14814186096191}, "from": "2024-03-19
00:00:00", "to": "2024-03-19 23:59:00"}, {"location": {"lat":
38.81031929163279,"lng": 7.15582370758057}, "from": "2024-03-19
00:00:00", "to": "2024-03-19 23:59:00"}]}'
```

The platform will reply to you with:

```
{
  "result": [4,0,3,1,2],
  "success": true
}
```

So the optimized route with start point from "lat": 38.81476676765485, "lng": -77.1608018875122 should be:

```
[
{"location": {"lat": 38.81031929163279,"lng": 7.15582370758057},
"from": "2024-03-19 00:00:00", "to": "2024-03-19 23:59:00"}, //
this one had index 4, now it is the first point to visit
{"location": {"lat": 38.81673961922754,"lng": -77.15569496154785},
"from": "2024-03-19 00:00:00", "to": "2024-03-19 23:59:00"}, // it
had index 0, now it is the second point to visit
{"location": {"lat": 38.81583679562883,"lng": -77.14814186096191},
"from": "2024-03-19 00:00:00", "to": "2024-03-19 23:59:00"}, //
this with index 3 becomes the third point to visit
{"location": {"lat": 38.82767290746902,"lng": -77.1445369720459},
"from": "2024-03-19 00:00:00", "to": "2024-03-19 23:59:00"}, // it
had index 1, now it is the fourth point to visit
{"location": {"lat": 38.834760258479704,"lng":
-77.14093208312988}, "from": "2024-03-19 00:00:00", "to":
"2024-03-19 23:59:00"} // and this one with index 2, now it is the
last fifth point to visit
]
```

#### errors

- 7 - Invalid parameters.

- 210 - Path distance exceeds the max distance limit - if the overall route distance is more than 5000 km.
- 264 - Timeout not reached - too high api call rate.

Last update: March 19, 2024





# Checkpoints


Every route consists of checkpoints. Using these actions, you can manipulate checkpoints individually.

## Checkpoint object

```
{
  "id": 111,
  "user_id": 3,
  "tracker_id": 22,
  "location": {
    "lat": 51.283546,
    "lng": 7.301086,
    "address": "Fichtenstrasse 11",
    "radius": 150
  },
  "label": "Deliver parcels",
  "description": "Quickly",
  "creation_date": "2014-01-02 03:04:05",
  "from": "2014-02-03 04:05:06",
  "to": "2014-03-04 05:06:07",
  "external_id": null,
  "status": "assigned",
  "status_change_date": "2014-01-02 03:04:05",
  "max_delay": 5,
  "min_stay_duration": 0,
  "arrival_date": "2014-01-02 03:04:05",
  "stay_duration": 0,
  "origin": "imported",
  "tags": [1, 2],
  "type": "checkpoint",
  "form": <form_object>,
  "form_template_id": 13245
}
```

- `id` - int. Primary key. Used in checkpoint/update. *IGNORED* in checkpoint/create.
- `user_id` - int. User ID. *IGNORED* in checkpoint/create, checkpoint/update.
- `tracker_id` - int. An ID of the tracker to which task assigned. Can be null. *IGNORED* in checkpoint/update.
- `location` - location associated with this checkpoint. cannot be null.
  - `address` - string. Address of the location.
  - `radius` - int. Radius of location zone in meters.

- `creation_date` - [date/time](#). When checkpoint created. *IGNORED* in checkpoint/create, checkpoint/update.
- `from` - [date/time](#). Date AFTER which checkpoint zone must be visited.
- `to` - [date/time](#). Date BEFORE which checkpoint zone must be visited.
- `external_id` - int. Used if task imported from external system. Arbitrary text string. Can be null.
- `status` - [enum](#). Checkpoint status. *IGNORED* in checkpoint/create, checkpoint/update.
- `status_change_date` - [date/time](#). When checkpoint status changed. *IGNORED* in checkpoint/create and checkpoint/update.
- `max_delay` - int. Maximum allowed checkpoint completion delay in minutes.
- `min_stay_duration` - int. Minimum duration of stay in checkpoint zone for checkpoint completion, minutes.
- `arrival_date` - [date/time](#). When tracker has arrived to the checkpoint zone. *IGNORED* in checkpoint/create, checkpoint/update.
- `stay_duration` - int. Duration of stay in the checkpoint zone, seconds.
- `origin` - string. Checkpoint origin. *IGNORED* in checkpoint/create, checkpoint/update.
- `tags` - int array. List of tag IDs.
- `form` - [form object](#). If present.
- `form_template_id` - int. An ID of form template. Used in create and update actions only if `create_form` parameter is `true` in them.

 **To associate the task with an address - this field should be added to the location object.**

## API actions

API base path: `/task/checkpoint`.

### create

Creates a new checkpoint.

**required sub-user rights:** `task_update`.

## parameters

| name       | description  | type        |
|------------|--|-------------|
| checkpoint | A <code>checkpoint</code> object without fields which are <i>IGNORED</i> . | JSON object |

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/checkpoint/create' \
-H 'Content-Type: application/json' \
-d '{"hash": "43c37b8d023272c5fd1b27d21244b", "checkpoint":
{"tracker_id": 10179999, "label": "Shop 1", "description":
"Description", "parent_id": 352665, "order": 0, "location":
{ "lat": 9.861999, "lng": -83.948999, "radius": 150},
"max_delay" : 5, "min_stay_duration": 0, "min_arrival_duration":
0, "from": "2022-12-14 11:00:00", "to": "2022-12-14 11:30:00",
"duration": 60, "tags": [1, 2], "form_template_id": 1}}'
```

## response

Inserts the specified checkpoint at the specified position ( `order` ) in the parent route checkpoints list. Shifts the checkpoint currently at that position (if any) and any subsequent checkpoints to the right (adds one to their orders).

Call returns the identifier of the created task in the form of JSON. The returned object also can include "external\_id\_counts" field see `task/route/create` [method description](#).

```
{
  "success": true,
  "id": 222,
  "external_id_counts": [{
    "external_id": "456",
    "count": 2
  }]
}
```

## errors

- 201 – Not found in the database - if task.tracker\_id is not null and belongs to nonexistent tracker.
- 236 – Feature unavailable due to tariff restrictions - if device's tariff does not allow usage of tasks.



## delete

Deletes a checkpoint with the specified ID.

**required sub-user rights:** `task_update`.

### parameters

| name          | description                     | type |
|---------------|---------------------------------|------|
| checkpoint_id | ID of the checkpoint to delete. | int  |

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/checkpoint/delete' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",  
    "checkpoint_id": 23144}'
```

#### HTTP GET

```
https://api.navixy.com/v2/task/checkpoint/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&checkpoint_id=23144
```

### response

```
{ "success": true }
```

### errors

- 201 – Not found in the database - if there is no checkpoint with such an ID.

## list

Get checkpoints belonging to user with given IDs

### parameters

| name           | description                                   | type      |
|----------------|---|-----------|
| checkpoint_ids | IDs of checkpoints, e.g. <code>[1,2]</code> . | int array |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/checkpoint/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
    "checkpoint_ids": [1,2]}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/checkpoint/list?
hash=a6aa75587e5c59c32d347da438505fc3&checkpoint_ids=[1,2]
```

## response

```
{
  "success": true,
  "list": [{
    "id": 111,
    "user_id": 3,
    "tracker_id": 22,
    "location": {
      "lat": 51.283546,
      "lng": 7.301086,
      "address": "Fichtenstrasse 11",
      "radius": 150
    },
    "label": "Deliver parcels",
    "description": "Quickly",
    "creation_date": "2014-01-02 03:04:05",
    "from": "2014-02-03 04:05:06",
    "to": "2014-03-04 05:06:07",
    "external_id": null,
    "status": "assigned",
    "status_change_date": "2014-01-02 03:04:05",
    "max_delay": 5,
    "min_stay_duration": 0,
    "arrival_date": "2014-01-02 03:04:05",
    "stay_duration": 0,
    "origin": "imported",
    "tags": [1, 2],
    "type": "checkpoint"
  }]
}
```

## errors

General types only.

## read

Gets route checkpoint by specified ID.

## parameters

| name          | description           | type |
|---------------|-----------------------|------|
| checkpoint_id | ID of the checkpoint. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/checkpoint/read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",  
    "checkpoint_id": 111}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/checkpoint/read?  
hash=a6aa75587e5c59c32d347da438505fc3&checkpoint_id=111
```

## response

```
{  
  "success": true,  
  "value": {  
    "id": 111,  
    "user_id": 3,  
    "tracker_id": 22,  
    "location": {  
      "lat": 51.283546,  
      "lng": 7.301086,  
      "address": "Fichtenstrasse 11",  
      "radius": 150  
    },  
    "label": "Deliver parcels",  
    "description": "Quickly",  
    "creation_date": "2014-01-02 03:04:05",  
    "from": "2014-02-03 04:05:06",  
    "to": "2014-03-04 05:06:07",  
    "external_id": null,  
    "status": "assigned",  
    "status_change_date": "2014-01-02 03:04:05",  
    "max_delay": 5,  
    "min_stay_duration": 0,  
    "arrival_date": "2014-01-02 03:04:05",  
    "stay_duration": 0,  
    "origin": "imported",  
    "tags": [1, 2],  
    "type": "checkpoint",  
    "form": <form_object>
```

```
}  
}
```

- `value` - checkpoint object described [here](#).

## errors

- 201 – Not found in the database - if there is no checkpoint with such an ID.

## transmute

Convert route checkpoint into a standalone task. If it's the only checkpoint in the route, the route deleted.

**required sub-user rights:** `task_update`.

## parameters

| name          | description           | type |
|---------------|-----------------------|------|
| checkpoint_id | ID of the checkpoint. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/checkpoint/transmute'  
\  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",  
    "checkpoint_id": 111}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/checkpoint/transmute?  
hash=a6aa75587e5c59c32d347da438505fc3&checkpoint_id=111
```

## response

```
{  
  "success": true  
}
```

## errors

- 201 – Not found in the database - if there is no checkpoint with such an ID, or tracker to which checkpoint assigned is unavailable to current sub-user.

- 255 – Invalid task state - if any of checkpoints are not in unassigned or assigned state.

## update

Updates existing checkpoint.

**required sub-user rights:** `task_update`.

### parameters

| name       | description  | type        |
|------------|--|-------------|
| checkpoint | A <code>checkpoint</code> object without fields which are <i>IGNORED</i> . | JSON object |

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/checkpoint/update' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "checkpoint":   
{ "id": 111, "tracker_id": 10179999, "label": "Shop 1",   
  "description": "Description", "parent_id": 352665, "order": 0,   
  "location": { "lat": 9.861999, "lng": -83.948999, "radius": 150},   
  "max_delay" : 5, "min_stay_duration": 0, "min_arrival_duration":   
  0, "from": "2022-12-14 11:00:00", "to": "2022-12-14 11:30:00",   
  "duration": 60, "tags": [1, 2], "form_template_id": 1}}'
```

Changing `order` reorders all other checkpoints.

### response

```
{  
  "success": true,  
  "external_id_counts": [{  
    "external_id": "456",  
    "count": 2  
  }]  
}
```

- `external_id_counts` - array of objects. Optional. A returned object also can include "external\_id\_counts" field see task/route/create [method description](#).

**errors**

- 201 – Not found in the database - if there is no task with such an ID.
- 255 – Invalid task state - if current task state is not "unassigned" or "assigned".

Last update: January 15, 2024







# Working with task forms

Forms can be attached to tasks to be filled by field employees using Mobile Tracker App ([Android](#) / [iOS](#)). This document describes API actions specific to working with task forms (except `task/form/list` which can return all kinds of forms).

For `<form_field>` and `<form_value>` object description, see [form fields and values](#).

For `<form>` object description, see [forms](#).

Contains API calls related to forms associated with tasks.

## API actions

API path: `/task/form`.

### create

Attaches new form to the existing task or checkpoint. Form always created on the basis of form template.

**required sub-user rights:** `task_update`.

### parameters

| name        | description                  | type |
|-------------|------------------------------|------|
| task_id     | An ID of the task to assign. | int  |
| template_id | An ID of the form template.  | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/form/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id": 11231, "template_id": 12548}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/form/create?
hash=a6aa75587e5c59c32d347da438505fc3&task_id=11231&template_id=12548
```

## response

```
{
  "success": true
}
```

## errors

- 201 – Not found in the database - if there is no task or template with such an ID, or task has the "route" type.
- 247 – Entity already exists - if task already has form attached to it.
- 255 – Invalid task state - if current task state is not `unassigned`, `assigned` or `arrived`.

## delete

Deletes a form (detach it from the task).  
All form data will be lost!

**required sub-user rights:** `task_update`.

## parameters

| name    | description                                  | type |
|---------|--|------|
| task_id | An ID of the task to which form is attached. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/form/delete' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id":  
11231}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/form/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&task_id=11231
```

## response

```
{  
  "success": true  
}
```

## errors

- 201 – Not found in the database - if there is no task with such an ID, or task has the "route" type, or it has no form attached.
- 255 – Invalid task state - if current task state is not `unassigned`, `assigned` or `arrived`.

## download

Retrieves attached form as file.

## parameters

| name    | description  | type |
|---------|--|------|
| task_id | An ID of the task.   | int  |
| format  | Format of the download file. Can be "xls", "csv" or "pdf". | enum |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/form/download' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id":  
11231, "format": "pdf"}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/form/download?  
hash=a6aa75587e5c59c32d347da438505fc3&task_id=11231&format=pdf
```

## response

A form rendered to file (standard file download).

## errors

- 201 – Not found in the database - if task does not exist or does not have attached form.

## list

Returns descriptions of forms, created on the basis of specified form template. In addition to the data on the forms, the list contains data on the objects related to each form – tracker / vehicle / employee, task.

## parameters

- **template\_id** (*integer, optional*). The returned list will contain forms, related to that template.  
**warning:** at least one of **template\_id** and **task\_ids** parameters must be not null.
- **task\_ids** (*list of integers, optional*). Maximum size of list is 5000 elements. List of task IDs. The returned list will contain forms, related to tasks, which IDs specified in this parameter.  
**warning:** at least one of **template\_id** and **task\_ids** parameters must not be null.
- **order\_by** (*optional, default = submitted*). Data field for list sorting. Available values:
  - *task\_id*
  - *created*
  - *submitted*
  - *task\_address*
  - *submit\_address*

- *employee\_full\_name*
- *vehicle\_label*
- *tracker\_label*
- *task\_label*
- *task\_creation\_date*
- *task\_from*
- *task\_to*
- *task\_arrival\_date*
- *task\_completion\_date*
- *form\_label*
- *form\_description*
- **ascending** (*boolean, required*). Sorting direction (ascending / descending).
- **include\_unsubmitted** (*boolean, required*). If true, unsubmitted forms shall be included in the list.
- **filters** (*object, optional*). Specifies the criteria for filtering the list based on the values of the data fields. Conditions are combined by logical AND.\Filters object contains following optional elements:

```
{
  "employee_full_name": "John Dow", // a sequence of characters
  for partial matching (against the name of the associated employee)
  "form_description": "Description",
  "form_label": "Form Label",
  "submit_address": "Submit Address",
  "task_id": 123, // strict match
  "task_address": "Task Address",
  "task_label": "Task Label",
  "tracker_label": "Tracker label",
  "vehicle_label": "Vehicle label",
}
```

- **submit\_period** (*period\_object, optional*).
- **task\_creation\_period** (*period\_object, optional*).
- **task\_from\_period** (*period\_object, optional*).
- **task\_to\_period** (*period\_object, optional*).
- **task\_arrival\_period** (*period\_object, optional*).
- **task\_completion\_period** (*period\_object, optional*).

where *period\_object* is:

```
{
  "from": "2020-02-03 03:00:00", // string <date/time>
  "to": "2020-02-03 08:00:00" // string <date/time>
}
```

- **offset, limit** (*integers, optional*). Specify which subset of elements from all matching results will be included in the returned list.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/form/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "ascending": true, "include_unsubmitted": true}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/form/list?
hash=a6aa75587e5c59c32d347da438505fc3&ascending=true&include_unsubmitt
```

## response

```
{
  "success": true,
  "count": 2,
  "list": [{
    "employee": {
      "id": 13,
      "first_name": "John",
      "middle_name": "",
      "last_name": "Dow"
    },
    "task": {
      "id": 7867,
      "label": "My task 3",
      "from": "2017-07-27 15:00:00",
      "to": "2017-07-28 14:59:59",
      "creation_date": "2017-07-27 12:12:23",
      "arrival_date": "2017-07-27 15:14:07",
      "address": "Moltkestrasse 32",
      "status": "done",
      "completion_date": "2017-07-28 14:36:28",
      "fact_duration": "PT22M21S"
    },
    "tracker": {
      "id": 15620,
      "label": "Navixy A6"
    },
    "vehicle": null,
    "form": {
      "id": 1012,
      "label": "A form",
      "description": ""
    }
  ]
}
```

```

    "fields": [],
    "created": "2017-07-28 03:48:06",
    "submit_in_zone": false,
    "task_id": 7867,
    "template_id": 449,
    "values": null,
    "submitted": "2017-03-21 18:40:54",
    "submit_location": {
      "lat": 26.826762,
      "lng": 20.5947311,
      "address": "Partizan st., 4"
    },
    "submit_places": {
      "location": {
        "lat": 26.826762,
        "lng": 20.5947311,
        "address": "Partizan st., 4"
      },
      "places": [{
        "id": 38,
        "label": "Office sweet office"
      }],
      "zones": [{
        "id": 18404,
        "label": "Zone 51"
      }]
    }
  }
}]

```

- `count` - int. Total number of forms matching the query.
- `form` - [form object](#), non-null.
- `submit_places` - additional info about places/zones related to form submission, can be null.
  - `places` - list of places associated with zone submission location. Can be empty.
  - `zones` - list of zones associated with zone submission location. Can be empty.

## errors

- 204 – Not found - if there is no form template with such ID belonging to authorized user.
- [General](#) types of errors.

## read

Gets form associated with the specified task.

## parameters

| name    | description        | type |
|---------|--------------------|------|
| task_id | An ID of the task. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/form/read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id":  
12546}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/form/read?  
hash=a6aa75587e5c59c32d347da438505fc3&task_id=12546
```

## response

```
{  
  "success": true,  
  "value" : <form>,  
  "files" : [  
    {  
      "id": 16,  
      "storage_id": 1,  
      "user_id": 12203,  
      "type": "image",  
      "created": "2020-09-06 11:54:28",  
      "uploaded": "2020-09-06 11:55:14",  
      "name": "lala.jpg",  
      "size": 72594,  
      "mime_type": "image/png",  
      "metadata": {  
        "orientation": 1  
      },  
      "state": "uploaded",  
      "download_url": "https://static.navixy.com/file/dl/  
1/0/1g/01gw2j5q7nm4r92dytolzd6koxy9e38v.png/lala.jpg",  
      "bindings": {  
        "form_field": {  
          "form_id": 40,  
          "field_id": "2222",  
          "submitted": false  
        }  
      },  
      "previews": [  
        {  
          "download_url": "https://localhost:8084/file/  
preview/1/0/1g/01gw2j5q7nm4r92dytolzd6koxy9e38v.png/lala.jpg"
```



```
}
  ]
}
]
```

- `value` - [form object](#), or null if no form attached.
- `files` - list of files, both submitted and unsubmitted, associated with this form's fields.
  - `id` - int. File ID.
  - `type` - [enum](#). Can be "image" or "file".
  - `created` - [date/time](#). Date when file created.
  - `uploaded` - [date/time](#). Date when file uploaded, can be null if file not yet uploaded.
  - `name` - string. Filename.
  - `size` - int. Size in bytes. If file not uploaded, show maximum allowed size for the upload.
  - `metadata` - metadata object.
  - `orientation` - int. Image exif orientation.
  - `state` - [enum](#). Can be "created", "in\_progress", "uploaded", "deleted".
  - `download_url` - string. Actual URL at which file is available. Can be null if file not yet uploaded.
  - `bindings` - all entities to which this file linked.
  - `previews` - available preview images for the file. Can be null or empty for any file in any state.

## errors

- 201 – Not found in the database - if there is no task with such an ID, or task assigned to tracker unavailable to current sub-user.

Last update: January 22, 2023





# Updating task form values

Task form values can only be submitted using the web API if there was a previous submission using the Mobile Tracker App ([Android](#) / [iOS](#)). The purpose of this feature is to correct any incorrectly filled data that was accidentally submitted. It is not intended for filling out an empty form from scratch. This action can only be used if the task status is "assigned," and the device must not be deleted. It will not work for any other task statuses.

## API actions

API path: `/task/form/values`.

### update

Updates existing form values of given task.

**required sub-user rights:** `task_update`.

#### PARAMETERS

| name    | description                   | type        |
|---------|-------------------------------|-------------|
| task_id | An ID of the task.            | int         |
| values  | Map of field_id-value object. | JSON object |

where values object is:

```
{
  "text1": {
    "type": "text",
    "value": "text field value"
  }
}
```

For **value** object description, see [form/form-fields-and-values/](#).

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/form/values/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id": 12546, "value": {"text1": {"type": "text", "value": "text field value"}}}'
```

## response

```
{
  "success": true
}
```

## errors

- 101 – In demo mode this function disabled - if current user has "demo" flag.
- 201 – Not found in the database - if task with the specified ID does not exist.
- 255 – Invalid task state - if task has already done or failed or no values submitted.
- 242 – There were errors during content validation - if given values are invalid for the form. Example:

```
{
  "success": false,
  "status": {
    "code": 242,
    "description": "There were errors during content validation"
  },
  "errors": [
    {
      "field_id": "111-aaa-whatever",
      "code": 5,
      "error": "text length constraints are not met"
    }
  ]
}
```

## Validation error codes:

- 1 – field required but has no value.
- 2 – field value type doesn't match field type.
- 3 – field value is null.
- 4 – value index out of bounds.
- 5 – invalid value size.

- 6 – value less than minimum.
- 7 – value more than maximum.
- 8 – field contains invalid references.
- 9 – invalid file type.
- 10 – invalid file state.

Last update: March 19, 2024







## Attaching files

When submitting form values of type [file](#), [photo](#) or [signature](#), you need to provide file ID. To obtain it, first you [create](#) a file entry, then upload a file using provided credentials. File must adhere to limitations specified in the form field. Note that each file consumes space and contributes to file storage limit associated with user's account.

### API actions

API path: `/task/form/file`.

#### create

Creates a new file entry associated with form's field. By making this call you basically "request permission" to upload a file. In return, you are provided with upload credentials (URL, form fields, etc.).

Note that in order to actually "include" file as form field's value, creating and uploading file is not enough. You must then submit a form with file ID as a value of corresponding form field.

If file created but not uploaded, it will be deleted after date/time specified in "expires" response field. If file uploaded but not included as form field's value, it will be deleted on next form submission.

**required sub-user rights:** `task_update`.

#### parameters

| name     | description   | type   |
|----------|---|--------|
| task_id  | ID of the task to which form attached.  | int    |
| field_id | ID of the form's field to which a new file should be attached.  | string |
| size     | Maximum size in bytes for the file which will be uploaded. This is needed to "reserve" the space for a file in user's disk space quota. | int    |

| name     | description  | type        |
|----------|--|-------------|
| filename | Optional. If specified, uploaded file will have the specified name. If not, name will be taken from actual file upload form. | string      |
| metadata | Optional. Metadata object (for images only).   | JSON object |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/form/file' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id": 11231, "field_id": "file1", "size": 10}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/form/file?
hash=a6aa75587e5c59c32d347da438505fc3&task_id=11231&field_id=file1&size=10
```

## response

when using internal storage:

```
{
  "success": true,
  "value": {
    "file_id": 111,
    "url": "http://bla.org/bla",
    "expires": "2020-02-03 03:04:00",
    "file_field_name": "file",
    "fields": {
      "token": "a43f43ed4340b86c808ac"
    }
  }
}
```

when using the Amazon S3:

```
{
  "success": true,
  "value": {
    "file_id": 111,
    "url": "https://bla.s3.amazonaws.com/",
    "expires": "2020-02-03 03:04:00",
    "file_field_name": "file",
    "fields": {
      "policy": "<Base64-encoded policy string>",
    }
  }
}
```

```

    "key": "user/user1/${filename}",
    "success_action_status": "200",
    "x-amz-algorithm": "AWS4-HMAC-SHA256",
    "x-amz-credential": "AKIAIOSFODNN7EXAMPLE/20151229/us-east-1/
s3/aws4_request",
    "x-amz-date": "20151229T000000Z",
    "x-amz-signature": "<signature-value>",
    "x-amz-server-side-encryption": "AES256",
    "content-type": "image/png"
  }
}
}

```

- `file_id` - int. This value will be submitted as form's field value.
- `url` - string. A URL to which POST form-data with file contents should be executed.
- `expires` - date/time. After this date file record will expire and upload requests will be rejected.
- `file_field_name` - string. Name for file field in POST upload request.
- `fields` - these fields should be passed as additional fields in POST multipart upload request, field with a file must be the last one.

### How to upload file data

Here's an example of upload you must make after receiving such response (assuming you uploading image named `actual_file_name.png`):

Internal storage example:

```

POST /bla HTTP/1.1
Host: bla.org
Content-Length: 1325
Origin: http://bla.org
... other headers ...
Content-Type: multipart/form-data; boundary=WebAppBoundary

--WebAppBoundary
Content-Disposition: form-data; name="token"

a43f43ed4340b86c808ac
--WebAppBoundary
Content-Disposition: form-data; name="file";
filename="actual_file_name.png"
Content-Type: image/png

... contents of file goes here ...
--WebAppBoundary--

```

Amazon S3 example:

```
POST / HTTP/1.1
Host: https://bla.s3.amazonaws.com
Content-Length: 1972
Origin: https://bla.s3.amazonaws.com/
... other headers ...
Content-Type: multipart/form-data; boundary=WebAppBoundary

--WebAppBoundary
Content-Disposition: form-data; name="policy"
Content-Type: text/plain

eyJleHBpcmF0aW9uIjogIjIwMjMtMDMtMjdUMjE6MTU6MzYuMDczWiIsImNvbmlRpdGlbn
--WebAppBoundary
Content-Disposition: form-data; name="key"
Content-Type: text/plain

nj9relv6m52qp01t0wv47wyk1ozd309g/${filename}
--WebAppBoundary
Content-Disposition: form-data; name="success_action_status"
Content-Type: text/plain

200
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-algorithm"
Content-Type: text/plain

AWS4-HMAC-SHA256
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-credential"
Content-Type: text/plain

AKIAIBQ6SRB65EVSSRMA/20230327/eu-central-1/s3/aws4_request
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-date"
Content-Type: text/plain

20230327T210036Z
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-signature"
Content-Type: text/plain

2df7efa0c0e0c5b97d0d9483acd77c9ec37360df921b019a4c4a93180a6136ad
--WebAppBoundary
Content-Disposition: form-data; name="x-amz-server-side-encryption"
Content-Type: text/plain

AES256
--WebAppBoundary
Content-Disposition: form-data; name="file";
filename="actual_file_name.png"
Content-Type: image/png

... contents of file goes here ...
--WebAppBoundary--
```

## errors

- 201 – Not found in the database - if there is no task with such an ID, or task doesn't have form, or form has no field with such a field\_id.
- 231 – Entity type mismatch - if form field is not file-based, i.e. doesn't use file ID as its value.
- 255 – Invalid task state - if current task state is not "unassigned", "assigned" or "arrived", or if task's form not submitted at least once.
- 267 – Too many entities - if there are 6 or more unsubmitted files already associated with this form's field.
- 268 – File cannot be created due to quota violation.
- 271 - File size is larger than the maximum allowed (by default 16 MB).

Last update: March 27, 2023





# Recurring tasks

Some tasks happen on regular basis, and it's tedious to create them by hand every time. Task schedules is the way to automate this process. At the beginning of each day (moments after 00:00 AM according to [user's timezone setting](#)), schedule checked and if there are tasks which start at this day, they are created and assigned to employees (if assignee specified).

Schedule entries are very similar to tasks, main difference is that `from` and `to` containing specific date and time replaced with `from_time`, `duration` and `parameters`.

## Task schedule entry object

```
{
  "id": 111,
  "user_id": 3,
  "tracker_id": 22,
  "location": {
    "lat": 53.787154,
    "lng": 9.757980,
    "address": "Moltkestrasse 32",
    "radius": 150
  },
  "label": "Shop",
  "description": "Buy things",
  "from_time": "12:34:00",
  "duration": 60,
  "max_delay": 5,
  "min_stay_duration": 0,
  "min_arrival_duration": 0,
  "parameters": {
    "type": "weekdays",
    "weekdays": [1, 5, 6]
  },
  "tags": [1, 2],
  "form_template_id": 1
}
```

- `id` - int. Primary key. Used in the update call, *IGNORED* in create.
- `user_id` - int. User ID. *IGNORED* in create/update.
- `tracker_id` - int. An ID of the tracker to which all generated tasks assigned. Nullable.



- `location` - location associated with this task. Cannot be null.
  - `address` - string. Address of the location.
  - `radius` - int. Radius of location zone in meters.
- `from_time` - string time. Time of day which defines start of the task within the days.
- `duration` - int. Total duration in minutes between "from" and "to" for generated tasks.
- `max_delay` - int. Maximum allowed task completion delay in minutes.
- `min_stay_duration` - int. Minimum duration of stay in task zone for task completion, minutes.
- `min_arrival_duration` - int. Visits less than these values will be ignored, minutes.
- `parameters` - schedule parameters can be "weekdays" or "month\_days". Described below.
- `tags` - int array. List of tag IDs.
- `form_template_id` - int. Form template ID. Nullable.

## API actions

API base path: `task/schedule`.

### create

Creates new task schedule entry.

**required sub-user rights:** `task_update`.

### parameters

| name     | description  | type        |
|----------|--|-------------|
| schedule | <code>schedule_entry</code> object without fields which are <i>IGNORED</i> . | JSON object |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/schedule/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "schedule":
{"tracker_id": 22, "location": {"lat": 53.787154, "lng": 9.757980,
"address": "Moltkestrasse 32", "radius": 150}, "label": "Shop",
"description": "Buy things", "from_time": "12:34:00", "duration":
60, "max_delay" : 5, "min_stay_duration": 0,
"min_arrival_duration": 0, "parameters": {"type": "weekdays",
"weekdays": [1, 5, 6]}, "tags": [1, 2], "form_template_id": 1}'
```

## response

```
{
  "success": true,
  "id": 111
}
```

- `id` - int. An ID of the created schedule entry.

## errors

- 201 – Not found in the database - if schedule.tracker\_id belongs to nonexistent tracker.
- 204 – Entity not found - if schedule.form\_template\_id belongs to nonexistent form template.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 236 – Feature unavailable due to tariff restrictions - if device's tariff does not allow usage of tasks.

## delete

Delete task schedule with the specified ID.

**required sub-user rights:** `task_update`.

## parameters

| name        | description                        | type |
|-------------|------------------------------------|------|
| schedule_id | ID of the task schedule to delete. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/schedule/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
    "schedule_id": 23144}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/schedule/delete?
hash=a6aa75587e5c59c32d347da438505fc3&schedule_id=23144
```

## response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - if there is no task schedule with such an ID.

## list

Get all task or route schedules belonging to user with optional filtering.  
Also this call returns all unassigned task schedules.

## parameters

| name     | description  | type         |
|----------|--|--------------|
| trackers | Optional. IDs of the trackers to which task schedule is assigned.  | int array    |
| filter   | Optional. Filter for task schedule label and description.  | string       |
| tag_ids  | Optional. Tag IDs assigned to the task schedule. The schedules found must include all the tags from the list.                                  | int array    |
| types    | Optional. Tasks of specific type will be returned in the list. Can be <code>task</code> or <code>route</code> . Default is <code>task</code> . | string array |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/schedule/list' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/schedule/list?  
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{  
  "success": true,  
  "list": [{  
    "id": 111,  
    "user_id": 3,  
    "tracker_id": 22,  
    "location": {  
      "lat": 53.787154,  
      "lng": 9.757980,  
      "address": "Moltkestrasse 32",  
      "radius": 150  
    },  
    "label": "Shop",  
    "description": "Buy things",  
    "from_time": "12:34:00",  
    "duration": 60,  
    "max_delay": 5,  
    "min_stay_duration": 0,  
    "min_arrival_duration": 0,  
    "parameters": {  
      "type": "weekdays",  
      "weekdays": [1, 5, 6]  
    },  
    "tags": [1, 2],  
    "form_template_id": 1  
  }]  
}
```

## errors

General types only.

## read

Gets task, route or checkpoint schedule by specified ID.

## parameters

| name | description                                  | type |
|------|--|------|
| id   | An ID of task, route or checkpoint schedule. | int  |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/schedule/read' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id": 12314}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/schedule/read?  
hash=a6aa75587e5c59c32d347da438505fc3&id=12314
```

## response

```
{  
  "success": true,  
  "value": {  
    "id": 111,  
    "user_id": 3,  
    "tracker_id": 22,  
    "label": "Shop",  
    "description": "Buy things",  
    "parameters": {  
      "type": "weekdays",  
      "weekdays": [1, 5, 6]  
    },  
  },  
  "checkpoints": [{  
    "id": 111,  
    "user_id": 3,  
    "tracker_id": 22,  
    "label": "Shop",  
    "description": "Buy things",  
    "parent_id": 1,  
    "order": 0,  
    "location": {  
      "lat": 53.787154,  
      "lng": 9.757980,  
      "address": "Moltkestrasse 32",  
      "radius": 150  
    },  
    "max_delay": 5,  
    "min_stay_duration": 0,  
    "min_arrival_duration": 0,  
    "from_time": "12:34:00",  
    "duration": 60,  
  }  
}]
```

```
    "tags": [1, 2],
    "form_template_id": 1
  }]
}
```

- `value` - object.
- `checkpoints` - if value is .

## errors

[General](#) types only.

## update

Updates existing task schedule.

**required sub-user rights:** `task_update`.

## parameters

| name     | description  | type        |
|----------|--|-------------|
| schedule | <code>schedule_entry</code> object without fields which are <i>IGNORED</i> . | JSON object |

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/schedule/update' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "schedule":  
{"tracker_id": 22, "location": {"lat": 53.787154, "lng": 9.757980,  
"address": "Moltkestrasse 32", "radius": 150}, "label": "Shop",  
"description": "Buy things", "from_time": "12:34:00", "duration":  
60, "max_delay" : 5, "min_stay_duration": 0,  
"min_arrival_duration": 0, "parameters": {"type": "weekdays",  
"weekdays": [1, 5, 6]}, "tags": [1, 2], "form_template_id": 1}'
```

## response

```
{  
  "success": true  
}
```

## **errors**

- 201 – Not found in the database - if schedule.tracker\_id belongs to nonexistent tracker.
- 204 – Entity not found - if there is no task schedule with specified ID.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 236 – Feature unavailable due to tariff restrictions - if device's tariff does not allow usage of tasks.

Last update: January 15, 2024







# Scheduling routes

These actions allow creating scheduled routes similarly to regular routes.

## Route schedule entry

```
```json { "id": 111, "user_id": 3, "tracker_id": 22, "label": "Shop", "description": "Buy things",  
"parameters": { "type": "month_days", "month_days": [1, 10, 31] } }
```

```
* `id` - int. Primary key. Used in the update call, *IGNORED* in  
create.  
* `user_id` - int. User id. *IGNORED* in create/update.  
* `tracker_id` - int. An ID of the tracker to which all generated  
tasks assigned. Nullable.  
* `parameters` - schedule parameters can be "weekdays" or  
"month_days". Described below.
```

\*\*\*

## Checkpoint schedule entry

```
```json  
{  
  "id": 111,  
  "user_id": 3,  
  "tracker_id": 22,  
  "label": "Shop",  
  "description": "Buy things",  
  "parent_id": 1,  
  "order": 0,  
  "location": {  
    "lat": 53.787154,  
    "lng": 9.757980,  
    "address": "Moltkestrasse 32",  
    "radius": 150  
  },  
  "max_delay" : 5,  
  "min_stay_duration": 0,  
  "min_arrival_duration": 0,  
  "from_time": "12:34:00",  
  "duration": 60,  
  "tags": [1, 2],  
  "form_template_id": 1  
}
```

- `id` - int. Primary key. Used in the update call, *IGNORED* in create.
- `user_id` - int. User id. *IGNORED* in create/update.

- `tracker_id` - int. An ID of the tracker to which all generated tasks assigned. Nullable.
- `location` - location associated with this task. Cannot be null.
- `address` - string. Address of the location.
- `radius` - int. Radius of location zone in meters.
- `max_delay` - int. Maximum allowed task completion delay in minutes.
- `min_stay_duration` - int. Minimum duration of stay in task zone for task completion, minutes.
- `min_arrival_duration` - int. Visits less than these values will be ignored, minutes.
- `from_time` - string time. Time of day which defines start of the task within the days.
- `duration` - int. Total duration in minutes between "from" and "to" for generated tasks.
- `tags` - int array. List of tag IDs.
- `form_template_id` - int. Form template id. Nullable.

`<schedule_parameters>` can be one of the following:

- `weekdays` - task creation based on week day.

```
{
  "type": "weekdays",
  "weekdays": [1, 5, 6]
}
```

\* ``weekdays`` - int array. Week days on which tasks will be created (1 = Monday, ... 7 = Sunday)

- `month_days` - task creation based on day of month.

```
{
  "type": "month_days",
  "month_days": [1, 10, 31]
}
```

\* ``month_days`` - int array. Days of month on which tasks will be created (1..31).

## API actions

API base path: `/task/schedule/route`.

### create

Creates route schedule with checkpoints.

#### parameters

name	description	type
route	<a href="#">Route schedule entry</a> without fields which are <i>IGNORED</i> .	JSON object
checkpoints	Array of route's <a href="#">checkpoints</a> without fields which are <i>IGNORED</i> .	JSON object

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/schedule/route/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "route": {"tracker_id": 22, "label": "Shop", "description": "Buy things", "parameters": {"type": "month_days", "month_days": [1, 10, 31]}}, "checkpoints": [{"tracker_id": 22, "label": "Shop", "description": "Buy things", "parent_id": 1, "order": 0, "location": {"lat": 53.787154, "lng": 9.757980, "address": "Moltkestrasse 32", "radius": 150}, "max_delay": 5, "min_stay_duration": 0, "min_arrival_duration": 0, "from_time": "12:34:00", "duration": 60, "tags": [1, 2], "form_template_id": 1}]}'
```

### response

```
{
  "success": true,
  "id": 111
}
```

- `id` - int. An ID of the created route schedule entry.

### errors

[General](#) types.

## delete

Deletes route schedule with checkpoints.

### parameters

name	description	type
id	Route schedule ID.	int

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/schedule/route/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id": 23144}'
```

#### HTTP GET

```
https://api.navixy.com/v2/task/schedule/route/delete?
hash=a6aa75587e5c59c32d347da438505fc3&id=23144
```

### response

```
{
  "success": true
}
```

### errors

[General](#) types.

## update

Updates route schedule with checkpoints. If checkpoint is being created, then it should have no id. If checkpoint is being updated, then it should have an ID. If old checkpoint is not present in request, then it will be deleted.

### parameters

name	description	type
route		

name	description	type
	<a href="#">Route schedule entry</a> without fields which are <i>IGNORED</i> .	JSON object
checkpoints	Array of route's <a href="#">checkpoints</a> without fields which are <i>IGNORED</i> .	JSON object

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/schedule/route/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "route": {"id": 111, "tracker_id": 22, "label": "Shop", "description": "Buy things", "parameters": {"type": "month_days", "month_days": [1, 10, 31]}}, "checkpoints": {"id": 111, "tracker_id": 22, "label": "Shop", "description": "Buy things", "parent_id": 1, "order": 0, "location": { "lat": 53.787154, "lng": 9.757980, "address": "Moltkestrasse 32", "radius": 150}, "max_delay" : 5, "min_stay_duration": 0, "min_arrival_duration": 0, "from_time": "12:34:00", "duration": 60, "tags": [1, 2], "form_template_id": 1}}'
```

## response

```
{ "success": true }
```

## errors

[General](#) types.

Last update: January 15, 2024







# Task schedule checkpoints

These actions allow manipulating schedule checkpoint entries similarly to regular route checkpoints.

## API actions

API path: `/task/schedule/checkpoint`.

### delete

Deletes a checkpoint from route and reorder others.

If route has two checkpoints then use transmute on the other checkpoint, because route must have at least two checkpoints.

#### parameters

name	description	type
checkpoint_id	Checkpoint ID.	int

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/schedule/checkpoint/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
    "checkpoint_id": 11231}'
```

##### HTTP GET

```
https://api.navixy.com/v2/task/schedule/checkpoint/delete?
hash=a6aa75587e5c59c32d347da438505fc3&checkpoint_id=11231
```

#### response

```
{ "success": true }
```

#### errors

[General](#) types only.

## transmute

Transmutes a checkpoint to task and delete its route and other checkpoints in the route.

### parameters

name	description	type
checkpoint_id	Checkpoint ID.	int

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/schedule/checkpoint/transmute' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",  
    "checkpoint_id": 11231}'
```

#### HTTP GET

```
https://api.navixy.com/v2/task/schedule/checkpoint/transmute?  
hash=a6aa75587e5c59c32d347da438505fc3&checkpoint_id=11231
```

### response

```
{ "success": true }
```

### errors

[General](#) types only.

Last update: December 26, 2022





# Schedule proposals

Schedule proposals are "preview" of what tasks and routes will be created at the specified date range.

## API actions

API base path: `/task/schedule/proposal`.

### list

Get all tasks and routes that will be created by schedule.

#### parameters

name	description	type
trackers	Optional. IDs of the trackers to which task is assigned.	int array
from	Show tasks that will be created AFTER this date, e.g. "2014-07-01 00:00:00", should not before now.	<a href="#">date/</a> <a href="#">time</a>
to	Show tasks will be created BEFORE this date, e.g. "2014-07-01 00:00:00", should not before <code>from</code> .	<a href="#">date/</a> <a href="#">time</a>
filter	Optional. Filter for task schedule label and description.	string
types	Optional. Tasks or routes. For example: <code>["task", "route"]</code> .	<a href="#">enum</a> array

- If `trackers`, `filter`, `from` or `to` is not passed or *null* then appropriate condition not used to filter results.

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/schedule/proposal/
list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "from":
"2020-11-24 00:00:00", "to": "2020-11-25 00:00:00"}'
```

## response

```
{
  "success": true,
  "list": [{
    "id": 111,
    "user_id": 3,
    "tracker_id": 22,
    "location": {
      "lat": 51.283546,
      "lng": 7.301086,
      "address": "Fichtenstrasse 11",
      "radius": 150
    },
    "label": "Deliver parcels",
    "description": "Quickly",
    "creation_date": "2014-01-02 03:04:05",
    "from": "2014-02-03 04:05:06",
    "to": "2014-03-04 05:06:07",
    "external_id": null,
    "status": "assigned",
    "status_change_date": "2014-01-02 03:04:05",
    "max_delay": 5,
    "min_stay_duration": 0,
    "arrival_date": "2014-01-02 03:04:05",
    "stay_duration": 0,
    "origin": "imported",
    "tags": [1, 2],
    "type": "task",
    "form": <form_object>
  }]
}
```

## errors

General types only.

Last update: January 15, 2024







# Task history

Our platform tracks changes to task fields and state for your convenience. Contains API calls to get this information.

## History entry

```
{
  "id": 22,
  "user_id": 3,
  "task_id": 1,
  "event_date": "2014-08-05 10:54:55",
  "operation": "assign",
  "payload": {
    "tracker_id": 2470
  }
}
```

- `id` - int. Entry ID.
- `user_id` - int. User ID.
- `task_id` - int. An ID of the task with which this entry associated.
- `event_date` - [date/time](#). Date when history event happened.
- `operation` - [enum](#). Operation which happened. Can be "create", "update", "assign" or "status\_change".
- `payload` - depends on operation. Typically, contains fields which were changed during operation.

## API actions

API base path: `task/history`.

### list

Returns history for the task with the specified ID.

## parameters

name	description	type
task_id	ID of the task.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/task/checkpoint/delete' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",  
    "checkpoint_id": 23144}'
```

### HTTP GET

```
https://api.navixy.com/v2/task/checkpoint/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&checkpoint_id=23144
```

## response

```
{  
  "success": true,  
  "list": [{  
    "id": 22,  
    "user_id": 3,  
    "task_id": 1,  
    "event_date": "2014-08-05 10:54:55",  
    "operation": "assign",  
    "payload": {  
      "tracker_id": 2470  
    }  
  }]  
}
```

## errors

- [General](#) types only.

Last update: December 26, 2022





# Garage

Contains garage object and API calls to interact with it. Depot (garage object) contains name, address, name of the mechanic, name of the dispatcher and others. This data can be used for more convenient and efficient maintenance and task management.

## Garage object

```
{
  "id": 222,
  "location": {
    "lat": 40.4,
    "lng": -3.6,
    "address": "Calle Salitre, 58",
    "radius": 150
  },
  "mechanic_name": "Martinez",
  "dispatcher_name": "Velasquez",
  "organization_name": "Bankia"
}
```

- `id` - int. Depot ID.
- `location` - location object. Valid location or null.
- `mechanic_name` - string. Mechanic name or null.
- `dispatcher_name` - string. Dispatcher name or null.
- `organization_name` - string. Organization name or null.

## API actions

API path: `/garage`.

### list

Gets all depots belonging to user.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/garage/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

### HTTP GET

```
https://api.navixy.com/v2/garage/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [{
    "id": 222,
    "location": {
      "lat": 40.4,
      "lng": -3.6,
      "address": "Calle Salitre, 58",
      "radius": 150
    },
    "mechanic_name": "Martinez",
    "dispatcher_name": "Velasquez",
    "organization_name": "Bankia"
  }]
}
```

## errors

[General](#) types only.

## create

Creates a new depot.

**required sub-user rights:** `vehicle_update`.

## parameters

name	description	type
garage	An <a href="#">garage object</a> without <code>id</code> field.	JSON object

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/garage/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "garage":
{"location": {"lat": 40.4, "lng": -3.6, "address": "Calle Salitre,
58", "radius": 150}, "mechanic_name": "Martinez",
"dispatcher_name": "Velasquez", "organization_name": "Bankia"}}'
```

## response

```
{
  "success": true,
  "id": 111
}
```

- `id` - int. An ID of a created depot.

## errors

[General](#) types only.

## update

Updates existing depot with the specified ID.

**required sub-user rights:** `vehicle_update`.

## parameters

name	description	type
garage	An <a href="#">garage object</a> with <code>id</code> field.	JSON object

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/garage/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "garage":
{"id": 222, "location": {"lat": 40.4, "lng": -3.6, "address":
"Calle Salitre, 58", "radius": 150}, "mechanic_name": "Martinez",
"dispatcher_name": "Velasquez", "organization_name": "Bankia"}}'
```

## response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - if there is no depot with such an ID.

## delete

Deletes a depot with the specified ID.

**required sub-user rights:** `vehicle_update`.

## parameters

name	description	type
garage_id	ID of the depot to delete.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/garage/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "garage_id": 111}'
```

### HTTP GET

```
https://api.navixy.com/v2/garage/delete?
hash=a6aa75587e5c59c32d347da438505fc3&garage_id=111
```

## response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - if there is no depot with such an ID.

Last update: December 26, 2022







# Driver journal entry

Contains driver journal entry object description and API calls to work with it. Using the driver journal, you can monitor trips and categorize them by status to see the full picture of transport usage. Driver Entry is an already categorized trip.

To get information on how-to work with driver journals refer to our [instructions](#).

## Driver journal entry object

```
{
  "id": 127722,
  "tracker_id": 1,
  "start_date": "2020-10-13 07:03:39",
  "end_date": "2020-10-14 08:05:02",
  "employee_id": 1,
  "type": "work",
  "comment": "comment string",
  "start_location": {
    "lat": 11.0,
    "lng": 22.0,
    "address": "address value"
  },
  "end_location": {
    "lat": 11.0,
    "lng": 22.0,
    "address": "address value"
  },
  "length": 1.44,
  "start_odometer": 1.34,
  "end_odometer": 5.34
}
```

- `id` - int. An ID of an entry.
- `tracker_id` - int. An ID of the tracker (aka "object\_id"). Tracker must belong to authorized user and not be blocked.
- `start_date` - [date/time](#). Start date of a journal entry.
- `end_date` - [date/time](#). End date of a journal entry.
- `employee_id` - nullable int. An ID of employee (driver).
- `type` - [enum](#). Type of journal entry. Can be "work", "personal", "other".
- `comment` - nullable string. Comment for entry.
- `start_location` - location object. Where entry starts.

- `end_location` - location object. Where entry ends.
- `length` - float. Length of the trip km.
- `start_odometer` - nullable float. Odometer's value at the start.
- `end_odometer` - nullable float. Odometer's value at the end.

## API actions

API path: `/driver/journal/entry`.

### list

Gets driver journal entries. There are two ways to get entries: by their IDs or by specifying date range. If there are no `entry_ids` in request, entries will be selected by intersecting their date range with date range from request ( `from` and `to` parameters).

#### parameters

name	description	type
<code>tracker_id</code>	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int
<code>from</code>	Include tracks which end after this date, e.g. "2020-10-13 00:00:00".	<a href="#">date/time</a>
<code>to</code>	Include tracks which end after this date, e.g. "2020-10-14 00:00:00".	<a href="#">date/time</a>
<code>entry_ids</code>	Optional. Array of entry IDs.	int array
<code>types</code>	Optional. Types of the driver journal entry, e.g. <code>["work", "personal", "other"]</code> .	string array

name	description	type
sort	Optional. Set of sort options. Each option is a pair of column name and sorting direction, e.g. ["start_date=asc", "type=desc"] .	string array

• Possible columns of `sort` parameter:

- `start_date` - Sort only by date, not considering time part.
- `start_datetime` - Just raw column value.
- `end_date` - Sort only by date, not considering time part.
- `end_datetime` - Just raw column value.
- `start_address` - Sort only by start address.
- `end_address` - Sort only by the end address.
- `driver` - Sort by last+first+middle driver name, not by driver ID.
- `type` - Sort by type.
- If no `sort` param is specified, then `sort` option will be "start\_date=asc".

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/driver/journal/entry/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 123456, "from": "2020-10-13 00:00:00", "to": "2020-10-14 00:00:00"}'
```

## response

```
{
  "success": true,
  "list": [{
    "id": 127722,
    "tracker_id": 1,
    "start_date": "2020-10-13 07:03:39",
    "end_date": "2020-10-14 08:05:02",
    "employee_id": 1,
    "type": "work",
    "comment": null,
    "start_location": {
      "lat": 23.25658,
      "lng": 21.89892,
      "address": "address"
    }
  }],
}
```

```

        "end_location": {
            "lat": 23.26227,
            "lng": 21.59321,
            "address": "address"
        },
        "length": 1.44,
        "start_odometer": 1.34,
        "end_odometer": 5.34
    }
}

```

## errors

- [General](#) types only.

## create

Creates driver journal entries.

## parameters

name	description	type
entries	Array of <code>driver_journal_entry</code> objects without <code>id</code> field.	array of objects

## example

### cURL

```

curl -X POST 'https://api.navixy.com/v2/driver/journal/entry/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "entries": [
    {"tracker_id": 1, "start_date": "2020-10-13 07:03:39", "end_date": "2020-10-14 08:05:02", "employee_id": 1, "type": "work", "comment": "comment string", "start_location": {"lat": 11.0, "lng": 22.0, "address": "address value"}, "end_location": {"lat": 11.0, "lng": 22.0, "address": "address value"}, "length": 1.44, "start_odometer": 1.34, "end_odometer": 5.34}}]'

```

## response

```
{ "success": true }
```

## errors

- [General](#) types.

## update

Updates driver journal entry. Only two fields `type` and `comment` are available to update.

### parameters

name	description	type
entry	<code>driver_journal_entry_update_request</code> type. See below.	JSON object

- `driver_journal_entry_update_request` object:

```
{
  "id": 1,
  "type": "work",
  "comment": "new comment"
}
```

- `id` - int. An ID of the driver journal entry.
- `type` - [enum](#). Type of journal entry. Can be "work", "personal", "other".
- `comment` - string. New comment of the driver journal entry.

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/driver/journal/entry/
update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "entry":
{"id": 1, "type": "work", "comment": "new comment"}}'
```

### response

```
{ "success": true }
```

### errors

- 201 – if tracker not found
- 204 - if entry not found.
- [General](#) types only.

## delete

Deletes driver journal entries.

### parameters

name	description	type
entry_ids	Array of driver journal entries' IDs.	int array

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/driver/journal/entry/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "entry_ids": [127722, 127724]}'
```

#### HTTP GET

```
https://api.navixy.com/v2/driver/journal/entry/delete?
hash=a6aa75587e5c59c32d347da438505fc3&entry_id=[127722, 127724]
```

### response

```
{ "success": true }
```

### errors

- [General](#) types only.

## download

Gets driver journal entries. Entries selected by intersecting their date range with date range from request ( `from` and `to` parameters).

### parameters

name	description	type
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int



name	description	type
from	Include tracks which end after this date, e.g. "2020-10-13 00:00:00".	<a href="#">date/time</a>
to	Include tracks which end after this date, e.g. "2020-10-14 00:00:00".	<a href="#">date/time</a>
entry_ids	Optional. Array of entry IDs.	int array
types	Optional. Types of the driver journal entry, e.g. <code>["work", "personal", "other"]</code> .	string array
sort	Optional. Set of <a href="#">sort options</a> . Each option is a pair of column name and sorting direction, e.g. <code>["start_date=asc", "type=desc"]</code> .	string array
add_filename_header	If <code>true</code> then Content-Disposition header will be appended to the response. Default value is <code>true</code> .	boolean
format	File format: "pdf", "xls" and "xlsx".	string
group_by	Optional. If specified, grouped entries will be in different sections of the table.	string

- Possible values of `group_by` parameter:
  - `type` - group entries by entry type.
  - `date` - group entries by start\_date per day.

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/driver/journal/entry/download' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 123456, "from": "2020-10-13 00:00:00", "to": "2020-10-14 00:00:00", "add_filename_header": true, "format": "pdf"}'
```

## response

A driver journal report file (standard file download).

## errors

- [General](#) types only.

Last update: April 8, 2024





# Trip proposal for driver journal

Contains API call to get the list of driver journal proposal. Proposal objects - trips per specified period that could be used for driver journal entry creation.

To get information on how-to work with driver journals refer to our [instructions](#).

## API actions

API path: `/driver/journal/proposal`.

### list

Gets proposal trips that could be used for driver journal entry creation. Proposal objects created by a track's division by driver changes. If there was no driver change on the track, then the track will be returned entirely. Tracks selected by intersecting their date range with date range from request ( `from` and `to` parameters).

### parameters

name	description	type
from	Include tracks which end after this date.	string
to	Include tracks which start before this date.	string
tracker_id	ID of the tracker.	int

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/driver/journal/proposal/
list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
123456, "from": "2020-10-13 00:00:00", "to": "2020-10-14
00:00:00"}'
```

### response

```

{
  "success": true,
  "list": [{
    "tracker_id": 1,
    "employee_id": 1,
    "start_date": "2020-10-14 07:03:39",
    "end_date": "2020-10-15 08:05:02",
    "start_location": {
      "lat": 11.111111,
      "lng": 22.222222,
      "address": "Address string"
    },
    "end_location": {
      "lat": 33.333333,
      "lng": 44.444444,
      "address": "Address string"
    },
    "length": 2.1,
    "start_odometer": 50.2,
    "end_odometer": 52.0,
    "overlapped": false
  }]
}

```

- `tracker_id` - int. An ID of the tracker (aka "object\_id"). Tracker must belong to authorized user and not be blocked.
- `employee_id` - nullable int. An ID of employee (driver).
- `start_date` - [date/time](#). Start date of a journal entry.
- `end_date` - [date/time](#). End date of a journal entry.
- `start_location` - location object. Where entry starts.
- `end_location` - location object. Where entry ends.
- `length` - float. Length of the trip km.
- `start_odometer` - nullable float. Odometer's value at the start.
- `end_odometer` - nullable float. Odometer's value at the end.
- `overlapped` - boolean. `true` if there is already driver journal entry with date range which is intersecting this proposal object's date range.

#### errors

- [General](#) types only.

Last update: August 1, 2023







# Vehicle

Contains the vehicle object and API calls to interact with it. This object is used to describe vehicle's information like VIN, speed, consumption and other. Vehicle object should be assigned to tracker object.

## Vehicle object

```
{
  "id": 222,
  "tracker_id": 1,
  "tracker_label": "Jimi LL301",
  "label": "AGV",
  "max_speed": 90,
  "model": "Renault KERAX",
  "type": "truck",
  "subtype": "tractor",
  "garage_id": 1,
  "garage_organization_name": "Fleet Systems",
  "trailer" : "trailer1",
  "manufacture_year" : 2001,
  "color" : "some color",
  "additional_info" : "additional info",
  "reg_number": "A001AA96",
  "vin": "TMBJF25LXC6080000",
  "chassis_number": "",
  "frame_number" : "",
  "payload_weight": 32000,
  "payload_height": 1.2,
  "payload_length": 1.0,
  "payload_width": 1.0,
  "passengers": 4,
  "gross_weight" : null,
  "fuel_type": "petrol",
  "fuel_grade": "A-80",
  "norm_avg_fuel_consumption": 9.0,
  "fuel_tank_volume": 50,
  "fuel_cost" : 100.3,
  "wheel_arrangement": "4x2",
  "tyre_size": "255/65 R16",
  "tyres_number": 4,
  "liability_insurance_policy_number": "12345",
  "liability_insurance_valid_till": "2020-10-15",
  "free_insurance_policy_number": "",
  "free_insurance_valid_till": null,
  "icon_id" : 55,
  "avatar_file_name": null,
```

```
"tags": [1,2]
}
```

- `id` - int. An ID of a vehicle.
- `tracker_id` - int. An ID of the tracker (aka "object\_id"). Tracker must belong to authorized user and not be blocked.
- `tracker_label` - optional string. Tracker's label.
- `label` - string. Vehicle's label.
- `max_speed` - int. Maximum speed of a vehicle.
- `model` - string. Vehicle's model.
- `type` [enum](#). Vehicle's type. Can be "truck" | "car" | "bus" | "special".
- `subtype` - optional [enum](#). Depends on type, null means undefined. Possible subtypes listed below.
- `garage_id` - nullable int. An ID of a garage.
- `garage_organization_name` - optional string. Garage organization name.
- `trailer` - optional string. Information about a trailer.
- `manufacture_year` - optional int. Manufacture year of a vehicle.
- `color` - optional string. Not RGB. A color of a vehicle.
- `additional_info` - optional string. Additional info about a vehicle.
- `reg_number` - string. Reg number/ license plate of a vehicle.
- `vin` - string. VIN of a vehicle.
- `chassis_number` - string. Chassis number of a vehicle.
- `frame_number` - optional string. Frame number of a vehicle.
- `payload_weight` - int. Payload weight in kilograms.
- `payload_height` - decimal. Payload height in millimeters.
- `payload_length` - decimal. Payload length in millimeters.
- `payload_width` - decimal. Payload width in millimeters.
- `passengers` - int. A maximum count of passengers.
- `gross_weight` - optional int. Gross weight in kilograms.
- `fuel_type` - [enum](#). Can be "petrol" | "diesel" | "gas".
- `fuel_grade` - string. Grade of fuel used in a vehicle.
- `norm_avg_fuel_consumption` - decimal. Normal average fuel consumption in liters per 100 km.
- `fuel_tank_volume` - int. Fuel tank capacity in liters.

- `fuel_cost` - optional decimal. Cost of fuel used in a vehicle per liter.
- `wheel_arrangement` - string. Wheel arrangement of a vehicle.
- `tyre_size` - string. Tyre size.
- `tyres_number` - int. Number of tyres.
- `liability_insurance_policy_number` - string. Liability insurance policy number.
- `liability_insurance_valid_till` - string date. The date till liability insurance valid.
- `free_insurance_policy_number` - string. Free insurance policy number.
- `free_insurance_valid_till` - string date. The date till free insurance valid.
- `icon_id` - nullable int. Can only be updated via [avatar/assign](#).
- `avatar_file_name` - string. File name.
- `tags` - int array. List of tag IDs.

#### Subtypes:

```
Type: "car"
Subtypes: "sedan", "universal", "hatchback", "liftback",
"limousine", "pickup", "minivan", "coupe", "coupe4d", "muscle",
"convertible", "phaeton", "lando", "crossover", "roadster", "suv"
```

```
Type: "truck"
Subtypes: "tipper", "board", "covered", "awning", "mixer",
"tanker", "refrigerator", "transporter", "container", "tractor"
```

```
Type: "bus"
Subtypes: "city", "shuttle", "platform", "school", "intercity",
"sightseeing"
```

```
Type: "special"
Subtypes: "mobile_crane", "racing", "buggy", "ambulance",
"firefighter", "hearse", "shop", "harvester", "snowplow",
"tractor", "grader", "excavator", "bulldozer", "armored",
"amphibian", "boat"
```

## API actions

API path: `/vehicle`.

## create

Creates a new vehicle.

**required sub-user rights:** `vehicle_update`

### PARAMETERS

name	description	type
vehicle	A <a href="#">vehicle object</a> without <code>id</code> field.	JSON object
force_reassign	Optional. Default is <code>true</code> . Will reassign the device to created vehicle even if it was assign to another one.	boolean

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "vehicle":
{"additional_info": null, "avatar_file_name": null,
"chassis_number": "", "color": null, "frame_number": "",
"free_insurance_policy_number": "", "free_insurance_valid_till":
null, "fuel_cost": null, "fuel_grade": "", "fuel_tank_volume":
null, "fuel_type": null, "garage_id": null, "gross_weight": null,
"icon_color": "1E96DC", "icon_id": null, "label": "Vehicle",
"liability_insurance_policy_number": "",
"liability_insurance_valid_till": null, "manufacture_year": 2020,
"max_speed": 160, "model": "", "norm_avg_fuel_consumption": null,
"passengers": 1, "payload_height": 1868, "payload_length": 2820,
"payload_weight": null, "payload_width": 1972, "reg_number":
"AB234D", "subtype": "sedan", "tags": [], "tracker_id": null,
"trailer": null, "type": "car", "tyre_size": "", "tyres_number":
null, "vin": "45468743418579751", "wheel_arrangement": null}}'
```

## response

```
{
  "success": true,
  "id": 111
}
```

- `id` - int. An ID of the created vehicle.

## errors

- 247 – Entity already exists, if tracker\_id!=null and exists a vehicle that already bound to this tracker\_id.

## delete

Deletes a vehicles with the specified IDs. Only one of the following parameters must be specified.

**required sub-user rights:** `vehicle_update`.

## parameters

name	description	type
vehicle_id	ID of the vehicle to delete.	int
vehicle_ids	An array of vehicle IDs to delete.	int array

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/delete' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "vehicle_id":  
127722}'
```

### HTTP GET

```
https://api.navixy.com/v2/vehicle/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&vehicle_id=127722
```

## response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - if there is no vehicle with such an ID. This error will not occur if the vehicle\_ids parameter is specified, deletion is silent in this case.

## list

Gets all vehicles belonging to user.

### PARAMETERS

name	description	type
limit	Pagination. Maximum number of vehicle records to return.	int
offset	Pagination. Get vehicles starting from.	int
sort	Optional. Set of sort options. Each option is a pair of property name and sorting direction, e.g. [ "type=desc", "label=asc" ]. Maximum 2 options in request. Available properties: <ul style="list-style-type: none"><li>- id</li><li>- label</li><li>- reg_number</li><li>- model</li><li>- type</li><li>- garage_organization_name</li><li>- vin</li><li>- tracker_label</li><li>- fuel_type</li><li>- fuel_grade</li><li>- norm_avg_fuel_consumption</li><li>- fuel_tank_volume</li><li>- payload_weight</li><li>- chassis_number</li><li>- subtype</li><li>- wheel_arrangement</li><li>- tyres_number</li><li>- tyres_size</li><li>- max_speed</li><li>- passengers</li></ul>	string array
filter	Optional. Filter vehicles by VIN, reg_number or label. Maximum 100 characters or null.	string

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

### HTTP GET

```
https://api.navixy.com/v2/vehicle/list?  
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{  
  "success": true,  
  "list": [{  
    "id": 222,  
    "tracker_id": 1,  
    "tracker_label": "Jimi LL301",  
    "label": "AGV",  
    "max_speed": 90,  
    "model": "Renault KERAX",  
    "type": "truck",  
    "subtype": "tractor",  
    "garage_id": 1,  
    "garage_organization_name": "Fleet Systems",  
    "trailer" : "trailer1",  
    "manufacture_year" : 2001,  
    "color" : "some color",  
    "additional_info" : "additional info",  
    "reg_number": "A001AA96",  
    "vin": "TMBJF25LXC6080000",  
    "chassis_number": "",  
    "frame_number" : "",  
    "payload_weight": 32000,  
    "payload_height": 1.2,  
    "payload_length": 1.0,  
    "payload_width": 1.0,  
    "passengers": 4,  
    "gross_weight" : null,  
    "fuel_type": "petrol",  
    "fuel_grade": "A-80",  
    "norm_avg_fuel_consumption": 9.0,  
    "fuel_tank_volume": 50,  
    "fuel_cost" : 100.3,  
    "wheel_arrangement": "4x2",  
    "tyre_size": "255/65 R16",  
    "tyres_number": 4,  
    "liability_insurance_policy_number": "12345",  
    "liability_insurance_valid_till": "2020-10-15",  
    "free_insurance_policy_number": "",  
    "free_insurance_valid_till": null,  
    "icon_id" : 55,  
    "avatar_file_name": null,  
    "tags": [1,2]  }]
```

```
}]
}
```

## errors

[General](#) types only.

## read

Gets vehicle by specified ID.

### parameters

name	description	type
vehicle_id	ID of a vehicle.	int

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "vehicle_id": 127722}'
```

#### HTTP GET

```
https://api.navixy.com/v2/vehicle/read?
hash=a6aa75587e5c59c32d347da438505fc3&vehicle_id=127722
```

### response

```
{
  "success": true,
  "value": {
    "id": 222,
    "tracker_id": 1,
    "label": "AGV",
    "max_speed": 90,
    "model": "Renault KERAX",
    "type": "truck",
    "subtype": "tractor",
    "garage_id": null,
    "trailer": "trailer1",
    "manufacture_year": 2001,
    "color": "some color",
    "additional_info": "additional info",
    "reg_number": "A001AA96",
    "vin": "TMBJF25LXC6080000",
  }
}
```



```

    "chassis_number": "",
    "frame_number" : "",
    "payload_weight": 32000,
    "payload_height": 1.2,
    "payload_length": 1.0,
    "payload_width": 1.0,
    "passengers": 4,
    "gross_weight" : null,
    "fuel_type": "petrol",
    "fuel_grade": "A-80",
    "norm_avg_fuel_consumption": 9.0,
    "fuel_tank_volume": 50,
    "fuel_cost" : 100.3,
    "wheel_arrangement": "4x2",
    "tyre_size": "255/65 R16",
    "tyres_number": 4,
    "liability_insurance_policy_number": "12345",
    "liability_insurance_valid_till": "2020-10-15",
    "free_insurance_policy_number": "",
    "free_insurance_valid_till": null,
    "icon_id" : 55,
    "avatar_file_name": null,
    "tags": [1,2]
  }
}

```

A [vehicle](#) object.

#### errors

- 201 – Not found in the database - if there is no vehicle with such an ID.

## update

Updates existing vehicle.

**required sub-user rights:** `vehicle_update`.

#### parameters

name	description	type
vehicle	A <a href="#">vehicle</a> object.	JSON object
force_reassign	Optional. Default is <code>true</code> . Will reassign the device to created vehicle even if it was assign to another one.	boolean

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "vehicle":
{"additional_info": null, "avatar_file_name": null,
"chassis_number": "", "color": null, "frame_number": "",
"free_insurance_policy_number": "", "free_insurance_valid_till":
null, "fuel_cost": null, "fuel_grade": "", "fuel_tank_volume":
null, "fuel_type": null, "garage_id": null, "gross_weight": null,
"icon_color": "1E96DC", "icon_id": null, "id": 223155, "label":
"Vehicle", "liability_insurance_policy_number": "",
"liability_insurance_valid_till": null, "manufacture_year": 2020,
"max_speed": 160, "model": "", "norm_avg_fuel_consumption": null,
"passengers": 1, "payload_height": 1868, "payload_length": 2820,
"payload_weight": null, "payload_width": 1972, "reg_number":
"AB234D", "subtype": "sedan", "tags": [], "tracker_id": null,
"trailer": null, "type": "car", "tyre_size": "", "tyres_number":
null, "vin": "45468743418579751", "wheel_arrangement": null}}'
```

## response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - if there is no vehicle with such an ID.
- 247 – Entity already exists, if tracker\_id!=null and exists a vehicle that already bound to this tracker\_id.
- 261 – Entity has external links - when `tracker_id` changes and there are some service tasks associated with this vehicle.

## batch\_convert

Convert batch of tab-delimited vehicles and return list of checked vehicles with errors.

**required sub-user rights:** `vehicle_update`.

## parameters

name	description	type
batch	Batch of tab-delimited vehicles.	string

name	description	type
file_id	Preloaded file ID.	string
fields	Optional, array of field names, default is ["label", "model", "reg_number", "fuel_grade"].	string array
geocoder	Geocoder type.	string

If `file_id` is set – `batch` parameter will be ignored.

#### response

```
{
  "success": true,
  "list": [<checked_vehicle>],
  "limit_exceeded": false
}
```

- `limit_exceeded` - `true` if given batch constrained by limit.

where `checked_vehicle` is:

```
{
  "id": 222,
  "tracker_id": 1,
  "label": "AGV",
  "max_speed": 90,
  "model": "Renault KERAX",
  "type": "truck",
  "subtype": "tractor",
  "garage_id": null,
  "trailer" : "trailer1",
  "manufacture_year" : 2001,
  "color" : "some color",
  "additional_info" : "additional info",
  "reg_number": "A001AA96",
  "vin": "TMBJF25LXC6080000",
  "chassis_number": "",
  "frame_number" : "",
  "payload_weight": 32000,
  "payload_height": 1.2,
  "payload_length": 1.0,
  "payload_width": 1.0,
  "passengers": 4,
  "gross_weight" : null,
  "fuel_type": "petrol",
  "fuel_grade": "A-80",
  "norm_avg_fuel_consumption": 9.0,
  "fuel_tank_volume": 50,
```

```
"fuel_cost" : 100.3,  
"wheel_arrangement": "4x2",  
"tyre_size": "255/65 R16",  
"tyres_number": 4,  
"liability_insurance_policy_number": "12345",  
"liability_insurance_valid_till": "2020-10-15",  
"free_insurance_policy_number": "",  
"free_insurance_valid_till": null,  
"icon_id" : 55,  
"avatar_file_name": null,  
"tags": [1,2],  
"errors": <array_of_objects>  
}
```

- errors - optional array of objects.

### errors

- 234 - Invalid data format.

Last update: April 8, 2024





# Vehicle avatar

API calls to upload and assign avatar to the vehicle.

## API actions

API path: `/vehicle/avatar`.

### assign

Assigns `icon_id` (from standard icon set) to specified vehicle.

**required sub-user rights:** `vehicle_update`

#### parameters

name	description	type
vehicle_id	ID of the vehicle.	int
icon_id	ID of the icon.	int

`icon_id` can be null – this means that uploaded avatar should be used instead of icon.

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/avatar/assign' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "vehicle_id":  
127722, "icon_id": 1342}'
```

##### HTTP GET

```
https://api.navixy.com/v2/vehicle/avatar/assign?  
hash=a6aa75587e5c59c32d347da438505fc3&vehicle_id=127722&icon_id=1342
```

##### RESPONSE

```
{ "success": true }
```

## ERRORS

- 201 – Not found in the database - when vehicle with `vehicle_id` not found.

## upload

Uploads avatar image for specified vehicle. Then it will be available from

`[api_base_url]/<api_static_path>/vehicle/avatars/<file_name>` e.g. `https://api.navixy.com/v2/static/vehicle/avatars/abcdef123456789.png`.

**required sub-user rights:** `vehicle_update`

**avatar\_file\_name** returned in response and will be returned from </vehicle/list>.

**MUST** be a POST multipart request (multipart/form-data), with one of the parts being an image file upload (with the name "file").

File part **mime** type must be one of :

- `image/jpeg`
- `image/pjpeg`
- `image/png`
- `image/gif`
- `image/webp`

## parameters

name	description	type
vehicle_id	Vehicle ID.	int
file	Image file.	string
redirect_target	Optional. URL to redirect.	string

If `redirect_target` passed a return redirect to `response=<urlencoded response json>`.

## response

```
{
  "success": true,
```



```
"value": "abcdef123456789.png"  
}
```

- `value` - string. Avatar file name.

#### errors

- 201 – Not found in the database - when vehicle with `vehicle_id` not found.
- 233 – No data file - if `file` part not passed.
- 234 – Invalid data format - if passed `file` with unexpected **mime** type.
- 254 – Cannot save file - on some file system errors.

Last update: December 26, 2022





# Vehicle import

## API actions

API calls to import vehicles.

## API actions

API path: `/vehicle/import/`.

### start

Starting the background process of importing vehicles.

#### parameters

name	description	type
filename	Name of file preloaded with <a href="#">/data/spreadsheet/parse</a>	string
headers	List of files' headers, see available fields above	string array
user_headers	Optional. List of user labels for headers	string array

Available fields:

- `label`
- `model`
- `max_speed`
- `type`
- `subtype`
- `garage`
- `trailer`
- `manufacture_year`

- color
- additional\_info
- trailer\_reg\_number
- reg\_number
- chassis\_number
- frame\_number
- vin
- passengers
- payload\_weight
- payload\_height
- payload\_width
- payload\_length
- gross\_weight
- fuel\_grade
- fuel\_tank\_volume
- fuel\_cost
- norm\_avg\_fuel\_consumption
- fuel\_type
- tyre\_size
- tyres\_number
- wheel\_arrangement
- free\_insurance\_policy\_number
- liability\_insurance\_policy\_number
- free\_insurance\_valid\_till
- liability\_insurance\_valid\_till
- tracker\_label
- tags
- undefined (if a meaning of a field is not known)

## response

```
{  
  "success": true,  
  "id": <int>  
}
```

## example

### cURL

```
curl -X POST "https://api.navixy.com/v2/vehicle/import/start" \  
-H "Content-Type: application/json" \  
--data-binary @- << EOF  
{  
  "hash": "a6aa75587e5c59c32d347da438505fc3",  
  "filename": "tmp-sheet640571613016981796.tsv",  
  "headers": ["label", "model", "max_speed", "type", "subtype",  
  "reg_number", "fuel_grade", "fuel_tank_volume",  
  "free_insurance_policy_number", "free_insurance_valid_till",  
  "tracker_label", "tags"],  
  "user_headers": [ "Model", "Max speed", "Type", "Subtype",  
  "Reg. number", "Fuel grade", "Fuel tank volume", "Free insurance  
policy number", "Free insurance valid till", "Object", "Tags"]  
}  
EOF
```

## errors

- 15 - Too many requests (rate limit exceeded) - if too many imports in progress
- 233 - No data file
- 234 - Invalid data format
- 247 - Entity already exists - there is another identical import with the same file

## read

Returns an import process with specified ID.

## parameters

name	description	type
process_id	Process ID	int

## response

```
{  
  "success": true,  
  "value": {  
    "id": <int>,  
    "user_id": <int>,  
    "created": <date>,  
    "type": "Vehicle",  
    "params": {  
      "headers": [<string>, <string>, ...] // List of files' headers
```

```

    },
    "filename": <string>, // Name of preloaded TSV.
    "status": <string>, // created | in_progress | done | failed |
finished
    "status_change_date": <date>,
    "progress": {
        "imported": <int>,
        "failed": <int>,
        "percent": <int>, // approximate percentage of processed
        "processed_lines": <int>,
        "warnings": [{line:<int>, error: <string>}], // first 25
        "errors": [{line:<int>, error: <string>}], // first 25
    }
}
}
}

```

## example

### cURL

```

curl -X POST "https://api.navixy.com/v2/vehicle/import/read" \
-H "Content-Type: application/json" \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "process_id":
1}'

```

## errors

- 201 – Not found in database (if import is not found)

## list

Returns the list of the user's vehicle import processes.

## response

```

{
    "success" : true,
    "list" : [ {
        "id": <int>,
        "user_id": <int>,
        "created": <date>,
        "type": "Vehicle",
        "params": {
            "headers": [<string>, <string>,...] // List of files' headers
        },
        "filename": <string>, // Name of preloaded TSV.
        "status": <string>, // created | in_progress | done | failed
        "status_change_date": <date>,
        "progress": {
            "imported": <int>,
            "failed": <int>,
            "percent": <int>, // approximate percentage of processed
            "processed_lines": <int>,

```

```
"warnings": [{line:<int>, error: <string>}], // first 25
"errors": [{line:<int>, error: <string>}], // first 25
}, ...]
}
```

## example

### cURL

```
curl -X POST "https://api.navixy.com/v2/vehicle/import/list" \
-H "Content-Type: application/json" \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

## download\_failed

Retrieve a file with lines that contained errors and did not pass validation.

## parameters

name	description	type
process_id	Process ID	int

## response

File (standard file download).

## example

### cURL

```
curl -X POST "https://api.navixy.com/v2/vehicle/import/
download_failed" \
-H "Content-Type: application/json" \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "process_id":
7}'
```

## errors

- 201 – Not found in database (if import is not found)
- 204 – Entity not found (if file is not found)

Last update: October 6, 2023







# Vehicle service work file

Contains call for creation the service work file.

## API actions

API path: `/vehicle/service_task/file`.

### create

Creates a file to specify its ID in service work later.

#### parameters

name	description	type
filename	Optional. If specified, uploaded file will have the specified name. If not, name will be taken from actual file upload form.	string
size	Maximum size in bytes for the file which will be uploaded. This is needed to "reserve" the space for file in user's disk space quota.	int
metadata	Optional. Metadata object. See <a href="#">task/form</a> .	JSON object
type	Can be "image" or "file". Default is "file".	enum

#### response

```
{
  "success": true,
  "value": {
    "file_id": 111,
    "url": "http://example.com",
    "expires": "2016-02-03 03:04:00",
    "file_field_name": "file",
    "fields": {
      "token": "skrjsokqsi21lskkw11783s12k5"
    }
  }
}
```

```
}  
}
```

- `file_field_name` - string. Name for file field in POST upload request.
- `fields` - object. These fields should be passed as additional fields in POST upload request, field with a file must be the last one.
  - `token` - string. Used for authentication of upload.

Last update: December 26, 2022





# Vehicle service work

Contains service task object description and API calls to interact with vehicle service works that is used for vehicle maintenance. Vehicle maintenance feature helps to make sure that any scheduled maintenance or urgent repair is carried out in a timely manner.

Described step-by-step about service task APIs in our [guides](#).

## Service task object

```
{
  "id": 725,
  "vehicle_id": 222,
  "status": "created",
  "prediction": {
    "end_date": "2015-05-03 09:35:00",
    "wear_percentage": 40
  },
  "description": "Service work",
  "comment": "",
  "cost": 100500.0,
  "completion": {
    "mileage": 31,
    "date": "2014-03-16 00:00:00",
    "engine_hours": 140
  },
  "conditions": {
    "mileage": {
      "limit": 100,
      "notification_interval": 10
    },
    "date": {
      "end": "2015-05-08 09:35:00",
      "notification_interval": 3,
      "repeat_interval": 42
    },
    "engine_hours": {
      "limit": 100,
      "notification_interval": 10
    }
  },
  "start": {
    "mileage": 1230,
    "date": "2015-05-01 17:46:44",
    "engine_hours": 50
  },
  "notifications": {
    "sms_phones": [
      "79221234567",
```

```

    "79227654321"
  ],
  "emails": [
    "email@domain.tld",
    "email@mail.com"
  ],
  "push_enabled": true
},
"completion_date" : "2014-03-16 00:00:00",
"repeat": true,
"unplanned": false,
"file_ids": [1, 2]
}

```

- `id` - int. An ID of created task.
- `vehicle_label` - string. Vehicle label.
- `status` - [enum](#). [Status](#).
- `prediction` - optional object. Legacy field, is not used anymore. check `return_prediction` parameter.
  - `end_date` - [date/time](#). Predicted end date.
  - `wear_percentage` - int. Wear percentage.
- `completion` - object. Date and counter's values when the task marked as done. Non-editable.
- `completion_date` - [date/time](#). Date when a service work completed.
- `current_position` - object. Current position values.
  - `mileage` - int. Current mileage.
  - `date` - [date/time](#). Current date.
  - `engine_hours` - int. Current engine hours.
- `start` - object. Consists initial values.
  - `mileage` - int. Initial odometer value for tasks with mileage condition.
  - `date` - [date/time](#). Task creation date for tasks with date condition.
  - `engine_hours` - int. Initial engine hours value for tasks with engine hours condition.
- `vehicle_id` - int. An ID of associated vehicle.
- `description` - string. Name of a service work. Max 255 characters.
- `comment` - string. Comment for a task. Max 255 characters.
- `cost` - float. Cost in the currency of the user. For information only.



- `conditions` - task end conditions. At least one of fields ("mileage" or "date" or "engine\_hours") must be passed.
  - `mileage` - optional object. Mileage condition.
    - `limit` - int. Task limit in kilometers.
    - `notification_interval` - int. Notify about task in specified number of kilometers.
    - `repeat_interval` - int. Interval in kilometers to set `limit` for a new repeatable task when current one is completed. If this parameter is not set, the initial `limit` value will be used.
  - `date` - optional date condition object.
    - `end` - `date/time`. Task end date.
    - `notification_interval` - int. Notify about task in specified number of days.
    - `repeat_interval` - int. Interval in days to calculate a new end date for repeatable tasks when they are completed. If this parameter is not specified, the interval will be calculated as the difference between the start date and the end date.
  - `engine_hours` - optional engine hours condition object.
    - `limit` - int. Task limit in hours.
    - `notification_interval` - int. Notify about task in specified number of hours.
    - `repeat_interval` - int. Interval in hours to set `limit` for a new repeatable task when current one is completed. If this parameter is not set, the initial `limit` value will be used.
- `notifications` - notifications object.
  - `sms_phones` - string array. Phones where sms notifications should be sent. In the international format wo `+` sign.
  - `emails` - string array. Email addresses where sms notifications should be sent.
  - `push_enabled` - boolean. If `true` push notifications enabled.
- `repeat` - boolean. If `true` then new task will be created when current task done.
- `unplanned` - boolean. If `true` service work is unplanned. For information only.
- `file_ids` - int array. One file will be specified in many service works. If one of the tasks will be deleted, then file will remain in others. File will be deleted only when the last task with it will be deleted.

## Task status

Task **status** may be one of:

- `created` – initial state of task.
- `notified` – one of conditions exceed notification limit.
- `expired` – one of conditions exceeded.
- `done` – user `set` task as "done".

## API actions

API path: `/vehicle/service_task`.

### batch\_create

Creates multiple service works.

#### parameters

name	description	type
vehicle_ids	List of vehicle IDs. Task will be created for every vehicle.	int array
task	Service work to create. <code>vehicle_id</code> field in these objects should not be specified.	JSON object

A `task` object is:

```
{
  "description": "Service work",
  "comment": "",
  "cost": 10050.0000,
  "conditions": {
    "mileage": {
      "limit": 100,
      "notification_interval": 10
    },
    "date": {
      "end": "2015-05-08 09:35:00",
      "notification_interval": 3
    },
    "engine_hours": {
      "limit": 100,
```

```

        "notification_interval": 10
    },
    "notifications": {
        "sms_phones": [
            "79221234567",
            "79227654321"
        ],
        "emails": [
            "email@domain.tld",
            "email@mail.com"
        ],
        "push_enabled": true
    },
    "repeat": false,
    "unplanned": false,
    "file_ids": [1, 2]
}

```

- `description` - string. Name of a service work. Max 255 characters.
- `comment` - string. Comment for a task. Max 255 characters.
- `cost` - float. Cost in the currency of the user. For information only.
- `conditions` - task end conditions. At least one of fields ("mileage" or "date" or "engine\_hours") must be passed.
  - `mileage` - optional object. Mileage condition.
    - `limit` - int. Task limit in kilometers.
    - `notification_interval` - int. Notify about task in specified number of kilometers.
  - `date` - optional date condition object.
    - `end` - `date/time`. Task end date.
    - `notification_interval` - int. Notify about task in specified number of days.
  - `engine_hours` - optional engine hours condition object.
    - `limit` - int. Task limit in hours.
    - `notification_interval` - int. Notify about task in specified number of hours.
- `notifications` - notifications object.
  - `sms_phones` - string array. Phones where sms notifications should be sent. In the international format without `+` sign.
  - `emails` - string array. Email addresses where sms notifications should be sent.
  - `push_enabled` - boolean. If `true` push notifications enabled.

- `repeat` - boolean. If `true` then new task will be created when current task done.
- `unplanned` - boolean. If `true` service work is unplanned. For information only.
- `file_ids` - int array. One file will be specified in many service works. If one of the tasks will be deleted, then file will remain in others. File will be deleted only when the last task with it will be deleted.

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/service_task/batch/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3",
    "vehicle_ids": [76801, 76449], "task": {"comment": "",
    "conditions": {"date": {"end": "2020-12-10 23:59:59",
    "notification_interval": 3}}, "cost": 100, "description":
    "service1", "file_ids": [], "notifications": {"sms_phones": [],
    "emails": [], "push_enabled": true}, "repeat": false, "unplanned":
    false}'
```

## response

```
{"success":true}
```

## errors

- [General](#) types only.

## create

Creates a new vehicle service work. For vehicles with associated tracker only.

## parameters

name	description	type
task	Service work to create.	JSON object

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/service_task/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "task": {"vehicle_id": 76801, "comment": "", "conditions": {"date": {"end": "2020-12-10 23:59:59", "notification_interval": 3}}, "cost": 100, "description": "service1", "file_ids": [], "notifications": {"sms_phones": [], "emails": [], "push_enabled": true}, "repeat": false, "unplanned": false}'
```

## response

```
{
  "success":true,
  "id": 33777
}
```

- `id` - int. An ID of created task.

## errors

- 201 - Not found in the database – vehicle or tracker not found.
- 214 - Requested operation or parameters not supported by the device – engine hours condition passed but tracker hasn't ignition sensor.

## delete

Deletes a vehicle service work.

### parameters

name	description	type
task_id	Optional. ID of service work.	int
task_ids	Optional. IDs of service works.	int array

Either **task\_id** or **task\_ids** should be specified.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/service_task/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "task_id": 33777}'
```

### HTTP GET

```
https://api.navixy.com/v2/vehicle/service_task/delete?hash=a6aa75587e5c59c32d347da438505fc3&task_id=33777
```

## response

```
{ "success": true }
```

## errors

- [General](#) types only.

## download

Downloads pdf report of service works.

## parameters

name	description	type
order_by	Sort option. Possible values listed below.	<a href="#">enum</a>
ascending	Optional. Default is <code>true</code> . Sort direction.	boolean
group_by	Optional. Group by option. Can be "vehicle" or "status".	<a href="#">enum</a>

- `order_by` possible values:
  - "vehicle" - order by `vehicle_id`.
  - "description" - order by `description`.
  - "status" - order by `status`.
  - "cost" - order by `cost`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/service_task/download' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "order_by": "vehicle", "group_by": "status"}'
```

### HTTP GET

```
https://api.navixy.com/v2/vehicle/service_task/download?hash=a6aa75587e5c59c32d347da438505fc3&order_by=vehicle&group_by=status
```

## response

Report file.

## errors

- [General](#) types only.

## list

List all service works of all user vehicles.

## parameters

name	description	type
return_prediction	Include legacy <b>prediction</b> field or not.	boolean

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/service_task/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "return_prediction": false}'
```

### HTTP GET

```
https://api.navixy.com/v2/vehicle/service_task/list?hash=a6aa75587e5c59c32d347da438505fc3&return_prediction=false
```

## response

```

{
  "success": true,
  "list": [
    {
      "id": 725,
      "vehicle_id": 222,
      "vehicle_label": "AGV",
      "status": "created",
      "prediction": {
        "end_date": "2015-05-03 09:35:00",
        "wear_percentage": 40
      },
      "description": "Service work",
      "cost": 10050.0,
      "completion": {
        "mileage": 31,
        "date": "2014-03-16 00:00:00",
        "engine_hours": 140
      },
      "completion_date": "2014-03-16 00:00:00",
      "conditions": {
        "mileage": {
          "limit": 100,
          "notification_interval": 10
        },
        "date": {
          "end": "2015-05-08 09:35:00",
          "notification_interval": 3
        },
        "engine_hours": {
          "limit": 100,
          "notification_interval": 10
        }
      },
      "current_position": {
        "mileage": 11,
        "date": "2012-03-06 15:55:03",
        "engine_hours": 100
      },
      "start": {
        "mileage": 1230,
        "date": "2015-05-01 17:46:44",
        "engine_hours": 50
      },
      "repeat": false,
      "unplanned": false,
      "file_ids": [2, 3]
    }
  ]
}

```

- list - array of vehicle objects described [here](#).

## errors

- 201 - Not found in the database – vehicle or tracker not found.



## read

Get service work info by its id.

### parameters

name	description	type
task_id	ID of service work.	int
return_prediction	Include legacy <b>prediction</b> field or not.	boolean

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/service_task/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "task_id": 37577, "return_prediction": false}'
```

#### HTTP GET

```
https://api.navixy.com/v2/vehicle/service_task/read?
hash=a6aa75587e5c59c32d347da438505fc3&task_id=37577&return_prediction=
```

### response

```
{
  "success": true,
  "value": {
    "id": 725,
    "vehicle_id": 222,
    "status": "created",
    "prediction": {
      "end_date": "2015-05-03 09:35:00",
      "wear_percentage": 40
    },
    "description": "Service work",
    "comment": "",
    "cost": 100500.0,
    "completion": {
      "mileage": 31,
      "date": "2014-03-16 00:00:00",
      "engine_hours": 140
    },
    "conditions": {
      "mileage": {
        "limit": 100,

```

```

        "notification_interval": 10
    },
    "date": {
        "end": "2015-05-08 09:35:00",
        "notification_interval": 3
    },
    "engine_hours": {
        "limit": 100,
        "notification_interval": 10
    }
},
"start": {
    "mileage": 1230,
    "date": "2015-05-01 17:46:44",
    "engine_hours": 50
},
"notifications": {
    "sms_phones": [
        "79221234567",
        "79227654321"
    ],
    "emails": [
        "email@domain.tld",
        "email@mail.com"
    ],
    "push_enabled": true
},
"completion_date" : "2014-03-16 00:00:00",
"repeat": false,
"unplanned": false,
"file_ids": [1, 2]
},
"files": [<file_object>]
}

```

All parameters described [here](#).

## errors

- 201 Not found in the database – does not exist one of tracker's counters which required to determine status.
- 204 Entity not found – when vehicle or service work not found.

## set\_status

Updates task status, and saved (on **done status**) current date and values of used (in condition) counters for "freeze" wearing percent.

## parameters

name	description	type
task_id	ID of service work.	int
status	A new task status. Only <code>done</code> status allowed for now.	<a href="#">enum</a>

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/service_task/set_status' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "task_id": 37577, "status": "done"}'
```

### HTTP GET

```
https://api.navixy.com/v2/vehicle/service_task/set_status?hash=a6aa75587e5c59c32d347da438505fc3&task_id=37577&status=done
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – does not exist one of tracker's counters which required to determine status.
- 204 - Entity not found – when vehicle or service work not found.

## update

Updates information fields and notification settings of vehicle service work.

## parameters

name	description	type
task	Service work to create.	JSON object

A [task object](#) described in a task create.

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/service_task/
update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "task":
{"vehicle_id": 76801, "comment": "", "conditions": {"date":
{"end": "2020-12-10 23:59:59", "notification_interval": 3}},
"cost": 100, "description": "service1", "file_ids": [],
"notifications": {"sms_phones": [], "emails": [], "push_enabled":
true}, "repeat": false, "unplanned": false}'
```

## response

```
{ "success": true }
```

## errors

- 204 - Entity not found – when vehicle or service work not found.
- 214 - Requested operation or parameters not supported by the device – engine hours condition passed but tracker hasn't ignition sensor.

Last update: August 1, 2023





# Vehicle status listing

## Deprecated

This API action deprecated and should not be used.

Contains vehicle status listing object and API calls to interact with it.

## Vehicle status listing object

```
{
  "id": 1,
  "order": 0,
  "label": "label123",
  "color": "FFFFFF"
}
```

- `id` - int. An ID of the status.
- `order` - int. Position of the status. Ignored when update because statuses already have position in an array.
- `label` - string. Status name (description).
- `color` - string. RGB-color.

## API actions

API path: `/vehicle/status/listing`.

### read

Gets all of user's vehicle statuses.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/status/listing/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

### HTTP GET

```
https://api.navixy.com/v2/vehicle/status/listing/read?hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [{
    "id": 1,
    "order": 0,
    "label": "label123",
    "color": "FFFFFF"
  }]
}
```

## errors

General types only.

## update

Updates user's vehicle statuses.

### parameters

name	description	type
statuses	List of vehicle_status_entry objects. If status ID is not null, then update, else create new vehicle status.	array of objects

Old vehicle statuses, which are not present in `statuses` array, will be deleted.



## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/vehicle/status/listing/
update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "statuses":
[{"id": 1, "order": 0, "label": "label123", "color": "FFFFFF"}]}'
```

## response

```
{ "success": true }
```

## errors

[General](#) types only.

Last update: December 26, 2022





# APN settings

API call to get APN settings by device's phone number.

## API actions

API base path: `/apn_settings`.

### read

Gets the APN name/user/password and mobile operator for registered device by phone number.

#### parameters

name	description	type	format
phone	string representing valid international phone number without <code>+</code> sign.	string	"1234567890"

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/apn_settings/read' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "phone":  
"1234567890"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/apn_settings/read?  
hash=a6aa75587e5c59c32d347da438505fc3&phone=1234567890
```

#### response

```
{  
  "success": true,  
  "value": {  
    "name": "internet",  
    "user": "",  
    "password": "",  
    "operator_name": "Deutsche Telekom"  }  
}
```

```
}  
}
```

### errors

- 201 – The phone number not found in the database.

Last update: December 26, 2022





# Delivery info

API calls to get delivery states and tasks by IDs.

## API actions

API base path: `/delivery`.

### read

Returns info sufficient for tracking certain task state, and the tracker assigned to it. Search conducted only among tasks and checkpoints, which have start date less than or equal now and have statuses: arrived, assigned or delayed. If multiple tasks or checkpoints found, then return first task, otherwise checkpoint.

#### session types:

In addition to standard user session, this call supports special *DELIVERY* session type.

#### parameters

name	description	type	format
external_id	An external ID of task.	int	259876

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/delivery/read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",  
    "external_id": 259876}'
```

##### HTTP GET

```
https://api.navixy.com/v2/delivery/read?  
hash=a6aa75587e5c59c32d347da438505fc3&external_id=259876
```

#### response

```
{  
  "success": true,
```



```

    "user_id": 3,
    "task" : {<task_object>},
    "tracker" : {<tracker_object>},
    "restrictions": {<restrictions_object>},
    "first_name": "John",
    "middle_name": "Micheel",
    "last_name": "Johnson",
    "vehicle_label": "Service car 002",
    "estimated_time": 1122
}

```

- `user_id` - master ID of the user to which the task belongs to.
- `task` - a task object, for more info see [/task](#) object structure.
- `tracker` - corresponding tracker object, for more info see [tracker/](#) object structure.
- `restrictions` - tariff restrictions object, for more info see [user/get\\_tariff\\_restrictions](#).
- `first_name` - string. The first name of employee assigned to the task, or null if missing.
- `middle_name` - string. The middle name of employee assigned to the task, or null if missing.
- `last_name` - string. The last name of employee assigned to the task, or null if missing.
- `vehicle_label` - string. A label of the vehicle assigned to the task, or null if missing.
- `estimated_time` - int. Estimated time of arrival in seconds, or null if unavailable.

## errors

- 201 – Not found in the database - when there is no task or checkpoint with specified conditions.

## list

External\_id can be repeated, so this request will return all matching delivery. Returns info sufficient for tracking certain task state, and the tracker assigned to it. Search conducted only among tasks and checkpoints, which have start date less than or equal now and have statuses: arrived, assigned or delayed.

## session types:

in addition to standard user session, this call supports special *DELIVERY* session type.

## parameters

name	description	type	format
external_id	An external ID of task.	int	259876

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/delivery/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",  
    "external_id": 259876}'
```

### HTTP GET

```
https://api.navixy.com/v2/delivery/list?  
hash=a6aa75587e5c59c32d347da438505fc3&external_id=259876
```

## response

```
{  
  "success": true,  
  "list": [{  
    "task" : {  
      "id": 111,  
      "user_id": 3,  
      "tracker_id": 22,  
      "location": {  
        "lat": 51.283546,  
        "lng": 7.301086,  
        "address": "Fichtenstrasse 11",  
        "radius": 150  
      },  
      "label": "Deliver parcels",  
      "description": "Quickly",  
      "creation_date": "2014-01-02 03:04:05",  
      "from": "2014-02-03 04:05:06",  
      "to": "2014-03-04 05:06:07",  
      "external_id": null,  
      "status": "assigned",  
      "status_change_date": "2014-01-02 03:04:05",  
      "max_delay" : 5,  
      "min_stay_duration": 0,  
      "arrival_date": "2014-01-02 03:04:05",  
      "stay_duration": 0,  
      "origin": "imported",  
      "tags": [1, 2],  
      "type": "task",  
    },  
    "tracker" : {  
      "id": 123456,
```

```

    "label": "tracker label",
    "clone": false,
    "group_id": 167,
    "avatar_file_name" : "file name",
    "source": {
      "id": 234567,
      "device_id": 99999999888888,
      "model": "telfmb920",
      "blocked": false,
      "tariff_id": 345678,
      "status_listing_id": null,
      "creation_date": "2011-09-21",
      "tariff_end_date": "2016-03-24",
      "phone" : "+71234567890"
    },
    "tag_bindings": [{
      "tag_id": 456789,
      "ordinal": 4
    }]
  },
  "first_name": "John",
  "middle_name": "Micheel",
  "last_name": "Johnson",
  "vehicle_label": "Service car 002",
  "estimated_time": 1122
}],
"user_id": 3,
"restrictions": {"restrictions_object":}
}

```

- `task` - a task object, for more info see [/task](#) object structure.
- `tracker` - corresponding tracker object, for more info see [tracker/](#) object structure.
- `first_name` - string. The first name of employee assigned to the task, or null if missing.
- `middle_name` - string. The middle name of employee assigned to the task, or null if missing.
- `last_name` - string. The last name of employee assigned to the task, or null if missing.
- `vehicle_label` - string. A label of the vehicle assigned to the task, or null if missing.
- `estimated_time` - int. Estimated time of arrival in seconds, or null if unavailable.
- `user_id` - master ID of the user to which the task belongs to.
- `restrictions` - tariff restrictions object, for more info see [user/get\\_tariff\\_restrictions](#).

## **errors**

- 201 – Not found in the database - when there is no task or checkpoint with specified conditions.

Last update: January 15, 2024





# Geocoder

API calls to search address and location using geocoder.

## Geocoder types

- google.
- yandex.
- progorod.
- osm.
- locationiq.

## API actions

API path: `/geocoder`.

### search\_address

Performs a forward geocoding. Returns a list of locations matching the given address. Items in the list sorted by relevance.

#### parameters

name	description	type	format
q	Address (or place) or coordinates to geocode.	string/ location	"750 Avenue E,San Francisco,CA 94130,USA"
geocoder	Optional. Geocoder type that will be preferably used for searching. Google geocoder is	enum	"google"

name	description	type	format
	always used for users with the premium GIS.		
bounds	Optional. JSON object. The bounding box, specified by coordinates of northwest and southeast corners. Geocoder will preferably return results from within these bounds. That is the parameter influences the priority of results, so if more relevant results exist outside of bounds, they may be included.	bounds_object	<pre>{"nw":{"lat": 37.9,"lng":-122.4},"se":{"lat": 37.8,"lng":-122.3}}</pre>
lang	Optional. Language in which results should be. ISO 639 <a href="#">language code</a> .	enum	"en_US"
with_details	Optional. If <code>true</code> then the response will contain details.	boolean	<code>true</code>



## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/geocoder/search_address' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "q": "750  
Avenue E, San Francisco, CA 94130, USA", "lang": "en", "geocoder":  
"google"}'
```

## response

```
{  
  "success": true,  
  "locations": [{  
    "lat": 37.825014712963565,  
    "lng": -122.37202062079945,  
    "address": "750 Avenue E, San Francisco",  
    "details": {  
      "country": "USA",  
      "province": "CA",  
      "locality": "San Francisco",  
      "street": "Avenue E",  
      "house": "750",  
      "postcode": "94130",  
      "bounds": {  
        "nw": {  
          "lat": 37.825064712964,  
          "lng": -122.3720706208  
        },  
        "se": {  
          "lat": 37.824964712964,  
          "lng": -122.3719706208  
        }  
      }  
    }  
  }  
}]  
}
```

- `lat` - double. Latitude.
- `lng` - double. Longitude.
- `address` - string. Address.
- `details` - details object.
  - `country` - optional string.
  - `province` - optional string.
  - `locality` - optional string.
  - `street` - optional string.
  - `house` - optional string.

- `postcode` - optional string.
- `bounds` - optional object, the bounding box which can fully contain the returned result.
  - `nw` - North West corner.
  - `se` - South East corner.

## search\_location

Search address by location using geocoder.

### parameters

name	description	type	format
location	Location coordinates (see: <a href="#">data types description section</a> section).	location	<code>{"lat": , "lng": }</code>
geocoder	Optional. Geocoder type that will be preferably used for searching. Google geocoder is always used for users with the premium GIS.	enum	"google"
lang	Optional. ISO 639 <a href="#">language code</a> .	enum	"en_US"
with_details	Optional. If <code>true</code> then the response will contain details.	boolean	<code>true</code>
goal	Helps to choose the target geocoder. Now supported <code>ui</code> , <code>ui_user_action</code> . Use <code>ui_user_action</code> for requests initiated by user, otherwise <code>ui</code> .	enum	"ui"

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/geocoder/search_location' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "location":  
{"lat": 37.825429, "lng": -122.371982}}'
```

## response

```
{
  "success": true,
  "value": "750 Avenue E, San Francisco, CA 94130, USA",
  "details": {
    "country": "USA",
    "province": "CA",
    "locality": "San Francisco",
    "street": "Avenue E",
    "house": "750",
    "postcode": "94130",
    "bounds": {
      "nw": {
        "lat": 37.825064712964,
        "lng": -122.3720706208
      },
      "se": {
        "lat": 37.824964712964,
        "lng": -122.3719706208
      }
    }
  }
}
```

- `value` - string. Address.
- `details` - optional details object.
  - `country` - optional string.
  - `province` - optional string.
  - `locality` - optional string.
  - `street` - optional string.
  - `house` - optional string.
  - `postcode` - optional string.
  - `bounds` - optional object, the bounding box which can fully contain the returned result.
    - `nw` - North West corner.
    - `se` - South East corner.

Last update: January 15, 2024





# Map layer

Contains map layer object structure and API calls to interact with it.

Map layer object structure:

```
{
  "id": 123456,
  "label": "test"
}
```

- `id` - int. Map layer entity ID.
- `label` - string. Map layer name.

## API actions

API path: `/map_layer`.

### read

Reads the body of the specified layer.

#### parameters

name	description	type	format
id	ID of the map layer.	int	123456

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/map_layer/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id": 123456}'
```

##### HTTP GET

```
https://api.navixy.com/v2/map_layer/read?
hash=a6aa75587e5c59c32d347da438505fc3&id=123456
```

## response

Layer body with content-type: `application/vnd.google-earth.kml+xml;`  
`charset=utf-8.`

## errors

- 201 - Not found in the database – if there is no map layer with such ID belonging to current user.

## list

Returns metadata for all map layers for the user.

## parameters

Only API key `hash`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/map_layer/list' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/map_layer/listd?  
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{  
  "success": true,  
  "list": [{  
    "id": 123456,  
    "label": "test"  
  }]  
}
```

## errors

No specific errors.

## upload

Uploads new map layer.

**MUST** be a POST multipart request (multipart/form-data), with one of the parts being a KML file upload (with the name "file").

#### parameters

name	description	type
label	Label for a new map layer.	string
file	A KML file upload containing map_layer data.	File upload
redirect_target	Optional. URL to redirect. If <b>redirect_target</b> passed return redirect to <code>&lt;redirect_target&gt;?response=&lt;urlencoded_response_json&gt;</code>	string

#### response

```
{
  "success": true,
  "id": 163
}
```

- `id` - int. ID of the created layer.

#### errors

- 233 - No data file – if file part is missing.
- 234 - Invalid data format – if file has wrong mime type.
- 242 - Validation error – if uploaded file is not valid KML.
- 268 - Over quota – if the user's quota for map layers exceeded.

## update

Updates metadata for the specified map layer.

#### parameters

name	description	type
layer	Map layer object described <a href="#">here</a>	JSON object



## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if there is no map layer with such ID belonging to current user.

## delete

Deletes specified layer.

## parameters

name	description	type	format
id	ID of the map layer.	int	123456

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/map_layer/delete' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id": 123456}'
```

### HTTP GET

```
https://api.navixy.com/v2/map_layer/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&id=123456
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if there is no map layer with such ID belonging to current user.

Last update: December 26, 2022





# Retranslator

Retranslator and retranslator protocol objects and CRUD actions for retranslators. They can be used to redirect the data that comes from a device to the platform to some third-party application specified by the user.

## Retranslator protocol object

```
{
  "id": 123456,
  "name": "protocol",
  "has_login": true,
  "has_password": false,
  "fake_device_id_pattern": "id_pattern",
  "required_login": true,
  "required_password": false
}
```

- `id` - int. Protocol ID.
- `name` - string. Protocol name.
- `has_login` - boolean. `true` if this protocol use login.
- `has_password` - boolean. `true` if this protocol use password.
- `fake_device_id_pattern` - optional string. Regex pattern for `fake_device_id` validation.
- `required_login` - boolean. `true` if for this protocol login required.
- `required_password` - boolean. `true` if for this protocol password required.

## Retranslator object

```
{
  "id": 1,
  "name": "Some server",
  "protocol_id": 123456,
  "address": "127.0.0.1",
  "port": 15000,
  "login": "login",
  "password": "password",
}
```

```
"enabled": true
}
```

- `id` - int. Retranslator ID.
- `name` - string. Zone label.
- `protocol_id` - int. Protocol ID.
- `address` - string. Network address, e.g. `127.0.0.1` or `localhost`.
- `port` - int. Port number.
- `login` - optional string.
- `password` - optional string.
- `enabled` - boolean. Status.

## API actions

API path: `/retranslator`.

### create

Creates new retranslator.

**required sub-user rights:** `admin` (available only to master users).

### parameters

name	description	type
retranslator	Retranslator object without <code>id</code> field.	JSON object

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/retranslator/create' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"retranslator": {"name": "Some server", "protocol_id": 123456,
"address": "127.0.0.1", "port": 15000, "login": "proto",
"password": "qewtyr", "enabled": true}}'
```

### response

```
{
  "success": true,
  "id": 123456
}
```

- `id` - int. ID of the created retranslator.

## errors

- 247 - Entity already exists - if retranslator with this address, port and login already exists.
- 7 - Invalid parameters - if retranslator have required fields (login or password), but was send empty.
- 268 - Over quota – if the user's quota for retranslators exceeded.

## delete

Deletes user's retranslator with specified `retranslator_id`.

**required sub-user rights:** `admin` (available only to master users).

## parameters

name	description	type	format
<code>retranslator_id</code>	ID of the retranslator that will be deleted.	int	123456

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/retranslator/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
    "retranslator_id": 123456}'
```

### HTTP GET

```
https://api.navixy.com/v2/retranslator/delete?
hash=a6aa75587e5c59c32d347da438505fc3&retranslator_id=123456
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database.

## list

Get all users' retranslators.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/retranslator/list' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/retranslator/list?  
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{  
  "success": true,  
  "list": [{  
    "id": 1,  
    "name": "Some server",  
    "protocol_id": 123456,  
    "address": "127.0.0.1",  
    "port": 15000,  
    "login": "login",  
    "password": "password",  
    "enabled": true  
  }]  
}
```

- `id` - int. Retranslator ID.
- `name` - string. Zone label.
- `protocol_id` - int. Protocol ID.
- `address` - string. Network address, e.g. `127.0.0.1` or `localhost`.
- `port` - int. Port number.
- `login` - optional string.
- `password` - optional string.
- `enabled` - boolean. Status.

## update

Updates retranslator parameters for the specified retranslator. Note that retranslator must exist, and must belong to the current user.

**required sub-user rights:** `admin` (available only to master users).

### parameters

name	description	type
retranslator	Retranslator object without <code>id</code> field.	JSON object

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/retranslator/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
    "retranslator": {"name": "Some server", "protocol_id": 123456,
    "address": "127.0.0.1", "port": 15000, "login": "proto",
    "password": "qewtyr", "enabled": true}}'
```

### response

```
{ "success": true }
```

### errors

- 201 - Not found in the database – if retranslator with the specified ID cannot be found or belongs to another user.
- 247 - Entity already exists – if retranslator with this address, port and login already exists.

## protocols list

Returns all available retranslator protocols.

### parameters

Only API key `hash`.



## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/retranslator/protocol/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/retranslator/protocol/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [{
    "id": 123456,
    "name": "protocol",
    "has_login": true,
    "has_password": false,
    "fake_device_id_pattern": "id_pattern",
    "required_login": true,
    "required_password": false
  }]
}
```

- `id` - int. Protocol ID.
- `name` - string. Protocol name.
- `has_login` - boolean. `true` if this protocol use login.
- `has_password` - boolean. `true` if this protocol use password.
- `fake_device_id_pattern` - optional string. Regex pattern for `fake_device_id` validation.
- `required_login` - boolean. `true` if for this protocol login required.
- `required_password` - boolean. `true` if for this protocol password required.

Last update: December 26, 2022





# BLE beacon data

Methods for obtaining collected BLE beacon data. BLE beacon data is data about radio tags (BLE beacons) visible to a tracker, e.g. iBeacon, Teltonika EYE Beacon\Sensor, Eddystone.

## BLE beacon data entry

```
{
  "tracker_id": 10181654,
  "hardware_id": "7cf9501df3d6924e423cabcd4c924ff",
  "rssi": -101,
  "get_time": "2023-04-17 17:14:42",
  "latitude": 50.3487321,
  "longitude": 7.58238,
  "ext_data": {
    "voltage": 3.075,
    "temperature": 24.0
  }
}
```

- `tracker_id` - int. An ID of the tracker (aka "object\_id").
- `hardware_id` - string. An ID of the beacon.
- `rssi` - int. RSSI stands for received signal strength indicator and represents the power of received signal on a device. According to it, you can understand how far away the beacon is from the tracker.
- `get_time` - [date/time](#). When this data received.
- `latitude` - float. Latitude.
- `longitude` - float. Longitude.
- `ext_data` - object. Additional beacon data.

## API actions

API path: `/beacon/data/read`.

### read

List of beacon data history between `from` date/time and `to` date/time sorted by **get\_time** field.

## parameters

name	description	type
from	Start date/time for searching.	string <a href="#">date/time</a>
to	End date/time for searching. Must be after "from" date.	string <a href="#">date/time</a>
trackers	Optional. Default: null. List of trackers.	int array
beacons	Optional. Default: null. List of beacons IDs. All IDs must not be empty and not more than 64 characters.	string array

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/beacon/data/read' \  
  -H 'Content-Type: application/json' \  
  -d '{  
    "hash": "59be129c1855e34ea9eb272b1e26ef1d",  
    "from": "2023-04-17 17:00:00",  
    "to": "2023-04-17 18:00:00",  
    "beacons": [ "ffffffffd86f4f75868d55aa831afa1f",  
    "7cf9501df3d6924e423cabcd4c924ff" ],  
    "trackers": [10181654]  
  }'
```

## response

```
{  
  "list": [  
    {  
      "tracker_id": 10181654,  
      "hardware_id": "ffffffffd86f4f75868d55aa831afa1f",  
      "rssi": -96,  
      "get_time": "2023-04-17 17:14:20",  
      "latitude": 50.3487301,  
      "longitude": 7.58207,  
      "ext_data": {  
        "minor": "0055",  
        "major": "3138"  
      }  
    },  
    {  
      "tracker_id": 10181654,  
      "hardware_id": "7cf9501df3d6924e423cabcd4c924ff",  
      "rssi": -101,  
      "get_time": "2023-04-17 17:14:20",  
      "latitude": 50.3487301,  
      "longitude": 7.58207,  
      "ext_data": {  
        "minor": "0055",  
        "major": "3138"  
      }  
    }  
  ]  
}
```

```

    "get_time": "2023-04-17 17:14:42",
    "latitude": 50.3487321,
    "longitude": 7.58238,
    "ext_data": {
      "voltage": 3.075,
      "temperature": 24.0
    }
  },
  "success": true
}

```

- `list` - list of zero or more `beacon_data_entry` objects which is described in [Beacon data entry](#).

API path: `/beacon/data/last_values`.

## last values

List of last BLE beacon data visible on the trackers.

### parameters

name	description	type
trackers	Optional. Default: null. List of trackers.	int array
skip_older_than_seconds	Optional. Default: 3600. Skip entries older than the specified number of seconds.	int

### example

#### cURL

```

curl -X POST 'https://api.navixy.com/v2/beacon/data/last_values' \
  -H 'Content-Type: application/json' \
  -d '{
    "hash": "59be129c1855e34ea9eb272b1e26ef1d",
    "trackers": [10181654],
    "skip_older_than_seconds": 3600
  }'

```

### response

```

{
  "list": [
    {
      "tracker_id": 10181654,

```

```

    "hardware_id": "7cf9501df3d6924e423cabcd4c924ff",
    "rssi": -101,
    "get_time": "2023-04-17 17:14:42",
    "latitude": 50.3487321,
    "longitude": 7.58238,
    "ext_data": {
      "voltage": 3.075,
      "temperature": 24.0
    }
  },
  "success": true
}

```

- `list` - list of zero or more `beacon_data_entry` objects which is described in [Beacon data entry](#).

Last update: April 25, 2023







# Getting route

API call for getting the route to destination point.

## API actions

API path: `/route`.

### get

Gets route points via specified route provider.

#### parameters

name	description	type
start	Location JSON object. Start of route.	JSON object
end	Location JSON object. End of route.	JSON object
waypoints	Optional. List of transitional points. <code>[{locationA},{locationN}]</code> .	array of JSON objects
point_limit	Optional. If specified, the returned route will be simplified to contain this number of points (or less). Min=2.	int
provider_type	Optional. If not specified, the default user provider is used. One of "progorod", or "google", "osrm".	enum

- `location` object described in [data types description section](#).

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/route/get' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "start": {"lat": 34.178868, "lng": -118.599672}, "end": {"lat": 31.738386, "lng": -106.453854}}'
```

## response

```
{
  "success": true,
  "distance": 1340584,
  "time": 43500,
  "list": [{"lat": 34.178868, "lng": -118.599672}, {"lat": 31.738386, "lng": -106.453854}],
  "key_points": [{
    "id": 123,
    "lat": 35.365948,
    "lng": -108.112104
  }]
}
```

- `distance` - int. Length in meters.
- `time` - int. Duration in seconds.
- `list` - list of route points. Location objects.
- `key_points` - list of points corresponding to `start` point, `waypoints` and `end` point (in that sequence).
  - `id` - int. index in points `list`.
  - `lat` - float. Latitude.
  - `lng` - float. Longitude.

## errors

- 215 - External service error.
- 218 - Malformed external service parameters.
- 236 - Feature unavailable due to tariff restrictions – if there is at least one tracker without "routing" tariff feature.

Last update: January 15, 2024





# Getting the route with Google

API call for getting the route to destination point using [Google Directions API](#).

## API actions

API path: `/route/google`.

### get

Gets route points using Google Directions API.

#### parameters

name	description	type
start	Location JSON object. Start of route.	JSON object
end	Location JSON object. End of route.	JSON object
waypoints	Optional. List of transitional points. [{locationA}, {locationN}].	array of JSON objects
point_limit	Optional. If specified, the returned route will be simplified to contain this number of points (or less). Min=2.	int

Where **location** described in [data types description section](#).

#### example

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/route/google/get' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "start": {"lat": 34.178868, "lng": -118.599672}, "end": {"lat": 31.738386, "lng": -106.453854}}'
```

## response

```
{
  "success": true,
  "distance": 1340584,
  "time": 43500,
  "list": [{"lat": 34.178868, "lng": -118.599672}, {"lat": 31.738386, "lng": -106.453854}],
  "key_points": [{
    "id": 123,
    "lat": 35.365948,
    "lng": -108.112104,
    "distance": 1052511,
    "time": 37800
  }]
}
```

- `distance` - int. Length in meters.
- `time` - int. Duration in seconds.
- `list` - list of route points. Location objects.
- `key_points` - list of points corresponding to `start` point, `waypoints` and `end` point (in that sequence).
  - `id` - int. index in points `list`.
  - `lat` - float. Latitude.
  - `lng` - float. Longitude.
  - `distance` - int. Length of full path from start in meters (0 for start point).
  - `time` - int. Duration of full path from start in seconds (0 for start point).

## errors

215 - External service error.

```
{
  "success": false,
  "status": {
    "code": 215,
    "description": "External service error"
  },
  "errors": ["OVER_QUERY_LIMIT"]
}
```

- `errors` - `enum` array. Status.
  - `OVER_QUERY_LIMIT` - indicates the service has received too many requests from your application within the allowed time period.

- `REQUEST_DENIED` – indicates that the service denied use of the directions service by your application.
- `UNKNOWN_ERROR` – indicates directions request could not be processed due to a server error. The request may succeed if you try again.

218 - Malformed external service parameters.

```
{
  "success": false,
  "status": {
    "code": 218,
    "description": "Malformed external service parameters"
  },
  "errors": ["NOT_FOUND"]
}
```

- `errors` - `enum` array. Status.
  - `NOT_FOUND` – indicates at least one of the locations specified in the request's origin, destination, or waypoints could not be geocoded.
  - `ZERO_RESULTS` – indicates no route could be found between the origin and destination.
  - `MAX_WAYPOINTS_EXCEEDED` – indicates that too many waypoints provided in the request. The maximum allowed waypoints is 8, plus the origin, and destination. Google Maps API for Business customers may contain requests with up to 23 waypoints.
  - `INVALID_REQUEST` – indicates that the provided request was invalid. Common causes of this status include an invalid parameter or parameter value.

236 - Feature unavailable due to tariff restrictions – if there is at least one tracker without "routing" tariff feature.

Last update: January 15, 2024







# Getting route with OSRM

API call for getting the route to destination point using [OSRM API](#).

## API actions

API path: `/route/osrm`.

### get

Gets route points via OSRM API.

#### parameters

name	description	type
start	Location JSON object. Start of route.	JSON object
end	Location JSON object. End of route.	JSON object
waypoints	Optional. List of transitional points. [{locationA}, {locationN}].	array of JSON objects
point_limit	Optional. If specified, the returned route will be simplified to contain this number of points (or less). Min=2.	int

Where **location** described in [data types description section](#).

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/route/osrm/get' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "start": {34.178868, "lng": -118.599672}, "end": {35.365948, "lng": -108.112104}}'
```

## response

```
{
  "success": true,
  "distance": 1340584,
  "time": 43500,
  "list": [{"lat": 34.178868, "lng": -118.599672}, {"lat": 31.738386, "lng": -106.453854}],
  "key_points": [{
    "id": 123,
    "lat": 35.365948,
    "lng": -108.112104
  }]
}
```

- `distance` - int. Length in meters.
- `time` - int. Duration in seconds.
- `list` - list of route points. Location objects.
- `key_points` - list of points corresponding to `start` point, `waypoints` and `end` point (in that sequence).
  - `id` - int. index in points `list`.
  - `lat` - float. Latitude.
  - `lng` - float. Longitude.

## errors

- 215 - External service error.
- 218 - Malformed external service parameters.

```
{
  "success": false,
  "status": {
    "code": 218,
    "description": "Malformed external service parameters"
  },
  "errors": [
    {
      "status": "NOT_FOUND",
      "status_code": 207,
      "message": "Cannot find route between points"
    }
  ]
}
```

```
]
}
```

- `status` - [enum](#).
  - `NOT_FOUND` – indicates at least one of the locations specified in the request's origin, destination, or waypoints could not be geocoded, or OSRM cannot find route.
  - `UNKNOWN_ERROR` – unexpected OSRM error code.
- `status_code` - int. OSRM status code (don't rely on it).
- `message` - string. OSRM error message (don't rely on it).
- 236 - Feature unavailable due to tariff restrictions – if there is at least one tracker without "routing" tariff feature.

Last update: January 15, 2024





# Getting route with Progorod

API call for getting the route to destination point using [Progorod router](#).

## API actions

API path: `/route/progorod`.

### get

Gets route points using Progorod router.

#### parameters

name	description	type
start	Location JSON object. Start of route.	JSON object
end	Location JSON object. End of route.	JSON object
waypoints	Optional. List of transitional points. <code>[{locationA}, {locationN}]</code> .	array of JSON objects
point_limit	Optional. If specified, the returned route will be simplified to contain this number of points (or less). Min=2.	int
minsize	Optional. Default=5. Smoothing parameter in conventional meters. Not recommended to set it less than distance between two neighbouring pixels on current zoom.	double
use_traffic	Optional. Default= <code>false</code> If it is <code>false</code> then use <code>mode=optimal</code> and use traffic=0, else <code>mode=comfort</code> and use traffic=1.	boolean



Where **location** described in [data types description section](#). Order of waypoints may be changed.

#### response

```
{
  "success": true,
  "distance": 1340584,
  "time": 43500,
  "list": [{"lat": 34.178868, "lng": -118.599672}, {"lat": 31.738386, "lng": -106.453854}],
  "key_points": [{
    "id": 123,
    "lat": 35.365948,
    "lng": -108.112104
  }]
}
```

- `distance` - int. Length in meters.
- `time` - int. Duration in seconds.
- `list` - list of route points. Location objects.
- `key_points` - list of points corresponding to `start` point, `waypoints` and `end` point (in that sequence).
  - `id` - int. index in points `list`.
  - `lat` - float. Latitude.
  - `lng` - float. Longitude.

#### errors

- 215 - External service error.
- 218 - Malformed external service parameters – Contains info about error:

```
{
  "success": false,
  "status": {
    "code": 218,
    "description": "Malformed external service parameters"
  },
  "errors": [{
    "type": "malformed",
    "point": "start",
    "index": 3
  }]
}
```

- `type` - [enum](#). Type of error. One of: "not\_set", "malformed" and "isolated".
- `point` - [enum](#). Error point. One of: "start", "end", "waypoint" and "all".

- `index` - int. Passed only for a waypoint. Index of bad point in waypoints array.

Last update: January 15, 2024





# Working status

Contains status object and API calls to interact with them. Working statuses used to track current activity for employees (in fact, of tracking devices owned by employees). The simplest example is "busy" | "not busy". This is a status listing consisting of two elements. Different trackers can be assigned different status lists.

Find details on working status usage in our [guides](#).

## Status object structure

```
{
  "id": 5,
  "label": "Busy",
  "color": "E57373"
}
```

- `id` - int. A unique identifier of the working status. Read-only.
- `label` - string. Human-readable label for the working status.
- `color` - string. Hex-representation of RGB color used to display this working status.

## API actions

API base path: `/status/`.

### create

Creates new possible working status for the specified working status list.

**required sub-user rights:** `tracker_update`.

### parameters

name	description	type
listing_id	ID of the list for this working status to attach to.	int
status	Status object without ID field.	JSON object

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "listing_id": 12345, "status": {"label": "Busy", "color": "E57373"}}'
```

## response

```
{
  "success": true,
  "id": 111
}
```

- `id` - int. ID of the created working status.

## errors

- 201 - Not found in the database – if listing with the specified ID does not exist.
- 236 - Feature unavailable due to tariff restrictions – if there are no trackers with "statuses" tariff feature available.
- 268 - Over quota – if the user's quota for working statuses exceeded.

## delete

Deletes working status entry.

**required sub-user rights:** `tracker_update`.

## parameters

name	description	type
status_id	ID of the working status belonging to authorized user.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "status_id": 123}'
```

### HTTP GET

```
https://api.navixy.com/v2/status/delete?
hash=a6aa75587e5c59c32d347da438505fc3&status_id=123
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if working status with the specified ID does not exist.
- 236 - Feature unavailable due to tariff restrictions – if there are no trackers with "statuses" tariff feature available.

## list

Gets working statuses belonging to the specified status list.

## parameters

name	description	type
listing_id	ID of the list for this working status to attach to.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "listing_id": 12345}'
```

### HTTP GET

```
https://api.navixy.com/v2/status/list?
hash=a6aa75587e5c59c32d347da438505fc3&listing_id=12345
```

## response

```
{
  "success": true,
  "list": [{
    "id": 5,
    "label": "Busy",
    "color": "E57373"
  }, {
    "id": 6,
    "label": "Free",
    "color": "A27373"
  }]
}
```

- `list` - ordered array of objects.

## errors

- 236 - Feature unavailable due to tariff restrictions – if there are no trackers with "statuses" tariff feature available.

## update

Updates working status properties.

**required sub-user rights:** `tracker_update`.

## parameters

name	description	type
status	Status object with ID field.	JSON object

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "status": {"id": 5, "label": "Busy", "color": "E57373"}}'
```

## response

```
{ "success": true }
```



**errors**

- 201 - Not found in the database – if working status with the specified ID does not exist.
- 236 - Feature unavailable due to tariff restrictions – if there are no trackers with "statuses" tariff feature available.

Last update: August 1, 2023





# Tracker's working status

This resource contains methods to read and assign working status of a particular tracker.

## API actions

API base path: `/status/tracker/`.

### assign

Assign a working status to the tracker.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
new_status_id	ID of the working status. Must belong to status list assigned to this tracker.	int	5

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/tracker/assign' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "new_status_id": 5}'
```

##### HTTP GET

```
https://api.navixy.com/v2/status/tracker/assign?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456&new_status_id=
```

#### response

```
{
  "success": true,
  "last_change": {
```

```

    "id": 11,
    "old_status_id": null,
    "new_status_id": 2,
    "location": {
        "lat": 11.0,
        "lng": 22.0,
        "address": "Jones st, 4"
    },
    "changed": "2015-11-22 02:02:02",
    "origin": "supervisor"
}

```

- `last_change` - object describing last change of the status. May be null.
  - `old_status_id` - int. Previous status ID. May be null.
  - `new_status_id` - int. Current status ID. May be null.
  - `location` - object. Location and address at which status change occurred.
    - `lat` - int. Latitude.
    - `lng` - int. Longitude.
    - `address` - string. Address of last change.
  - `changed` - [date/time](#). Change date and time.
  - `origin` - [enum](#). Origin – who changed the status ("employee" or "supervisor").

## errors

- 13 - Operation not permitted – if status list does not allow for a supervisor to change status.
- 201 - Not found in the database – if there is no tracker with such ID belonging to authorized user.
- 204 - Entity not found – if there is no status list assigned to this tracker containing with such ID.
- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions or some other reason.
- 219 - Not allowed for clones of the device – if specified tracker is a clone.
- 236 - Feature unavailable due to tariff restrictions – if there are no trackers with "statuses" tariff feature available.
- 263 - No change needed, old and new values are the same – if new status is equal to current status of tracker.

## list

Gets current assigned statuses for the specified trackers.

### parameters

name	description	type	format
trackers	List of the tracker's IDs belonging to authorized user.	int array	[ 123456, 234567 ]

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/tracker/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "trackers":  
  [123456,234567]}'
```

### response

```
{  
  "success": true,  
  "value": {  
    "5344": {  
      "current_status": {  
        "id": 66,  
        "label": "Busy",  
        "color": "FFC107"  
      },  
      "last_change": {  
        "id": 441,  
        "old_status_id": 65,  
        "new_status_id": 66,  
        "location": {  
          "lat": 35.365948,  
          "lng": -108.112104,  
          "address": "Navajo Nation Off-Reservation  
Trust Land, Chinle, NM, USA"  
        },  
        "changed": "2017-05-02 07:40:39",  
        "origin": "supervisor"  
      }  
    },  
    "15595": {  
      "current_status": null,  
      "last_change": {  
        "id": 123,  
        "old_status_id": 67,  
        "new_status_id": null,  
        "location": {
```

```

        "lat": 34.178868,
        "lng": -118.599672,
        "address": ""
      },
      "changed": "2016-03-14 04:58:32",
      "origin": "employee"
    }
  }
}

```

- `value` - Map with a tracker's IDs as keys.
  - `current_status` - Status object showing current status of tracker. May be null.
  - `last_change` - Object describing last change of the status. May be null.
  - `old_status_id` - int. Previous status ID. May be null.
  - `new_status_id` - int. Current status ID. May be null.
  - `location` - Location and address at which status change occurred.
  - `changed` - [date/time](#). Date and time of change.
  - `origin` - [enum](#). Origin – who changed the status ("employee" or "supervisor").

## errors

- 217 - List contains nonexistent entities - if one or more of tracker IDs belong to nonexistent tracker (or to a tracker belonging to different user).
- 221 - Device limit exceeded – if device limit set for the user's dealer has been exceeded.
- 236 - Feature unavailable due to tariff restrictions – if there are no trackers with "statuses" tariff feature available.

## read

Gets current assigned working status of the tracker.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/tracker/read' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456}'
```

### HTTP GET

```
https://api.navixy.com/v2/status/tracker/read?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456
```

## response

```
{
  "success": true,
  "current_status": {
    "id": 2,
    "label": "On duty",
    "color": "FFFF99"
  },
  "last_change": {
    "id": 11,
    "old_status_id": null,
    "new_status_id": 2,
    "location": {
      "lat": 11.0,
      "lng": 22.0,
      "address": "Jones st, 4"
    },
    "changed": "2015-11-22 02:02:02",
    "origin": "supervisor"
  }
}
```

- `current_status` - status object showing current status of tracker. May be null.
- `last_change` - object describing last change of the status. May be null.
  - `old_status_id` - int. Previous status ID. May be null.
  - `new_status_id` - int. Current status ID. May be null.
  - `location` - Location and address at which status change occurred.
  - `changed` - [date/time](#). Date and time of change.
  - `origin` - [enum](#). Origin – who changed the status ("employee" or "supervisor").

## errors

- 201 - Not found in the database – if there is no tracker with such ID belonging to authorized user.



- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions, or some other reason.
- 219 - Not allowed for clones of the device – if specified tracker is a clone.
- 236 - Feature unavailable due to tariff restrictions – if there are no trackers with "statuses" tariff feature available.

Last update: April 8, 2024





# Working status list

Contains status listing object and API calls to interact with status listings. Status listings are lists of possible statuses that can be assigned to trackers.

## Status listing object structure

```
{
  "id": 1,
  "label": "Taxi driver statuses",
  "employee_controlled": true,
  "supervisor_controlled": false,
  "entries": [ 5, 2, 1, 4, 6 ]
}
```

- `id` - int. A unique identifier of this working status list. Read-only.
- `label` - string. Human-readable label for the working status list.
- `employee_controlled` - boolean. If `true` employees can change their own working status, e.g. using mobile tracking app.
- `supervisor_controlled` - boolean. If `true` supervisors can change working status, e.g. using mobile monitoring app.
- `entries` - int array. List of IDs of working statuses which belong to this list. Order matters, and is preserved.

## API actions

API base path: `/status/listing/`.

### create

Creates new empty working status list.

**required sub-user rights:** `tracker_update`.

## parameters

name	description	type
listing	<a href="#">status_listing</a> object without "id" and "entries" fields.	JSON object

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/listing/create' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "listing": \  
{"label": "Taxi driver statuses", "employee_controlled": false, \  
"supervisor_controlled": true}'
```

## response

```
{  
  "success": true,  
  "id": 111  
}
```

- `id` - int. ID of the created working status list.

## errors

- 236 - Feature unavailable due to tariff restrictions – if there are no trackers with "statuses" tariff feature available.
- 268 - Over quota – if the user's quota for working status lists exceeded.

## delete

Deletes working status list.

**required sub-user rights:** `tracker_update`.

## parameters

name	description	type
listing_id	ID of the working status list for this status to attach to.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/listing/delete' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "listing_id": 12345}'
```

### HTTP GET

```
https://api.navixy.com/v2/status/listing/delete?
hash=a6aa75587e5c59c32d347da438505fc3&listing_id=12345
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if working status list with the specified ID does not exist.
- 236 - Feature unavailable due to tariff restrictions – if there are no trackers with "statuses" tariff feature available.

## list

Gets working status lists belonging to authorized user.

## parameters

Only API key `hash`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/listing/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/status/listing/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [{
```

```

    "id": 1,
    "label": "Taxi driver statuses",
    "employee_controlled": true,
    "supervisor_controlled": false,
    "entries": [ 5, 2, 1, 4, 6]
  }]
}

```

- `list` - ordered array of `status_listing` objects.

## errors

- 236 - Feature unavailable due to tariff restrictions – if there are no trackers with "statuses" tariff feature available.

## update

Updates working status list properties.

**required sub-user rights:** `tracker_update`.

`entries` field allows changing order of statuses attached to this working status list.

## parameters

name	description	type
listing	<code>status_listing</code> object with "id" and "entries" fields.	JSON object

## examples

### cURL

```

curl -X POST 'https://api.navixy.com/v2/status/listing/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "listing":
{"id": 12345, "label": "Taxi driver statuses",
"employee_controlled": false, "supervisor_controlled": true,
"entries": [ 5, 2, 1, 4, 6]}'

```

## response

```

{ "success": true }

```

## **errors**

- 201 - Not found in the database – if working status list with the specified ID does not exist.
- 236 - Feature unavailable due to tariff restrictions – if there are no trackers with "statuses" tariff feature available.
- 262 - Entries list is missing some entries or contains nonexistent entries – if entries does not contain full set of status IDs associated with this working status list, or if it contains nonexistent status IDs.

Last update: December 26, 2022







# Tracker's working status list

Contains api call which link together trackers and working status lists.

## API actions

API base path: `/status/listing/tracker`.

### assign

Assigns a working status list (or remove assignment) to the tracker.

**required sub-user rights:** `tracker_update`.

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
listing_id	ID of the working status list. Omit this parameter completely, if you want remove the assignment.	int	12345

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/status/listing/tracker/assign' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "listing_id": 12345}'
```

#### HTTP GET

```
https://api.navixy.com/v2/status/listing/tracker/assign?hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456&listing_id=123
```

### response

```
{ "success": true }
```

## **errors**

- 201 - Not found in the database – if there is no tracker with such ID belonging to authorized user.
- 204 - Entity not found – if there is no working status list with such ID.
- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions or some other reason.
- 219 - Not allowed for clones of the device – if specified tracker is a clone.
- 236 - Feature unavailable due to tariff restrictions – if there are no trackers with "statuses" tariff feature available.

Last update: December 26, 2022





# Track

This section includes API calls that allow you to interact with tracks and retrieve track points.

Learn more about the track API by following our [instructions](#).

## API actions

API path: `/track`.

### download

This method allows you to download track points as a KML/KMZ file which can be used in other apps to show tracks.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). The tracker must be associated with the user whose hash is being used for the request, and not tariff-blocked.	int	123456
from	The start date/time for your KML file's track points. The file begins with the next point after this time.	<a href="#">date/</a> <a href="#">time</a>	"2020-09-23 03:24:00"
to	An end date/time for your KML file's track points. The file concludes with the last point before this time. Ensure this date is later than the "from" date.	<a href="#">date/</a> <a href="#">time</a>	"2020-09-23 06:24:00"

name	description	type	format
track_ids	Optional. If specified, the file will only contain points from selected tracks. If not, it includes all valid points between the "from" and "to" times.	int array	[123456, 234567]
include_gsm_lbs	Optional. If set to <code>false</code> without specified track_ids, GSM LBS points will be excluded. Default is <code>true</code> .	boolean	<code>true</code>
simplify	Optional. If set to <code>true</code> , tracks in the returned file will be simplified with fewer points, optimized for uploading to other apps. Default is <code>true</code> .	boolean	<code>true</code>
point_limit	Optional. If it is specified and <code>simplify=true</code> , the returned tracks in a file will be reduced to contain that specified number of points. The minimum value is 2, and the maximum is 3000. If it is not specified, the server's default settings for simplifying tracks will be applied. This is not a strict limit; the returned file can potentially contain more points than specified.	int	300
filter	Optional. If this is set to <code>true</code> , the returned tracks in a file will be filtered. This is currently only applicable to LBS tracks.	boolean	<code>true</code>
format	File format can be "kml" or "kmz". Default is "kml".	enum	"kml"
split		boolean	<code>false</code>



name	description	type	format
	If set to <code>true</code> , tracks in the file will be split by stops into folders with start/end markers. Default is <code>false</code> .		

## example

Let's consider an example, where we're looking for a KML file for a tracker with ID 1683258 that doesn't break down by stops. The data should cover trips starting from 3:24 AM to 6:24 AM on November 19, 2023, according to the user's local time.

### cURL

```
curl -X POST 'https://api.navixy.com/v2/track/download' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 1683258, "from": "2023-11-19 03:24:00", "to": "2023-11-19 06:24:00", "format": "kml", "split": false}'
```

## response

In case the available storage period is not exceeded, you will get the file.

!!!+ example "KML file example with two points"

```
```.xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<kml xmlns="http://www.opengis.net/kml/2.2" xmlns:gx="http://www.google.com/kml/ext/2.2" xmlns:atom="http://www.w3.org/2005/Atom" xmlns:xal="urn:oasis:names:tc:ciq:xdschema:xAL:2.0">
  <Document>
    <name>track-John (Scania) 2023-11-19 03:24:00</name>
    <Placemark>
      <name>point #1</name>
      <visibility>1</visibility>
      <description>2023-11-19 03:24:03</description>
      <TimeStamp>
        <when>2023-11-19T03:24:03.000-06:00</when>
      </TimeStamp>
      <ExtendedData>
        <Data name="speed">
          <value>37</value>
        </Data>
        <Data name="heading">
          <value>27</value>
        </Data>
      </ExtendedData>
      <Point>
        <coordinates>-78.768105,43.1172216</coordinates>
      </Point>
    </Placemark>
  </Document>
</kml>
```

```

    </Placemark>
    <Placemark>
      <name>point #2</name>
      <visibility>1</visibility>
      <description>2023-11-19 03:27:11</description>
      <TimeStamp>
        <when>2023-11-19T03:27:11.000-06:00</when>
      </TimeStamp>
      <ExtendedData>
        <Data name="speed">
          <value>57</value>
        </Data>
        <Data name="heading">
          <value>13</value>
        </Data>
      </ExtendedData>
      <Point>
        <coordinates>-78.7549233,43.1356483</coordinates>
      </Point>
    </Placemark>
  </Document>
</kml>
...

```

For example, if the device's plan has maximum available storage period 3 months (default value) and we request data from 6 months, the response will contain JSON with the next information:

```

{
  "list": [],
  "limit_exceeded": true,
  "success": true
}

```

## errors

- 201 - Not found in database – the tracker ID in your request may not match any trackers linked to the user account with this session hash. Ensure the correct tracker\_id and hash of an appropriate user are used.
- 208 - Device blocked – if a tracker exists under this user account but is currently inactive due to tariff plan restrictions or any other reason.
- 211 - Requested time span is too big – If the interval between the "from" and "to" dates is too large, it may exceed the maximum value defined in the API configuration.

## list

This method retrieves a list of tracks for a given tracker within a specified time period.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). The tracker must be associated with the user whose hash is being used for the request, and not tariff-blocked.	int	123456
from	The start date/time for trips. The response begins with the next trip point after this time.	date/ time	"2020-09-23 03:24:00"
to	An end date/time for trips. The response concludes with the last point before this time. Ensure this date is later than the "from" date.	date/ time	"2020-09-23 06:24:00"
filter	Optional. Default is <code>true</code> . If set to <code>true</code> , any tracks that are deemed too short, based on their length and number of points, will be excluded from the final list.	boolean	<code>true</code>
split	Optional. Default is <code>true</code> . If set to <code>false</code> , all the tracks will be combined into one single track within the period.	boolean	<code>true</code>
include_gsm_lbs	Optional. Default is <code>true</code> . If set to <code>false</code> , GSM LBS points will be excluded.	boolean	<code>true</code>
cluster_single_reports		boolean	<code>false</code>

name	description	type	format
	Optional. Default is <code>false</code> . If set to <code>true</code> , trips consisting of a single point will be grouped together based on their coordinates.		
count_events	Optional. Default is <code>false</code> . If set to <code>true</code> , the system will return the count of events that occurred during each track that isn't a single point.	boolean	<code>false</code>
omit_addresses	Optional. Default is <code>false</code> . If set to <code>true</code> , address parameters will be empty.	boolean	<code>false</code>
with_points	Optional. Default is <code>false</code> . If set to <code>true</code> , track point lists will be included.	boolean	<code>false</code>
point_limit	Optional. If specified, the returned data will be reduced to contain that specified number of points. The minimum value is 2, and the maximum is 3000. If it is not specified, the server's default settings for simplifying tracks will be applied. This is not a strict limit; the returned data can potentially contain more points than specified.	int	<code>300</code>

## example

For example, if we need to retrieve all trips for tracker 1683258 in November, without applying smart filter, including LBS points recorded, without clustering, separated by parkings, while also counting the number of events that occur during each trip, we only need to specify the "filter" and "count\_events" from optional parameters. This is because the other optional settings will provide us with necessary info by default.

### cURL

```
curl -X POST 'https://api.navixy.com/v2/track/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
1683258, "from": "2023-11-01 03:24:00", "to": "2023-11-30
06:24:00", "filter": false, "count_events": true}'
```

## response

```
{
  "success": true,
  "limit_exceeded": false,
  "list": [track_info]
}
```

- `limit_exceeded` - boolean. It will be `true` if the requested time period surpasses the limit set in the tracker's tariff. For instance, if the device's plan has a maximum storage period of three months (the default value), and we request trips for six months.
- `list` - an array of JSON objects containing track information. It consists of zero or more JSON objects. Zero objects indicates that there were no trips based on the track options or the device didn't supply any points to the platform.

where `track_info` is either `regular`, `single_report`, `merged` or `cluster`:

`regular` object:

```
{
  "id": 123456,
  "start_date": "2023-11-23 03:39:44",
  "start_address": "1255 6th Ave, New York, NY 10020, USA",
  "max_speed": 62,
  "end_date": "2023-11-23 06:39:44",
  "end_address": "888 5th Ave, New York, NY 10021, USA",
  "length": 5.5,
  "points": 327,
  "avg_speed": 49,
  "event_count": 3,
  "norm_fuel_consumed": 1.07,
  "type": "regular",
  "gsm_lbs": false,
```

```

    "points_list": [point_info]
}

```

- `id` - int. Track id.
- `start_date` - [date/time](#). Track start date, in user's timezone e.g. "2011-06-18 03:39:44".
- `start_address` - string. Track start address.
- `max_speed` - int. Maximum speed registered during track in km/h, e.g. 96.
- `end_date` - [date/time](#). Track end date, in user's timezone e.g. "2011-06-18 05:18:36".
- `end_address` - string. Track end address.
- `length` - float. Track length in kilometers, e.g. 85.5.
- `points` - int. Total number of points in a track, e.g. 724.
- `avg_speed` - int. Average speed in km/h, e.g. 70.
- `event_count` - int. It represents the number of events recorded during this track. This field will not be present if "count\_events" is set to `false`.
- `norm_fuel_consumed` - float. It is representing the amount of fuel consumed during the track, measured in litres. This field will not be present if there's no [vehicle\\_object](#) linked to the tracker or if "normAvgFuelConsumption" is not defined for the linked vehicle object.
- `type` - [enum](#) with possible values: `regular`, `single_report`, `merged`, `cluster`. It's used to differentiate this regular track type from other types.
- `gsm_lbs` - optional boolean. GSM LBS point flag.
- `points_list` - array of JSON objects. A list of [point info](#).

`single_report` object is returned when the device operates in "interval" mode or only one point per track is provided (for example, an M7 tracker operating in interval mode):

```

{
  "id": 123456,
  "start_date": "2023-11-24 03:39:44",
  "start_address": "1255 6th Ave, New York, NY 10020, USA",
  "avg_speed": 34,
  "gsm_lbs": false,
  "type": "single_report",
  "precision": 10,
  "points_list": [point_info]
}

```

- `id` - int. Track id.

- `start_date` - [date/time](#). It represents the date and time when the tracker registered the point, adjusted to the user's timezone, for example, "2011-06-18 03:39:44".
- `start_address` - string. Point address.
- `avg_speed` - int. Average speed in km/h, e.g. 70.
- `gsm_lbs` - optional boolean. GSM LBS point flag.
- `type` - [enum](#): `regular`, `single_report`, `merged`, `cluster`. Used to distinguish this track type (`single_report`) from the others.
- `precision` - optional int. It signifies the precision of the location in meters. Its presence relies on the device model.
- `points_list` - array of JSON objects. A list of [point info](#).

`merged` object. Only returned if "split" is set to `false`:

```
{
  "start_date": "2023-11-24 03:39:44",
  "start_address": "1255 6th Ave, New York, NY 10020, USA",
  "max_speed": 62,
  "end_date": "2023-11-24 06:39:44",
  "end_address": "888 5th Ave, New York, NY 10021, USA",
  "length": 5.5,
  "points": 327,
  "avg_speed": 49,
  "event_count": 3,
  "norm_fuel_consumed": 1.07,
  "type": "merged",
  "gsm_lbs": false,
  "points_list": [point_info]
}
```

- `start_date` - [date/time](#). Track start date, in user's timezone e.g. "2011-06-18 03:39:44". It signifies the initial point identified as a track for a specified time period.
- `start_address` - string. Track start address.
- `max_speed` - int. Maximum speed registered during period in km/h, e.g. 96.
- `end_date` - [date/time](#). Track end date, in user's timezone e.g. "2011-06-18 05:18:36". It signifies the last point identified as a track for a specified time period.
- `end_address` - string. Track end address.
- `length` - float. Track length during period in kilometers, e.g. 85.5.
- `points` - int. Total number of points in a track, e.g. 724.
- `avg_speed` - int. Average speed in km/h, e.g. 70.
- `event_count` - int. It represents the number of events recorded during this track. This field will not be present if "count\_events" is set to `false`.

- `norm_fuel_consumed` - float. It is representing the amount of fuel consumed during the track, measured in litres. This field will not be present if there's no [vehicle\\_object](#) linked to the tracker or if "normAvgFuelConsumption" is not defined for the linked vehicle object.
- `type` - [enum](#): `regular`, `single_report`, `merged`, `cluster`. Used to distinguish this track type ( `merged` ) from the others.
- `gsm_lbs` - optional boolean. GSM LBS flag.
- `points_list` - array of JSON objects. A list of [point info](#).

`cluster` object. Can be returned only if "split" is set to `true`:

```
{
  "start_date": "2023-11-24 03:39:44",
  "start_address": "1255 6th Ave, New York, NY 10020, USA",
  "end_date": "2020-09-24 06:39:44",
  "precision": 500,
  "points": [{"lat": 34.178868, "lng": -118.599672}, {"lat": 31.738386, "lng": -106.453854}],
  "type": "cluster",
  "gsm_lbs": false
}
```

- `start_date` - [date/time](#). Track start date, in user's timezone e.g. "2011-06-18 03:39:44".
- `start_address` - string. Track start address.
- `end_date` - [date/time](#). Track end date, in user's timezone e.g. "2011-06-18 05:18:36".
- `precision` - optional int. It signifies the precision of the location in meters. Its presence relies on the device model.
- `points` - array of point objects in a cluster.
- `type` - [enum](#): `regular`, `single_report`, `merged`, `cluster`. Used to distinguish this track type ( `cluster` ) from the others.
- `gsm_lbs` - optional boolean. GSM LBS flag. If a cluster only contains GSM LBS points, then this value will be set to `true`.

## errors

- 201 - Not found in database – the tracker ID in your request may not match any trackers linked to the user account with this session hash. Ensure the correct `tracker_id` and hash of an appropriate user are used.
- 208 - Device blocked – if a tracker exists under this user account but is currently inactive due to tariff plan restrictions or any other reason.



- 211 - Requested time span is too big – If the interval between the "from" and "to" dates is too large, it may exceed the maximum value defined in the API configuration.

## read

This method fetches all track points that a GPS tracker has recorded and sent to the platform within a specified time frame. The timestamp for each point corresponds to when the tracker recorded the point, adjusted to the user's time zone.

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). The tracker must be associated with the user whose hash is being used for the request, and not tariff-blocked.	int	123456
from	The start date/time for trips. The response begins with the next trip point after this time.	date/ time	"2020-09-23 03:24:00"
to	An end date/time for trips. The response concludes with the last point before this time. Ensure this date is later than the "from" date.	date/ time	"2020-09-23 06:24:00"
track_id	Optional. If a specific track is identified, only points related to that track will be provided. If no track is specified, all valid track points recorded within the specified "from" and "to" timeframe will be returned.	int	234567
include_gsm_lbs	Optional. Default is <code>true</code> . If the value is <code>false</code> && a track_id is not provided, the GSM LBS points will be excluded from the results.	boolean	true

name	description	type	format
simplify	Optional. Default is <code>true</code> . If set to <code>true</code> , the returned data will be simplified, resulting in fewer points.	boolean	<code>true</code>
point_limit	Optional. If it is specified and <code>simplify=true</code> , the returned data will be reduced to contain that specified number of points. The minimum value is 2, and the maximum is 3000. If it is not specified, the server's default settings for simplifying tracks will be applied. This is not a strict limit; the returned data can potentially contain more points than specified.	int	<code>300</code>
filter	Optional. If this is set to <code>true</code> , the returned tracks will be filtered. This is currently only applicable to LBS tracks. If set to <code>false</code> , the response will include parking points.	boolean	<code>false</code>

### example

For instance, if we need to obtain track points for tracker 1683258 that fall within November 1, and are solely part of track ID 923150, without applying a smart filter, LBS points and simplifier. Since these optional parameters by default are `true`, we should list them in our request.

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/track/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 1683258, "track_id": 923150, "from": "2023-11-01 00:00:00", "to": "2023-11-01 23:59:59", "filter": false, "simplify": false, "include_gsm_lbs": false}'
```

### response

```
{
  "success": true,
  "limit_exceeded": true,
  "list": [point_info]
}
```

- `limit_exceeded` - boolean. It will be `true` if the requested time period surpasses the limit set in the tracker's tariff. For instance, if the device's plan has a maximum storage period of three months (the default value), and we request trips for six months.
- `list` - array of JSON objects. A list of [point info](#).

### point info

```
{
  "lat": 43.0375133,
  "lng": -79.226505,
  "alt": 0,
  "satellites": 10,
  "mileage": 43.93,
  "get_time": "2023-11-01 04:38:39",
  "address": "Kottmeier Road, Thorold, Golden Horseshoe,
Ontario, Canada, L3B 5N6",
  "heading": 280,
  "speed": 53,
  "precision": 100,
  "gsm_lbs": false,
  "parking": false,
  "buffered": true
}
```

- `lat` - float. Represents latitude.
- `lng` - float. Represents longitude.
- `alt` - int. Indicates the altitude in meters.
- `satellites` - int. Shows the number of GPS satellites used to determine this point.
- `mileage` - float. Represents mileage.
- `get_time` - [date/time](#). This is the GPS timestamp of the point, adjusted to the user's timezone.
- `address` - string. Represents the location's address. Will be "" if no address recorded. If no address has been recorded, it will appear as "". An address is recorded when it marks the beginning or end of a trip, or when an event occurs.
- `heading` - int. A value that represents the direction in degrees, with a range of 0 to 360. 0 corresponds to North.
- `speed` - int. A value representing speed in kilometers per hour.

- `precision` - optional int. A value indicating precision in meters. Its presence relies on the device model.
- `gsm_lbs` - optional boolean. It returns `true` if the location was detected by GSM LBS.
- `parking` - optional boolean. It will return true if the point does not correspond to a trip. [Parking detection](#) feature on the platform influences the categorization of points as either trip or parking states.
- `buffered` - optional boolean. It will return `true` if the point was initially saved in the device's memory and then sent to the server later. This parameter may vary based on the tracker model.

## errors

- 201 - Not found in database – the tracker ID in your request may not match any trackers linked to the user account with this session hash. Ensure the correct `tracker_id` and hash of an appropriate user are used.
- 208 - Device blocked – if a tracker exists under this user account but is currently inactive due to tariff plan restrictions or any other reason.
- 211 - Requested time span is too big – If the interval between the "from" and "to" dates is too large, it may exceed the maximum value defined in the API configuration.

## visit/list

This method fetches IDs of zones and places that contain at least one track point.

### parameters

name	description	type	format
<code>tracker_id</code>	ID of the tracker (aka "object_id"). The tracker must be associated with the user whose hash is being used for the request, and not tariff-blocked.	int	123456
<code>from</code>	Start date/time for searching.	<a href="#">date/</a> <a href="#">time</a>	"2024-01-10 00:00:00"
<code>to</code>			

name	description	type	format
	End date/time for searching. Must be after <code>from</code> date.	<a href="#">date/</a> <a href="#">time</a>	"2024-01-20 00:00:00"
<code>include_zones</code>	Optional. Default is <code>true</code> . If the value is <code>false</code> , zones IDs will be excluded.	boolean	true
<code>include_places</code>	Optional. Default is <code>true</code> . If the value is <code>false</code> , places IDs will be excluded.	boolean	true

### example

For instance, if we need to obtain IDs of zones and places that contain at least one track point related to tracker 1683258 in January.

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/track/visit/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":  
1683258, "from": "2024-01-01 00:00:00", "to": "2024-01-31  
00:00:00"}'
```

### response

```
{  
  "success": true,  
  "value": {  
    "zones": [  
      54865,  
      35284  
    ],  
    "places": [  
      18404  
    ]  
  }  
}
```

- `zones` - int array. List of zones IDs.
- `places` - int array. List of places IDs.

**errors**

- 204 - Entity not found – the tracker ID in your request may not match any trackers linked to the user account with this session hash. Ensure the correct tracker\_id and hash of an appropriate user are used.

Last update: April 2, 2024







# Waybill

This resource contains information to download waybill report for tracks.

## API actions

API path: `/track/waybill`.

### download

Downloads a waybill report DOCX file for tracks of the specified tracker and time period.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
from	From date/time.	date/ time	"2020-09-23 03:24:00"
to	To date/time. Specified date must be after "from" date.	date/ time	"2020-09-23 06:24:00"
filter	Optional, default= <code>true</code> . If <code>true</code> , tracks which are too short (in terms of length and number of points) will be omitted from resulting list.	boolean	true
split	Optional, default= <code>true</code> . If <code>false</code> , all tracks will be merged into single one.	boolean	false

name	description	type	format
include_gsm_lbs	Optional, default= <code>true</code> . If <code>false</code> , GSM LBS tracks will be filtered out.	boolean	false
cluster_single_reports	Optional, default= <code>false</code> . If <code>true</code> , single point reports will be clustered by its coordinates.	boolean	false
type	Should be one of "form3", "form3ext", "form4c".	enum	"form4c"
fill_history	If <code>false</code> , only basic info about driver/garage/ vehicle will be filled (no trips or parkings).	boolean	false
fill_odometer	Optional, default= <code>false</code> . If <code>true</code> , mileage readings will be inserted in appropriate fields of the document.	boolean	false
series	Optional. Waybill series.	string	"A-1"
number	Waybill number.	string	"123456789"

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/track/waybill/download' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "from": "2020-09-23 03:24:00", "to": "2020-09-23 06:24:00", "type": "form4c", "fill_history": false, "number": "1234567"}'
```

## response

A docx file with the waybill.

**errors**

- 236 - Feature unavailable due to tariff restrictions – if one of the trackers has tariff without "app\_fleet" feature.

Last update: December 26, 2022



# Waybill settings

Contains API call to get the last waybill number. Waybill number saved when new waybill had downloaded. If it had only digits, then it was incremented before saving.

## API actions

API base path: `track/waybill/settings/`.

### read

Gets last waybill number.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/track/waybill/settings/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/track/waybill/settings/read?
hash=a6aa75587e5c59c32d347da438505fc3
```

### response

```
{
  "success": true,
  "value": {
    "number": "test123"
  }
}
```

### errors

- 201 - Not found in the database – if user have never downloaded a waybill.

Last update: July 19, 2022





# Working with trackers

This document contains tracker object structure and API calls to interact with it. Tracker is one of the key entities in our API. It represents tracking device registered in our GPS monitoring system. Lots of API calls created for manipulation of tracker and/or its properties.

## Tracker object structure

```
{
  "id": 123456,
  "label": "tracker label",
  "clone": false,
  "group_id": 167,
  "avatar_file_name" : "file name",
  "source": {
    "id": 234567,
    "device_id": 99999999888888,
    "model": "telmb920",
    "blocked": false,
    "tariff_id": 345678,
    "status_listing_id": null,
    "creation_date": "2011-09-21",
    "tariff_end_date": "2016-03-24",
    "phone": "71234567890"
  },
  "tag_bindings": [{
    "tag_id": 456789,
    "ordinal": 4
  }]
}
```

- `id` - int. Tracker ID aka `object_id`.
- `label` - string. Tracker label.
- `clone` - boolean. `true` if this tracker is clone.
- `group_id` - int. Tracker group ID, 0 when no group.
- `avatar_file_name` - string. Optional. Passed only if present.
- `source` - object.
  - `id` - int. Source ID.
  - `device_id` - string. Device ID aka `source_imei`.
  - `model` - string. Tracker model name from "models" table.



- `blocked` - boolean. `true` if tracker blocked due to tariff end.
- `tariff_id` - int. An ID of tracker tariff from "main\_tariffs" table.
- `status_listing_id` - int. An ID of the status listing associated with this tracker, or null.
- `creation_date` - [date/time](#). Date when the tracker registered.
- `tariff_end_date` - [date/time](#). Date of next tariff prolongation, or null.
- `phone` - string. Phone of the device. Can be null or empty if device has no GSM module or uses bundled SIM which number hidden from the user.
- `tag_binding` - object. List of attached tags. Appears only for [tracker/list](#) call.
  - `tag_id` - int. An ID of tag. Must be unique for a tracker.
  - `ordinal` - int. Number that can be used as ordinal or kind of tag. Must be unique for a tracker. Max value is 5.

## API actions

API base path: `/tracker`.

### read

Gets tracker info by ID.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id")	int	999199

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 123456}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/read?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456
```

## response

```
{
  "success": true,
  "value": {
    "id": 123456,
    "label": "Object 1",
    "group_id": 0,
    "source": {
      "id": 10021901,
      "device_id": "123456789009876",
      "model": "atrack_ak11",
      "blocked": false,
      "tariff_id": 1294,
      "phone": "79161234533",
      "status_listing_id": null,
      "creation_date": "2021-09-20",
      "tariff_end_date": "2021-09-24"
    },
    "tag_bindings": [],
    "clone": false
  }
}
```

See tracker object structure description [here](#).

## errors

- 201 - Not found in the database – if tracker not found.

## list

Gets user's trackers with optional filtering by labels. We described this API call in our [how-tos](#).

## parameters

name	description	type	format
labels	Optional. List of tracker label filters. If specified, only trackers that labels contains any of the given filter will be returned.	string array	[ "aa", "b" ]

Constraints for labels:

- Labels array size: minimum 1, maximum 1024.
- No null items.

- No duplicate items.
- Item length: minimum 1, maximum 60.

For example, we have trackers with labels "aa1", "bb2", "cc3", if we pass `labels=["aa","b"]` only trackers containing "aa1" and "bb2" will be returned.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [{
    "id": 123456,
    "label": "tracker label",
    "clone": false,
    "group_id": 167,
    "avatar_file_name" : "file name",
    "source": {
      "id": 234567,
      "device_id": 9999999988888,
      "model": "telfmb920",
      "blocked": false,
      "tariff_id": 345678,
      "status_listing_id": null,
      "creation_date": "2011-09-21",
      "tariff_end_date": "2016-03-24",
      "phone" : "+71234567890"
    },
    "tag_bindings": [{
      "tag_id": 456789,
      "ordinal": 4
    }]
  }]
}
```

See tracker object structure description [here](#).

## errors

[General](#) types only.

## corrupt

Marks tracker as deleted and corrupt its source, device\_id and phone.

**required sub-user rights:** `tracker_register`.

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999119

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/corrupt' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tracker/corrupt?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

### response

```
{ "success": true }
```

### errors

- 13 – Operation not permitted – if tracker already connected to server, or if user has insufficient rights.
- 243 – Device already connected.
- 201 – Not found in the database - if tracker not found.
- 219 – Not allowed for clones of the device - if source tracker is clone itself.
- 252 – Device already corrupted.
- 208 – Device blocked.

## delete

Deletes a tracker if it is "clone". Will not work if specified ID of the original tracker.

**required sub-user rights:** `admin` (available only to master users).

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999119

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/delete' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/delete?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

#### response

```
{ "success": true }
```

#### errors

- 201 - Not found in the database – if tracker not found.
- 249 - Operation available for clones only – if tracker is not clone.
- 203 - Delete entity associated with – if there are some rules or vehicles associated with tracker.

```
{
  "success": false,
  "status": {
    "code": 203,
    "description": "Delete entity associated with"
  },
  "rules": [10]
}
```

or

```
{
  "success": false,
  "status": {
    "code": 203,
```

```

        "description": "Delete entity associated with"
    },
    "vehicles": [11]
}

```

- `rules` - list of associated rule IDs.
- `vehicles` - list of associated vehicle IDs.

## change\_phone

Changes tracker's phone and setup new apn.

**required sub-user rights:** `tracker_configure`.

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199
phone	The phone number of the sim card inserted into device in international format without "+" sign.	string	"6156680000"
apn_name	The name of GPRS APN of the sim card inserted into device. Max length 40.	string	"fast.tmobile.com"
apn_user	The user of GPRS APN of the sim card inserted into device. Max length 40, can be empty.	string	"tmobile"
apn_password	The password of GPRS APN of the sim card inserted into device. Max length 40, can be empty.	string	"tmobile"

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/change_phone' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489, "phone": "6156680000", "apn_name": "fast.tmobile.com", "apn_user": "tmobile", "apn_password": "tmobile"}'
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if tracker not found.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 219 – Not allowed for clones of the device - if specified tracker is a clone.
- 214 – Requested operation or parameters are not supported by the device - if device does not have GSM module.
- 223 – Phone number already in use - if specified phone number already used in another device.
- 241 – Cannot change phone to bundled sim. Contact tech support. If specified phone number belongs tp sim card bundled with the device.

## get\_diagnostics

Gets last CAN and OBD sensors and states values received from the device.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999119

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/get_diagnostics' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/get_diagnostics?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

## response

```
{
  "success": true,
  "user_time": "2021-05-20 13:49:09",
  "inputs": [{
    "label": "OBD: RPM",
    "units": "",
    "name": "obd_rpm",
    "type": "rpm",
    "value": 889.0,
    "units_type": "custom",
    "converted_units_type": null,
    "converted_value": null
  }],
  "states": {
    "obd_vin": "123",
    "obd_mil_status": "0"
  },
  "update_time": "2021-05-20 13:48:02"
}
```

- `user_time` - [date/time](#). Current time in user's timezone.
- `inputs` - list of `sensor value` objects.
  - `label` - string. Sensor's label. E.g. "Sensor #1".
  - `name` - [enum](#). Name of sensor's raw input.
  - `type` - [enum](#). Type of quantity, measured by a sensor.
  - `value` - float. Reading's value, measured in units from an eponymous field. E.g. 100.0.
  - `units_type` - [enum](#). Unit of measurement of input to the sensor.
  - `units` - string. User label for sensor's units.
  - `converted_units_type` - [enum](#). Unit of measurement system preferred by current user (according to user/settings), suitable for this sensor. Can be null, if



there is no need in conversion (unit of sensor's input (field `units_type`) belongs to user's measurement system).

- `converted_value` - float. Reading's value in units from field `converted_units_type`. Can be null if there is no need in conversion.
- `states` - map of last state values or null (see below).
- `update_time` - [date/time](#). Date and time when the data updated.

List of available sensor's input names for the object `sensor value`:

- **`obd_consumption.`**
- **`obd_rpm.`**
- **`obd_fuel.`**
- **`obd_coolant_t.`**
- **`obd_intake_air_t.`**
- **`obd_throttle.`**
- **`obd_speed.`**
- **`obd_engine_load.`**
- **`obd_absolute_load_value`** (normalised value of air mass per intake stroke in percents).
- **`obd_control_module_voltage`** (in volts).
- **`obd_time_since_engine_start`** (run time since engine start in seconds).
- **`obd_mil_run_time`** (in minutes).
- **`can_engine_temp.`**
- **`can_engine_hours.`**
- **`can_mileage.`**
- **`can_throttle.`**
- **`can_consumption.`**
- **`can_rpm.`**
- **`can_speed.`**
- **`can_r_prefix.`**
- **`can_coolant_t.`**
- **`can_intake_air_t.`**
- **`can_engine_load.`**
- **`can_adblue_level.`**
- **`can_fuel_rate`** (instant fuel consumption liter/hour).

- **raw\_can\_x** (range for x: [1 – 16]).
- **can\_axle\_load\_x** (range for x: [1 – 15]).

List of state names for the field `states` :

- **obd\_vin** (value type: string).
- **obd\_dtc\_number** (DTC codes number; value type: integer).
- **obd\_dtc\_codes** (DTC codes; value type: string).
- **obd\_dtc\_cleared\_distance** (distance traveled since codes cleared in km; value type: double).
- **obd\_mil\_activated\_distance** (distance traveled with MIL on in km; value type: double).
- **hardware\_key** (driver identification key; value type: string).
- **external\_power\_state** (connected/disconnected; value type: string).
- **driver\_ident\_state** (identified/not identified; value type: string).
- **tacho\_vin** (value type: string).
- **tacho\_card1\_sn** (value type: string).
- **tacho\_card2\_sn** (value type: string).
- **tacho\_vin\_last\_download** (value type: string).
- **tacho\_card1\_last\_download** (value type: string).
- **tacho\_card2\_last\_download** (value type: string).
- **can\_hood\_state** (value type: string, 0 or 1 means "close" or "open").
- **can\_airbag\_state** (value type: string, 0 or 1 means "normal" or "malfunction").
- **can\_trunk\_state** (value type: string, 0 or 1 means "close" or "open").
- **can\_seat\_belt\_driver\_state** (value type: string, 0 or 1 means "untied" or "tied").
- **can\_seat\_belt\_passenger\_state** (value type: string, 0 or 1 means "untied" or "tied").
- **can\_door\_state** (value type: string, 0 or 1 means "close" or "open").
- **can\_door\_driver\_state** (value type: string, 0 or 1 means "close" or "open").
- **can\_door\_passenger\_state** (value type: string, 0 or 1 means "close" or "open").

You can locate all inputs, states, and definitions by utilizing the [tracker/sensor/input\\_name/list](#) API call.

#### errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.

- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

## get\_fuel

Gets current fuel level (in liters) of tracker's fuel tanks.

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999119

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/get_fuel' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tracker/get_fuel?  
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

### response

```
{  
  "success": true,  
  "user_time": "2021-05-20 13:49:09",  
  "inputs": [{  
    "label": "Sensor #1",  
    "name": "can_fuel",  
    "type": "fuel",  
    "value": 100.0,  
    "units_type": "litre",  
    "units": "litres",  
    "converted_units_type": null,  
    "converted_value": null  
  }],  
  "update_time": "2021-05-20 13:48:02"  
}
```

- user\_time - [date/time](#). Current time in user's timezone.

- `inputs` - array of last readings of fuel-related sensors. Items are object listed below.

List of available sensor's input names for the object `sensor value` :

- **`fuel_level`**.
- **`fuel_frequency`**.
- **`lfs_level_x`** (range for x: [1 – 16]).
- **`fuel_consumption`**.
- **`rs232_x`** (range for x: [1 – 6]).
- **`can_fuel`** (fuel level in percents or in unknown units).
- **`can_fuel_2`** (fuel level in percents or in unknown units).
- **`can_fuel_litres`** (fuel level in litres).
- **`can_fuel_economy`** (fuel economy in km/litres).
- `update_time` - [date/time](#). Date and time when the data updated.

#### errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

#### get\_inputs

Gets current state of tracker's digital inputs and "semantic" inputs (ignition, buttons, car alarms, etc.) bound to them (if any).

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999119

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/get_inputs' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/get_inputs?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

## response

```
{
  "success": true,
  "user_time": "2021-05-20 13:49:09",
  "inputs": [true, true, false],
  "states": [
    {
      "type": "ignition",
      "name": "DIN1",
      "status": true,
      "input_number": 1
    }
  ],
  "update_time": "2021-05-20 13:48:02"
}
```

- `user_time` - [date/time](#). Current time in user's timezone.
- `inputs` - array (boolean) of states of all digital inputs. `[true, true, false]` means input 1 is on, input 2 is on, input 3 is off.
- `states` - array of state objects.
  - `type` - [enum](#). One of predefined semantic input types (see below).
  - `name` - string. User-defined name for semantic input, or null if not specified.
  - `status` - boolean. True if input is active, false otherwise.
  - `input_number` - int. Number of the associated discrete input.
- `update_time` - [date/time](#). Date and time when the data updated.

List of `input` types:

- **ignition** - Car's ignition. There can be only one sensor of this type.
- **engine** - Engine's working status.
- **mass** - Car's "ground".
- **car\_alarm** - Expected to be "on" when car alarm triggered.

- **sos\_button** - An emergency "red" button.
- **hood** - "on" if engine's hood is open.
- **door** - "on" if car's door is open.
- **car\_lock** - "on" if car's central lock is open.
- **custom** - user-defined type. In general, should have non-empty "name" field.

#### errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

### get\_last\_gps\_point

Gets last point of the tracker located by GPS. Points located by GSM LBS are excluded from consideration.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999119

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/get_last_gps_point' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/get_last_gps_point?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

#### response

```
{
  "success" : true,
  "value" : {
```

```

    "get_time" : "2012-03-05 12:00:00",
    "heading" : 11,
    "lat" : 22.0,
    "lng" : 33.0,
    "satellites" : 5,
    "speed" : 20,
    "precision" : 100
  }
}

```

- `value` - track point object.
- `get_time` - [date/time](#). GPS timestamp of the point, in user's timezone.
- `heading` - int. Direction bearing in degrees (0-360).
- `lat` - float. Latitude.
- `lng` - float. Longitude.
- `satellites` - int. Number of satellites used in fix for this point.
- `speed` - int. Speed in km/h.
- `precision` - int. Optional. Exists if not equal to 0. Precision in meters.

#### errors

- 201 - Not found in the database – if there is no tracker with such ID belonging to authorized user.
- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions or some other reason.

## get\_readings

Gets last sensor values for sensors that are:

- **metering.**
- **not can- or obd-based.**
- **not "fuel" sensors.**

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999119

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/get_readings' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/get_readings?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

## response

```
{
  "success":true,
  "user_time":"2021-05-20 13:49:09",
  "inputs":[{
    "label":"Board voltage",
    "units":"V",
    "name":"board_voltage",
    "type":"power",
    "value":13.562,
    "units_type":"custom",
    "converted_units_type":null,
    "converted_value":null
  }],
  "update_time":"2021-05-20 13:48:02"
}
```

- `user_time` - [date/time](#). Current time in user's timezone.
- `inputs` - list of `sensor value` objects. See below.
- `label` - string. Sensor's label. E.g. "Sensor #1".
- `name` - [enum](#). Name of sensor's raw input.
- `type` - [enum](#). Type of quantity, measured by a sensor.
- `value` - float. Reading's value, measured in units from an eponymous field. E.g. 100.0.
- `units_type` - [enum](#). Unit of measurement of input to the sensor.
- `units` - string. User label for sensor's units.
- `converted_units_type` - [enum](#). Unit of measurement system preferred by current user (according to user/settings), suitable for this sensor. Can be null, if there is no need in conversion (unit of sensor's input (field `units_type`) belongs to user's measurement system).
- `converted_value` - float. Reading's value in units from field `converted_units_type`. Can be null if there is no need in conversion.



- `update_time` - [date/time](#). Date and time when the data updated.

List of available sensor's input names for the object `sensor value`:

- **composite.**
- **input\_status.**
- **analog\_x** (range for x: [1 – 8]).
- **freq\_x** (range for x: [1 – 8]).
- **impulse\_counter\_x** (range for x: [1 – 8]).
- **fuel\_temperature.**
- **lts\_temperature\_x** (range for x: [1 – 16]).
- **rs232\_x** (range for x: [1 – 6]).
- **board\_voltage.**
- **temp\_sensor.**
- **ext\_temp\_sensor\_x** (range for x: [1 – 10]).

#### errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

#### get\_state

Gets current tracker state (gps, gsm, outputs, etc.).

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999119

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/get_state' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":  
265489}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/get_state?  
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

## response

```
{  
  "user_time": "2022-08-31 13:47:13",  
  "state": {  
    "source_id": 545139,  
    "gps": {  
      "updated": "2022-08-31 13:47:09",  
      "signal_level": 100,  
      "location": {  
        "lat": 42.82769,  
        "lng": -78.26290833333333  
      },  
      "heading": 45,  
      "speed": 0,  
      "alt": 0  
    },  
    "connection_status": "active",  
    "movement_status": "parked",  
    "gsm": {  
      "updated": "2022-08-31 13:47:09",  
      "signal_level": 100,  
      "network_name": "Mobile",  
      "roaming": false  
    },  
    "last_update": "2022-08-31 13:47:09",  
    "battery_level": 97,  
    "battery_update": "2022-08-31 13:47:09",  
    "inputs": [false, false, false],  
    "inputs_update": "2022-08-31 13:47:09",  
    "outputs": [true, false],  
    "outputs_update": "2022-08-31 13:47:09",  
    "additional": {  
      "hardware_key": {  
        "value": "20910998202956382057",  
        "updated": "2022-08-31 10:47:09"}},  
      "actual_track_update": "2022-08-31 13:40:44"  
    },  
    "success": true  
  }  
}
```

- `user_time` - [date/time](#). Current time in user's timezone.

- `source_id` - int. Tracker data source ID (from "sources" table).
- `gps` - gps object.
  - `updated` - [date/time](#). Date of last gps coordinates update in a timezone of the user or null if there are no updates.
  - `signal_level` - int. GPS signal level in percent, e.g. 25, or null if device cannot provide such info.
  - `lat` - float. Latitude.
  - `lng` - float. Longitude.
  - `heading` - int. Direction bearing in degrees (0-360).
  - `speed` - int. Speed in km/h, e.g. 20.
  - `alt` - int. Altitude in meters, e.g. 10.
  - `precision` - int. Optional. Precision in meters.
  - `gsm_lbs` - boolean. Optional. True if location detected by GSM LBS.
- `connection_status` - [enum](#). Device connection status, possible values: "signal\_lost", "just\_registered", "offline", "idle", "active".
- `movement_status` - [enum](#). Movement status, possible values: "moving", "stopped", "parked".
- `gsm` - object. Can be null if device does not support transmission of gsm info.
  - `updated` - [date/time](#). Date of last gsm status update in a timezone of the user or null if there are no updates.
  - `signal_level` - int. GSM signal level in percent, e.g. 25, or null if device cannot provide such info.
  - `network_name` - string. GSM network name, e.g. "T-MOBILE", or null if device cannot provide such info.
  - `roaming` - boolean. Roaming state, or null if device cannot provide such info.
- `last_update` - [date/time](#). Date of last device state update in a timezone of the user or null if there are no updates.
- `battery_level` - int. Battery level in percent, e.g. 25, or null if device cannot provide such info.
- `battery_update` - [date/time](#). Date of last battery update in a timezone of the user or null if there are no updates.
- `inputs` - array of boolean. States of all digital inputs. `[true, true, false]` means input 1 is on, input 2 is on, input 3 is off.
- `inputs_update` - [date/time](#). Date of last inputs update in a timezone of the user or null if there are no updates.

- `outputs` - array of boolean. States of all digital outputs. `[true, true, false]` means output 1 is on, output 2 is on, output 3 is off.
- `outputs_update` - [date/time](#). Date of last outputs update in a timezone of the user or null if there are no updates.
- `additional` - object. map of additional states, keys depends on tracker model.
  - `hardware_key` - last scanned hardware key object.
    - `value` - int. Hardware key.
    - `updated` - [date/time](#). Date of last hardware key update in a timezone of the user or null if there are no updates.
- `actual_track_update` - [date/time](#). When the last track was updated last time, when device last time moved.

### errors

- 201 – Not found in the database (if there is no tracker with such ID belonging to authorized user).
- 208 – Device blocked (if tracker exists but was blocked due to tariff restrictions or some other reason).

### get\_states

Gets current states (gps, gsm, outputs, etc.) for several trackers.

## parameters

name	description	type	format
trackers	ID of trackers (aka "object_id"). Trackers must belong to authorized user and not be blocked.	int array	[999119, 999199]
list_blocked	Optional. If <code>true</code> call returns list of blocked tracker IDs instead of error 208. Default is <code>false</code> .	boolean	true/false
allow_not_exist	Optional. If <code>true</code> call returns list of nonexistent tracker IDs instead of error 217 or 201. Default is <code>false</code> .	boolean	true/false

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/get_states' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "trackers": "[999119, 999199, 9991911]"}'
```

## response

```
{
  "success": true,
  "user_time": "2014-07-09 07:50:58",
  "states": {
    "999119": {
      "source_id": 65894,
      "gps": {
        "updated": "2013-02-19 10:48:08",
        "signal_level": 25,
        "location": {
          "lat": 34.178868,
          "lng": -118.599672
        },
        "heading": 45,
        "speed": 20,
        "alt": 10,
        "precision": 50,
        "gsm_lbs": false
      },
      "connection_status": "active",
    }
  }
}
```

```

    "movement_status": "moving",
    "gsm": {
      "updated": "2013-02-19 10:48:08",
      "signal_level": 70,
      "network_name": "T-MOBILE",
      "roaming": false
    },
    "last_update": "2013-02-19 10:48:08",
    "battery_level": 100,
    "battery_update": "2013-02-19 10:48:08",
    "inputs": [
      true,
      true,
      false
    ],
    "inputs_update": "2013-02-19 10:48:08",
    "outputs": [
      true,
      true,
      false
    ],
    "outputs_update": "2013-02-19 10:48:08",
    "additional": {
      "hardware_key": {
        "value": 564648745158875,
        "updated": "2013-02-19 10:48:08"
      }
    }
  },
  "blocked": [999199],
  "not_exist": [9991911]
}

```

- `user_time` - `date/time`. Current time in user's timezone.
- `states` - object. A map containing state objects for requested trackers, where the key is the tracker ID and the value is the state (see state object description in [tracker/get\\_state](#) response).
- `blocked` - array of tracker IDs. Returned only if `list_blocked= true`.
- `not_exist` - array of tracker IDs. Returned only if `allow_not_exist= true`.

## errors

- 201 – Not found in the database (if tracker corrupted and `allow_not_exist = false`).
- 208 – Device blocked (if `list_blocked = false` and tracker exists but was blocked due to tariff restrictions or some other reason).
- 217 – List contains nonexistent entities (if `allow_not_exist = false` and there are nonexistent trackers belonging to an authorized user).

## list\_models

Gets all integrated tracker models (from "models" table).

### parameters

name	description	type	format
compact_view	Optional. <code>true</code> to compact view. Default is <code>false</code> .	boolean	true/false
compact_index	Optional. <code>true</code> to compact view the indexed inputs: returns only input with max index. Default is <code>false</code> , but this value is deprecated.	boolean	true/false
codes	Optional. Array of model codes. If passed only given models will be returned.	string array	<code>[model_1,</code> <code>model_2, ...]</code>

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/list_models' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3",  
    "compact_index": true}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tracker/list_models?  
hash=a6aa75587e5c59c32d347da438505fc3
```

### response

```
{  
  "id": 166,  
  "code": "tt1_wp",  
  "type": "vehicle",  
  "name": "WondeProud TT1",  
  "id_type": "10,2",  
  "has_phone": true,  
  "has_apn_settings": true,  
  "register": true,  
  "battery": {  
    "min_charge": 3.4,  
    "low_charge": 3.7,  
  },  
}
```

```

        "max_charge": 4.1
    },
    "altitude": true,
    "satellites": true,
    "gsm_level": true,
    "gsm_network": true,
    "gsm_roaming": true,
    "has_detach_button": false,
    "has_fuel_input": true,
    "analog_inputs": 2,
    "digital_inputs": 4,
    "rs232_inputs": 0,
    "digital_outputs": 4,
    "track_control": "tt1",
    "output_control": "default",
    "special_control": "none",
    "vendor": "WondeProud",
    "rules": [
        "offline",
        "input_change",
        "sos",
        "sensor_range",
        "speedup",
        "route",
        "track_change",
        "inoutzone",
        "battery_off"
    ],
    "inputs": ["analog_2"],
    "state_fields": [],
    "special_settings": ["none"],
    "sms_control": [],
    "has_led_control": false,
    "has_location_request": true,
    "has_gsm_lbs_location_request": true,
    "has_chat": false,
    "check_bundle": false,
    "has_odometer": true
}

```

- `id` - int. Model ID.
- `vendor` - string. Vendor name.
- `parent_code` - string. Can be null.
- `type` - enum. Can be "logger", "portable", "vehicle", or "personal".
- `name` - string. Model name.
- `has_auto_registration` - boolean. If `true` device may register by automatic commands from the platform.
- `battery` - object. An internal device's battery.
  - `low_charge` - float. Charge level for the "low battery" rule triggers.
- `analog_inputs` - int. Number of analog inputs.



- `digital_inputs` - int. Number of digital inputs.
- `digital_outputs` - int. Number of digital outputs.
- `rs232_inputs` - int. Number of RS232 inputs.
- `inputs` - array of [enum](#). All available input types.
- `rules` - array of [enum](#). Supported rules.
- `has_led_control` - boolean. `true` if a switching LED supported by this tracker.
- `has_location_request` - boolean. `true` if the tracker has an opportunity to request a location with a command by SMS.
- `has_gprs_location_request` - boolean. `true` if the tracker has an opportunity to request a location with a command over a GPRS connection.
- `has_gsm_lbs_location_request` - boolean. `true` if the tracker has an opportunity to request a location by LBS with a command over a GPRS connection.
- `has_chat` - boolean. `true` if chat available for the device.
- `has_odometer` - boolean. `true` if the tracker has an integrated odometer.
- `has_lbs` - boolean. `true` if the tracker sends information about cell info.
- `has_motion_sensor` - boolean. `true` if the tracker has an integrated motion sensor.
- `has_hardware_key` - boolean. `true` if the tracker has an opportunity for identification of a driver by a hardware key.
- `additional_fields` - optional. List of descriptions of special fields using for control trackers that users fill on time of registration.

#### ID TYPE

An ID type used to determine the information needed to register device in our system (see [tracker/register](#)).

Possible values are:

- **imei** – means device uses IMEI as its identifier, e.g. "356938035643809". See [Wikipedia article](#). When needed, you should pass only digits of IMEI, no spaces, minus signs, etc.
- **meid** means device uses MEID consisting of 14 HEX digits as its identifier, e.g. "A10000009296F2". See [Wikipedia article](#).
- **id,n** – means device uses n-digit identifier (factory ID with length N), for example, "id,7" means that you must pass 7-digit number, for example "1234567".
- **n,m** – n-digit generated ID starting with M. This means that device has configurable ID and our platform generates and configures it automatically. You don't need to pass any identifier during device registration in this case.

## errors

[General](#) types only.

## tags/set

Set tags for a tracker. Tags must be created.

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999119
tag_bindings	List of <code>tag_binding</code> objects.	array of Json objects	<pre>[{"tag_id" : 1, "ordinal" : 1}, {"tag_id" : 2, "ordinal" : 2}]</pre>

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/tags/set' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
123456, "tag_bindings": "[{"tag_id" : 1, "ordinal" : 1},
{"tag_id" : 2, "ordinal" : 2}]"}'
```

## response

```
{ "success": true }
```

## errors

[General](#) types only.

## location\_request

Execute this command to get current position of the device. The device must support requesting function.

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999119
type	Optional. Default type <code>sms</code> .	enum	"sms"

Request types:

- **sms** – GNSS data via SMS. Will send an SMS to request location. SMS gateway must be installed for the panel.
- **gsm** – GSM LBS data via GPRS. Device must have `online` or `GPS not updated` status.
- **gprs** – GNSS data via GPRS. Device must have `online` or `GPS not updated` status.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/location_request' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":  
123456}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tracker/location_request?  
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456
```

### response

```
{ "success": true }
```

### errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

- 213 – Cannot perform action: the device is offline.
- 214 – Requested operation or parameters are not supported by the device.
- 256 – Location already actual.

## register\_quick

Registers a new tracker using only IMEI. Automatic SMS commands will not be sent for a register. The device must be preconfigured. This API call can be used only for bundles.

**required sub-user rights:** `tracker_register`.

### parameters

name	description	type	format
label	User-defined label for this tracker. Must consist of printable characters and have length between 1 and 60.	string	"Courier"
group_id	Tracker group ID, 0 if tracker does not belong to any group. The specified group must exist. See <a href="#">group/list</a> .	int	0
imei	Tracker's IMEI.	string	"35645587458999"

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/register_quick' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "label":  
  "Courier", "group_id": 0, "imei": "35645587458999"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tracker/register_quick?  
hash=a6aa75587e5c59c32d347da438505fc3&label=Courier&group_id=0&imei=35
```

### response

```
{  
  "success": true,  
  "value": {
```

```

    "id": 123456,
    "label": "tracker label",
    "clone": false,
    "group_id": 167,
    "avatar_file_name" : "file name",
    "source": {
      "id": 234567,
      "device_id": 9999999988888,
      "model": "telfmb920",
      "blocked": false,
      "tariff_id": 345678,
      "status_listing_id": null,
      "creation_date": "2011-09-21",
      "tariff_end_date": "2016-03-24",
      "phone" : "71234567890"
    },
    "tag_bindings": [{
      "tag_id": 456789,
      "ordinal": 4
    }]
  }
}

```

For `tracker` object structure, see [tracker/](#).

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 201 – Not found in the database - if there is no bundle with such IMEI.
- 204 – Entity not found - if specified group does not exist.
- 220 – Unknown device model - if specified device model does not exist.
- 221 – Device limit exceeded - if device limit set for the user's dealer has been exceeded.
- 222 – Plugin not found - if specified plugin not found or is not supported by device model.
- 223 – Phone number already in use - if specified phone number already used in another device.
- 224 – Device ID already in use - if specified device ID already registered in the system.
- 225 – Not allowed for this legal type - if tariff of the new device is not compatible with user's legal type.
- 226 – Wrong ICCID - if specified ICCID was not found.
- 227 – Wrong activation code - if specified activation code not found or is already activated.

## register\_retry

Resends registration commands to the device. The panel must have installed SMS gateway.

**required sub-user rights:** `tracker_register`.

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999119
device_id	Optional. Device ID that was used to register, e.g. IMEI. It can be used instead of <code>tracker_id</code> for models with a fixed ID.	string	"4568005588562"
apn_name	The name of GPRS APN of this sim card inserted into device. Max length 40.	string	"fast.tmobile.com"
apn_user	The user of GPRS APN of this sim card inserted into device. Max length 40, can be empty.	string	"tmobile"
apn_password	The password of GPRS APN of the sim card inserted into device. Max	string	"tmobile"

name	description	type	format
	length 40, can be empty.		
send_register_commands	Indicates send or not to send activation commands to device (via SMS or GPRS channel). If parameter is not specified or equals <code>null</code> will be used the platform settings. Default: <code>null</code> .	boolean	true or false

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/register_retry' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 999119, "apn_name": "fast.tmobile.com", "apn_user": "tmobile", "apn_password": "tmobile", "send_register_commands": true}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/register_retry?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=999119&apn_name=fast.
```

## response

```
{
  "success": true,
  "value": {
    "id": 123456,
    "label": "tracker label",
    "clone": false,
    "group_id": 167,
    "avatar_file_name" : "file name",
    "source": {
      "id": 234567,
      "device_id": 9999999988888,
      "model": "telfmb920",
      "blocked": false,
      "tariff_id": 345678,
      "status_listing_id": null,
    }
  }
}
```

```
    "creation_date": "2011-09-21",
    "tariff_end_date": "2016-03-24",
    "phone" : "+71234567890"
  },
  "tag_bindings": [{
    "tag_id": 456789,
    "ordinal": 4
  }]
}
```

For `tracker` object structure, see [tracker/](#).

### errors

- 13 – Operation not permitted – if user has insufficient rights.
- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 219 – Not allowed for clones of the device - if specified tracker is a clone.
- 214 – Requested operation or parameters are not supported by the device - if device does not have GSM module.
- 242 – Device already connected - if tracker connected to the server.

## register

Registers a new tracker device. During registration, device linked with current API user's account and automatically configured to send data to our servers (if device model supports it). The panel must have installed SMS gateway.

Find detailed instructions on tracker registration [there](#).

**required sub-user rights:** `tracker_register`.

### parameters

#### Important

Because of the variety of tracker models and business applications, there are different ways to register tracker in our system. They are called [Registration plugins](#). Each of registration plugins has its own set of additional parameters.



In addition to parameters specified in this section, pass all parameters which are required by the plugin you have chosen. See example below.

Common parameters are:

name	description	type	format
label	User-defined label for this tracker. Must consist of printable characters and have length between 1 and 60.	string	"Courier"
group_id	Tracker group ID, 0 if tracker does not belong to any group. The specified group must exist. See <a href="#">group/list</a> .	int	0
model	A code of one of the supported models. See <a href="#">tracker/list_models</a> .	string	"pt10"
plugin_id	An ID of a registration plugin which will be used to register the device. See <a href="#">Registration plugins</a> .	int	37
device_id	<b>Must</b> be specified if device model uses fixed device ID. See <a href="#">tracker/list_models</a> .	string	"4568005588562"
send_register_commands	Indicates send or not to send activation commands to	boolean	true or false

name	description	type	format
	device (via SMS or GPRS channel). If parameter is not specified or equals <code>null</code> will be used the platform settings. Default: <code>null</code> .		

## examples

In this example we use plugin ID = 37 (see [Plugin description](#)) to register Queclink GV55Lite. We chose to include the device to default group, so group ID is 0. As this device identified by IMEI, we include it as device ID (123451234512346).

Also, we include **phone**, **apn\_name**, **apn\_user**, **apn\_password** of the sim card installed in device and **activation\_code** since these parameters required by the plugin.

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/register' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "label": "Courier", "group_id": 0, "plugin_id": 37, "model": "qlgv55lite", "phone": "79123122312", "activation_code": "123123123", "device_id": "123451234512346", "apn_name": "fast.tmobile.com", "apn_user": "tmobile", "apn_password": "tmobile"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/register?
hash=a6aa75587e5c59c32d347da438505fc3&label=Courier&group_id=0&plugin_
```

## response

```
{
  "success":true,
  "value":{
    "id":833389,
    "label":"Courier",
    "group_id":0,
    "source":{
      "id":526383,
      "device_id":"123451234512346",
      "model":"qlgv55lite",
      "blocked":false,
      "tariff_id":12163,
      "phone":"79123122312",
      "status_listing_id":null,
    }
  }
}
```

```

    "creation_date": "2021-06-03",
    "tariff_end_date": "2021-06-17"
  },
  "clone": false
}

```

For `tracker` object structure, see [tracker/](#).

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 204 – Entity not found - if specified group does not exist. See [group/list](#).
- 220 – Unknown device model - if specified device model does not exist.
- 221 – Device limit exceeded - if device limit set for the user's dealer has been exceeded.
- 222 – Plugin not found - if specified plugin not found or is not supported by device model.
- 223 – Phone number already in use - if specified phone number already used in another device.
- 224 – Device ID already in use - if specified device ID already registered in the system.
- 225 – Not allowed for this legal type - if tariff of the new device is not compatible with user's legal type.
- 226 – Wrong ICCID. Plugin specific: if specified ICCID was not found.
- 227 – Wrong activation code. Plugin specific: if specified activation code not found or is already activated.
- 258 – Bundle not found. Plugin specific: if bundle not found for specified device ID.

## replace

Lets to replace the device without losing its history and some of its settings.

Replacement allows you to register a new device with history, sensors (optional), and rules (optional) of the current tracker saved.

**required sub-user rights:** `tracker_configure`.

## parameters

### Important

Because of the variety of tracker models and business applications, there are different ways to register a new tracker in our system. They are called [Registration plugins](#). Each of registration plugins has its own set of additional parameters.

In addition to parameters specified in this section, pass all parameters which are required by the plugin you have chosen. See example below.

Common parameters are:

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	
model	A code of one of the supported models. See <a href="#">tracker/list_models</a> .	string	"pt10"
device_id	<b>Must</b> be specified if device model uses fixed device ID. See <a href="#">tracker/list_models</a> .	string	"4568005588562"
plugin_id	An ID of a registration plugin which will be used to register the device. See <a href="#">Registration plugins</a> .	int	37
send_register_commands	Indicates send or not to send activation commands to a new	boolean	true/false

name	description	type	format
	device (via SMS or GPRS channel). If parameter is not specified or equals <code>null</code> will be used the platform settings. Default: <code>null</code> .		

### examples

In this example we use plugin ID = 37 (see [Plugin description](#)) to replace device with QuecLink GV55Lite. As this device identified by IMEI, we include it as device ID (123451234512346).

Also, we include **phone**, **apn\_name**, **apn\_user**, **apn\_password** of the sim card installed in device. Activation code is not used when replacing a device.

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/replace' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 123456, "plugin_id": 37, "model": "qlgv55lite", "phone": "79123122312", "device_id": "123451234512346", "apn_name": "fast.tmobile.com", "apn_user": "tmobile", "apn_password": "tmobile"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tracker/replace?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456&plugin_id=37&m
```

### response

```
{
  "success": true,
  "value": {
    "id": 833389,
    "label": "Courier",
    "group_id": 0,
    "source": {
      "id": 526383,
      "device_id": "123451234512346",
      "model": "qlgv55lite",
      "blocked": false,
      "tariff_id": 12163,
      "phone": "79123122312",
      "status_listing_id": null,
    }
  }
}
```

```
    "creation_date": "2021-06-03",  
    "tariff_end_date": "2021-06-17"  
  },  
  "clone": false  
}
```

For `tracker` object structure, see [tracker/](#).

## errors

- 7 – Invalid parameters - if fields violate restrictions described above or one of the models is a mobile app.
- 13 – Operation not permitted - if user has insufficient rights.
- 204 – Entity not found - if specified group does not exist. See [group/list](#).
- 220 – Unknown device model - if specified device model does not exist.
- 221 – Device limit exceeded - if device limit set for the user's dealer has been exceeded.
- 222 – Plugin not found - if specified plugin not found or is not supported by device model.
- 223 – Phone number already in use - if specified phone number already used in another device.
- 224 – Device ID already in use - if specified device ID already registered in the system.
- 225 – Not allowed for this legal type - if tariff of the new device is not compatible with user's legal type.
- 226 – Wrong ICCID. Plugin specific: if specified ICCID was not found.
- 258 – Bundle not found. Plugin specific: if bundle not found for specified device ID.
- 266 – Cannot perform action for the device in current status: if the device is not activated yet

## replace\_quick

Replaces a device using only IMEI. Automatic SMS commands will not be sent for an activation. The replacement device must be preconfigured. This API call can be used only for bundles.

**required sub-user rights:** `tracker_configure`.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	
imei	IMEI of the new device	string	"35645587458999"

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/replace_quick' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":  
123456, "imei": "35645587458999"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/replace_quick?  
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456&imei=356455874
```

## response

```
{  
  "success": true,  
  "value": {  
    "id": 123456,  
    "label": "tracker label",  
    "clone": false,  
    "group_id": 167,  
    "avatar_file_name" : "file name",  
    "source": {  
      "id": 234567,  
      "device_id": 99999999888888,  
      "model": "telfmb920",  
      "blocked": false,  
      "tariff_id": 345678,  
      "status_listing_id": null,  
      "creation_date": "2011-09-21",  
      "tariff_end_date": "2016-03-24",  
      "phone" : "71234567890"  
    },  
    "tag_bindings": [{  
      "tag_id": 456789,  
      "ordinal": 4  
    }]  
  }  
}
```

For `tracker` object structure, see [tracker/](#).

#### errors

- 7 – Invalid parameters - if fields violate restrictions described above or one of the models is a mobile app.
- 13 – Operation not permitted - if user has insufficient rights.
- 201 – Not found in the database - if there is no bundle with such IMEI.
- 204 – Entity not found - if specified group does not exist.
- 220 – Unknown device model - if specified device model does not exist.
- 221 – Device limit exceeded - if device limit set for the user's dealer has been exceeded.
- 222 – Plugin not found - if specified plugin not found or is not supported by device model.
- 223 – Phone number already in use - if specified phone number already used in another device.
- 224 – Device ID already in use - if specified device ID already registered in the system.
- 225 – Not allowed for this legal type - if tariff of the new device is not compatible with user's legal type.
- 226 – Wrong ICCID - if specified ICCID was not found.
- 266 – Cannot perform action for the device in current status: if the device is not activated yet

#### replace\_retry

Resends registration commands to the new device. The panel must have installed SMS gateway.

**required sub-user rights:** `tracker_configure`.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999119



name	description	type	format
apn_name	The name of GPRS APN of this sim card inserted into device.	string	"fast.tmobile.com"
apn_user	The user of GPRS APN of this sim card inserted into device.	string	"tmobile"
apn_password	The password of GPRS APN of the sim card inserted into device.	string	"tmobile"

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/register_retry' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 999119, "apn_name": "fast.tmobile.com", "apn_user": "tmobile", "apn_password": "tmobile"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/register_retry?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=999119&apn_name=fast.
```

## response

```
{
  "success": true,
  "value": {
    "id": 123456,
    "label": "tracker label",
    "clone": false,
    "group_id": 167,
    "avatar_file_name" : "file name",
    "source": {
      "id": 234567,
      "device_id": 99999999888888,
      "model": "telfmb920",
      "blocked": false,
      "tariff_id": 345678,
      "status_listing_id": null,
      "creation_date": "2011-09-21",
      "tariff_end_date": "2016-03-24",
      "phone" : "+71234567890"
    },
    "tag_bindings": [{
      "tag_id": 456789,
      "ordinal": 4
    }
  ]
}
```

```

    }
  }
}

```

For `tracker` object structure, see [tracker/](#).

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 219 – Not allowed for clones of the device - if specified tracker is a clone.
- 214 – Requested operation or parameters are not supported by the device - if device does not have GSM module.
- 242 – Device already connected - if tracker connected to the server.
- 266 – Cannot perform action for the device in current status: if the old device is not activated yet

## send\_command

Sends command to tracker for performing special control, determined with `special_control` field of tracker model.

**required sub-user rights:** `tracker_configure`, `tracker_set_output`.

common command format is:

```

{
  "command": {
    "name": "command name",
    "some_parameter1": 12,
    "some_parameter2": "parameter",
    "special_settings": {
      "type": "settings type",
      "some_field1": 10,
      "some_field2": 32
    }
  }
}

```

- `name` - Command name.
- `some_parameter` - Parameters depend on certain command.

- `special_settings` - optional field. Its structure defined with `special_control` field of tracker model.

Certain commands which can be used is defined with `special_control` field of **tracker model** and corresponds the table below:

special control	available commands
jointech_lock_password	electronic_lock_command, set_special_settings_command
hhd_lock_password	electronic_lock_command, set_special_settings_command
vg_lock_password	electronic_lock_command, set_special_settings_command
any other special control	set_special_settings_command

## command types

### electronic\_lock\_command

This command used to seal/unseal electronic lock.

```
{
  "name": "electronic_lock_command",
  "command_code": "unseal",
  "special_settings": {<special settings JSON object>}
}
```

- `command_code` - [enum](#). Can be "seal" or "unseal".
- `special_settings` - This command is equivalent to API call [tracker/settings/special/update](#).

```
{
  "name": "set_special_settings_command",
  "special_settings": {<special settings JSON object>}
}
```

See [special settings JSON object](#)

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999119
command	Command that will be sent to device. Not Null.	JSON object	See format above

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/send_command' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 999119, "command": {"name": "electronic_lock_command", "command_code": "unseal", "special_settings":{"type":"electronic_lock_password", "password": "345892", "remember_password": true}}}'
```

## response

```
{
  "success": true,
  "list": [{
    "id": 123456,
    "label": "tracker label",
    "clone": false,
    "group_id": 167,
    "avatar_file_name" : "file name",
    "source": {
      "id": 234567,
      "device_id": 1234567890,
      "model": "telfmb920",
      "blocked": false,
      "tariff_id": 345678,
      "status_listing_id": null,
      "creation_date": "2011-09-21",
      "tariff_end_date": "2016-03-24",
      "phone" : "+71234567890"
    },
    "tag_bindings": [{
      "tag_id": 456789,
      "ordinal": 4
    }]
  }]
}
```

For `tracker` object structure, see [tracker/](#).

## errors

[General](#) types only.

## raw\_command/send

Sends the GPRS command to the device, processing it in a protocol-dependent manner beforehand.

Find more information about this API call usage in our [instructions](#).

**required sub-user rights:** `tracker_configure`, `tracker_set_output`.

## parameters

name	description	type
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int
command	Text or hexadecimal representation of the command.	string
type	Optional. <code>text</code> or <code>hex</code> format. Default is <code>text</code> .	string
reliable	Optional. <code>false</code> if the command does not need to be resent when the device is disconnected or if no acknowledgment is received. Default is <code>true</code> .	boolean

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/raw_command/send' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489, "command": "AT+GTRT0=gv200,A,,,,,,0001$", "type": "text"}'
```

## response

```
{
  "success": true
}
```

## errors

- 7 - Invalid parameters.
- 201 - Not found in the database – if there is no tracker with such device ID belonging to authorized user.

## example response with an error:

```
{
  "success": false,
  "status": {
    "code": 7,
    "description": "Invalid parameters"
  },
  "errors": [
    {
      "parameter": "command",
      "error": "Non-hex string"
    }
  ]
}
```

Last update: January 15, 2024







# Alarm mode for tracker

Contains API calls to read and set alarm mode of device.

## API actions

API base path: `/tracker/alarm_mode`.

### read

Gets the state of alarm mode of device.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/alarm_mode/read' \  
  -H 'Content-Type: application/json' \  
  -d '{"tracker_id": 123456, "hash":  
  "a6aa75587e5c59c32d347da438505fc3"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/alarm_mode/read?  
tracker_id=123456&hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{  
  "success": true,  
  "enabled": true  
}
```

- `enabled` - `true` if alarm mode enabled.

## errors

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 214 – Requested operation or parameters are not supported by the device - if device does not support alarm mode.

## set

Changes the state of alarm mode of device. The device must be online.

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199
enabled	<code>true</code> if alarm mode should be enabled.	boolean	true/ false

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/alarm_mode/set' \  
  -H 'Content-Type: application/json' \  
  -d '{"tracker_id": 123456, "enabled": true, "hash":  
  "a6aa75587e5c59c32d347da438505fc3"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/alarm_mode/set?  
tracker_id=123456&enabled=true&hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{ "success": true }
```

## **errors**

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 213 – Cannot perform action: the device is offline - if corresponding tracker is not connected to the server.
- 214 – Requested operation or parameters are not supported by the device - if device does not support alarm mode.
- 219 – Not allowed for clones of the device - if tracker is clone.

Last update: December 26, 2022





# APN settings by tracker ID

This resource contains API call to get APN settings by tracker ID. APN is short of Access Point Name and provides a device with the information needed to connect to wireless service.

## API actions

API base path: `/tracker/apn_settings`.

### read

Gets the APN name/user/password and mobile operator of device by a `tracker_id`.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/apn_settings/read' \
  -H 'Content-Type: application/json' \
  -d '{"tracker_id": 123456, "hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/apn_settings/read?
tracker_id=123456&hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{
  "success": true,
  "value": {
    "name": "fast.tmobile.com",
    "user": "tmobile",
    "password": "tmobile"
```

```
}  
}
```

#### **errors**

- 201 – Not found in the database - if tracker or APN settings not found.
- 208 – Device blocked.
- 214 – Requested operation not supported by the device - if the tracker does not have a GSM module or uses a bundled SIM card, the number of which is hidden from the user.

Last update: December 26, 2022







# Avatar for the tracker

Contains API call to upload avatar for the tracker.

## API actions

API base path: `/tracker/avatar`.

### upload

Uploads avatar image for specified tracker. Then it will be available from `https://api.navixy.com/v2/[api_static_path]/tracker/avatars/<file_name>` e.g.  
`https://api.navixy.com/v2/static/tracker/avatars/abcdef123456789.png`.

**required sub-user rights:** `tracker_update`.

**MUST** be a POST multipart request (multipart/form-data), with one of the parts being an image file upload (with the name "file").

File part **mime** type must be one of (see: [source:api-server/src/main/java/com/navixy/common/util/ImageFormats.java ImageFormats.IMAGE\_FORMATS]):

- `image/jpeg`
- `image/pjpeg`
- `image/png`
- `image/gif`
- `image/webp`

## parameters

name	description	type
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int
file	image file.	string
redirect_target	Optional. URL to redirect If <code>redirect_target</code> passed return redirect to <code>?response=</code> .	URL

## response

```
{
  "success": true,
  "value": "file name"
}
```

- `value` - avatar file name.

## errors

- 201 – Not found in the database - when tracker with a `tracker_id` not found in the database.
- 208 – Device blocked.
- 233 – No data file - if file part not passed.
- 234 – Invalid data format - if passed file with unexpected mime type.
- 254 – Cannot save file - on some file system errors.

Last update: December 26, 2022





# Chat

API calls to work with chat module. A chat module allows remote employees and supervisors quickly exchange their ideas and feedback, as well as helps employers to boost team culture. This can prove to be a useful communication tool.

## API actions

API base path: `/tracker/chat`.

### list

Gets a list of chat messages.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199
from	Optional. Start date/time of searching. Default value is now minus 7 days.	date/ time	yyyy-MM-dd HH:mm:ss
to	Optional. End date/time for searching. Default value is now.	date/ time	yyyy-MM-dd HH:mm:ss
limit	Optional. Limit of messages in list. Default and max limit is 1024.	int	1024
ascending	Optional. Ascending order direction from the first message to last. Default value is <code>true</code> .	boolean	true/false

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/chat/list' \
  -H 'Content-Type: application/json' \
  -d '{"tracker_id": 123456, "hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/chat/list?
tracker_id=123456&hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [{<message1>}, {<message2>}]
}
```

- `list` - array of messages.

Where **message** object is:

```
{
  "id": 1,
  "submit_time": "2014-04-15 09:02:24",
  "update_time": null,
  "text": "text of message",
  "type": "INCOMING",
  "status": "PENDING",
  "employee_id": 123456
}
```

- `submit_time` - time when the message submitted.
- `update_time` - delivering time for outgoing messages.
- `type` - INCOMING or OUTGOING.
- `status` - PENDING or DELIVERED.
- `employee_id` - optional, nullable employee identifier.

## errors

- 201 – Not found in the database (if there is no tracker with such ID belonging to authorized user).
- 208 – Device blocked (if tracker exists but was blocked due to tariff restrictions or some other reason).
- 214 – Requested operation or parameters are not supported by the device.

- 236 – Feature unavailable due to tariff restrictions (if one of the trackers has tariff without "chat" feature).

## mark\_read\_all

Marks all incoming chat messages as read for all or for given user trackers.

### parameters

name	description	type	format
trackers	Optional array of IDs of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int array	[999199, 999919]

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/chat/mark_read_all' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tracker/chat/mark_read_all?
hash=a6aa75587e5c59c32d347da438505fc3
```

### response

```
{ "success": true }
```

### errors

- 201 – Not found in the database.

## mark\_read

Marks incoming chat message as read by `message_id` or array of `message_ids`.



## parameters

name	description	type	format
message_id	ID of incoming message.	int	123
message_ids	IDs of incoming messages.	int array	[123,213]

Use only one parameter.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/chat/mark_read' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "message_id": 123}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/chat/mark_read?
hash=a6aa75587e5c59c32d347da438505fc3&message_id=123
```

## response

```
{ "success": true }
```

## errors

- 201 – Not found in the database.

## send

Sends chat message to a specified tracker.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
message	Message text, not null, max size - 20000.	string	

name	description	type	format
			"Hello World"

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/chat/send' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 123456, "message": "Hello World"}'
```

## response

```
{
  "success": true,
  "id": 222
}
```

- `id` - ID of the submitted message.

## errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 214 – Requested operation or parameters are not supported by the device.
- 236 – Feature unavailable due to tariff restrictions - if one of the trackers has tariff with disabled reports (`"has_reports"` is `false`).

## broadcast

Sends chat message to specified trackers.

### parameters

name	description	type	format
trackers	Array of IDs of the tracker (aka "object_id"). Tracker must belong to	int array	[999199, 999919]

name	description	type	format
	authorized user and not be blocked. Max size - 300.		
message	Message text, not null, max size - 20000.	string	"Hello World"

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/chat/broadcast' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "trackers": [999199, 991999], "message": "Hello World"}'
```

## response

```
{
  "success": true,
  "sent_to": [14],
  "not_sent_to": [5234]
}
```

- `sent_to` - list of tracker IDs to whom the message sent.
- `not_sent_to` - list of tracker IDs, who failed to send the message.

## errors

- 217 – The list contains non-existent entities – if one of the specified trackers does not exist, is blocked or doesn't have required tariff features.
- 221 – Device limit exceeded - if device limit set for the user's dealer has been exceeded.

## updated/list

Gets date-times of last messages in chat of trackers.

### parameters

name	description	type	format
trackers		int array	[999199, 999919]

name	description	type	format
	Array of IDs of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked. Max size - 300.		

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/chat/updated/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "trackers": [999199, 991999]}'
```

## response

```
{
  "success": true,
  "value": {
    "101": "2016-02-29 00:23:00",
    "122": "2017-02-28 00:23:00"
  }
}
```

- `value` - map of tracker IDs to date-times.

## errors

- 217 – The list contains non-existent entities – if one of the specified trackers does not exist, is blocked or doesn't have required tariff features.
- 221 – Device limit exceeded - if device limit set for the user's dealer has been exceeded.

## unread/count

Gets count of user's unread chat messages grouped by tracker ID.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/chat/unread/count' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/chat/unread/count?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "value": {
    "1": 123,
    "2": 321
  }
}
```

- `value` - map of tracker IDs to counts.

## errors

- 236 – Feature unavailable due to tariff restrictions - if there is no tracker which has a tariff with "chat" feature.

Last update: December 26, 2022





# Contact

## Deprecated

This API action deprecated and should not be used.

API call to get user's trackers with special grouping by "contacts"

## API actions

API base path: `/tracker/contact`.

### list

Gets all user's trackers with special grouping by "contacts".

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/contact/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tracker/contact/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

### response

```
{
  "success": true,
  "contacts": [{<contact1>}, {<contact n>}],
  "trackers": [{<tracker1>}, {<tracker n>}]
}
```

- `contacts` - all established contacts.
- `trackers` - normal trackers belonging to current user.

where `contact` object is:



```
{
  "user_id": 12059,
  "first_name": "Adam",
  "middle_name": "James",
  "last_name": "Williams",
  "trackers": [{<tracker1>}, {<tracker n>}]
}
```

- `user_id` - ID of the user with which "contact" is established.
- `trackers` - trackers belonging to "contact" which locations shared with current user. Click to see descriptions of type [tracker](#).

#### errors

- 201 – Not found in the database.

Last update: December 26, 2022





# Counters

This resource contains counter specific actions

Find information on how to get counters data [here](#).

## Resource specific actions

Actions with counter entities:

- [/tracker/counter/read](#)
- [/tracker/counter/update](#)

Actions with counter values:

- [/tracker/get\\_counters](#)
- [/tracker/counter/value/get](#)
- [/tracker/counter/value/list](#)
- [/tracker/counter/value/set](#)
- [/tracker/counter/data/read](#)

## read

Reads counter of passed `type`.

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
type	Counter type. One of [ "odometer", "fuel_consumed", "engine_hours" ].	enum	"odometer"

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/counter/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "type": "odometer"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/counter/read?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456&type=odometer
```

## response

```
{
  "success": true,
  "value": {
    "id": 111,
    "type": "odometer",
    "multiplier": 1.0
  }
}
```

## errors

- 204 - Entity not found – if there is no tracker with such ID belonging to authorized user.
- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions or some other reason.
- 219 - Not allowed for clones of the device – if specified tracker is a clone.

## update

Updates counter of passed `type`.

**required sub-user rights:** `tracker_update`.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456

name	description	type	format
type	Counter type. One of [ "odometer", "fuel_consumed", "engine_hours" ].	enum	"odometer"
multiplier	A new value of counter multiplier.	float	1.34
sensor_id	ID of the sensor, which must be used as the source of odometer data (in case when parameter "type" equals "odometer"). If "type" is not "odometer", "sensor_id" must be null.	int	123

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/counter/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "type": "odometer", "multiplier": 3.14, "sensor_id": 1234}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/counter/update?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456&type=odometer&
```

## response

```
{ "success": true }
```

## errors

- 8 - Queue service error, try again later – cannot set counter value, try later.
- 204 - Entity not found – if there is no tracker with such ID belonging to authorized user.
- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions, or some other reason.
- 219 - Not allowed for clones of the device – if specified tracker is a clone.
- 7 - Invalid parameters –
  - if type is not "odometer" and sensor\_id is not null.
  - if sensor with specified sensor\_id is not a metering sensor.
  - if sensor with specified sensor\_id belongs to another tracker.

- if `sensor_id` is negative.
- if sensor with such a `sensor_id` is not exists.
- if type value is not one of list above.

## get\_counters

Gets last values of the tracker's counters.

### parameters

name	description	type	format
tracker_id	Tracker ID (aka "object_id").	int	999119

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/get_counters' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tracker/get_counters?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

### response

```
{
  "success": true,
  "user_time": "2014-07-09 07:50:58",
  "list": [
    {
      "type": "odometer",
      "value": 100500.1,
      "update_time": "2014-03-06 13:57:00"
    }
  ]
}
```

- `user_time` - date/time. Current time in user's timezone.
- `list` - array of counter value objects.
  - `type` - enum. One of predefined semantic counter types (see below).

- `value` - double. Counter value.
- `update_time` - date/time. Date and time when the data updated.

List of counter types:

- `odometer` - odometer.
- `fuel_consumed` - total fuel consumed.
- `engine_hours` - engine hours.

#### errors

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

## value/get

Gets actual value of specified `type` of sensor.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
type	Counter type. One of ["odometer", "fuel_consumed", "engine_hours"].	enum	"odometer"

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/counter/value/get' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "type": "odometer"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/counter/value/get?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456&type=odometer
```



## response

```
{
  "success": true,
  "value": 18.9
}
```

- `value` - float. The last value of counter.

## errors

- 204 - Entity not found – if there is no tracker with such ID belonging to authorized user, counter does not exist or there are no values yet. use `/tracker/counter/set` to create new counter (if not exist) and save some value.
- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions or some other reason.

## value/list

Get actual values for counters of passed `type` and `trackers`.

## parameters

name	description	type	format
trackers	List of the tracker's ID belonging to authorized user.	int array	[123456, 234567]
type	Counter type. One of ["odometer", "fuel_consumed", "engine_hours"].	enum	"odometer"

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/counter/value/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "trackers": [123456, 234567], "type": "odometer"}'
```

## response

```
{
  "success": true,
  "value": {
    "14": 18.9
  }
}
```

```
}  
}
```

- `value` - a map with tracker's IDs as keys.

## errors

- 204 - Entity not found – if one of the specified counter does not exist or there are no values yet. Use `/tracker/counter/set` to create new counter (if not exist) and save some value.
- 217 - List contains nonexistent entities – if one of the specified trackers does not exist or is blocked.

## value/set

Creates new counter of passed `type` (if not) and update its `value`.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
type	Counter type. One of [ "odometer", "fuel_consumed", "engine_hours" ].	enum	"odometer"
value	A new value of counter.	float	233.21

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/counter/value/set'  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":  
123456, "type": "odometer", "value": 233.21}'
```

## response

```
{ "success": true }
```

## errors

- 8 - Queue service error, try again later - can't set counter value, try later.
- 204 - Entity not found – if there is no tracker with such ID belonging to authorized user.
- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions or some other reason.
- 219 - Not allowed for clones of the device – if specified tracker is a clone.

## data/read

Returns counter values for a period.

### parameters

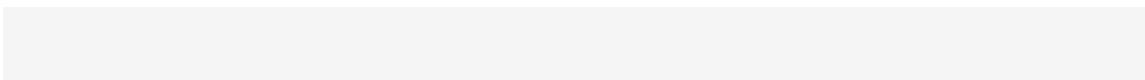
name	description	type	format
tracker_id	Tracker ID (aka "object_id").	int	123456
type	Counter type. One of [ "odometer", "fuel_consumed", "engine_hours" ].	enum	"odometer"
from	Requested period start.	date/ time	"2021-02-25 12:21:17"
to	Requested period end.	date/ time	"2021-03-25 12:21:17"

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/counter/data/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "type": "odometer", "from": "2021-02-01 00:00:00", "to": "2021-02-01 03:00:00"}'
```

## response



```
{
  "success": true,
  "list": [{
    "value": 3835.52,
    "update_time": "2021-02-01 02:52:55"
  }, {
    "value": 3835.7,
    "update_time": "2021-02-01 02:57:18"
  }]
}
```

## errors

- 204 - Entity not found – if there is no tracker or counter belonging to authorized user.
- 211 - Requested time span is too big – if interval between "from" and "to" is too big (maximum value specified in API config)
- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions or some other reason.
- 7 - Invalid parameters –
  - if `from` is after `to`;
  - if between `from` and `to` more than 31 days.

Last update: August 1, 2023





# Datalogger

API call for uploading datalogger information.

## API actions

API base path: `/tracker/datalogger`.

### upload

Uploads track data for specified tracker. Tracker must be a datalogger.

**MUST** be a POST multipart request (multipart/form-data), with one of the parts being a CSV file upload (with the name "file").

#### parameters

name	description	type
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int
file	A CSV file upload containing datalogger track data.	file

#### response

```
{ "success": true }
```

#### errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 219 – Not allowed for clones of the device - if tracker is clone.
- 233 – No data file - if file part is missing.
- 214 – Requested operation or parameters are not supported by the device - if specified tracker is not datalogger.

Last update: December 26, 2022







# Assigning employee to tracker

Allows assigning employee ("driver") to a device. Also, read who is on a vehicle now, hardware key and when, where was assigned.

## API actions

API base path: `/tracker/employee`.

### assign

Assigns another employee ("driver") to the tracker.

**required sub-user rights:** `employee_update`. **required tariff feature:** `app_fleet`.

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
new_employee_id	ID of the new employee.	int	12345

### examples

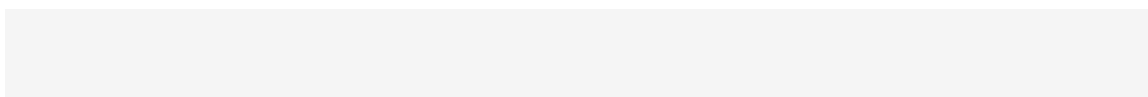
#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/employee/assign' \  
  -H 'Content-Type: application/json' \  
  -d '{"tracker_id": 123456, "new_employee_id": 12345, "hash":  
  "a6aa75587e5c59c32d347da438505fc3"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tracker/employee/assign?  
tracker_id=123456&new_employee_id=12345&hash=a6aa75587e5c59c32d347da43
```

### response



```
{
  "success": true
}
```

## errors

- 201 – Not found in the database - if there is no tracker or employee with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 263 – No change needed, old and new values are the same - if new employee matches a currently assigned employee.

## read

Requests to read the current employee (driver) assigned to tracker, and when it was assigned.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/employee/read' \
-H 'Content-Type: application/json' \
-d '{"tracker_id": 123456, "hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/employee/read?
tracker_id=123456&hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "current": {
    "id": 1,
    "icon_id": 55,
```

```

    "tracker_id": 560,
    "first_name": "John",
    "middle_name": "M",
    "last_name": "Johnson",
    "email": "",
    "phone": "",
    "driver_license_number": "34534545",
    "driver_license_cats": "AB, sgeg",
    "driver_license_issue_date": "2005-06-04",
    "driver_license_valid_till": "2015-06-04",
    "hardware_key": "ab8def",
    "department_id": null,
    "location": {
      "lat": 0.0,
      "lng": 0.0,
      "address": ""
    }
  },
  "last_change": {
    "old_employee_id": null,
    "new_employee_id": 1,
    "location": {
      "lat": 11.0,
      "lng": 22.0,
      "address": "Haraze-Mangueigne"
    },
    "changed": "2016-11-17 17:01:20",
    "origin": "tracker",
    "hardware_key": "ab8def"
  }
}

```

- `current` - current employee (driver) info, standard employee object, can be `null`.
- `last_change` - information about the employee's last assignment, can be `null`.
- `old_employee_id` - deprecated. Always `null`.
- `new_employee_id` - ID of an employee assigned to the tracker. Can be `null`.
- `location` - an address where it was. Can be `null`.
  - `lat` - latitude.
  - `lng` - longitude.
  - `address` - an address where it was. Can be `null`.
- `origin` - supervisor (if the assignment was made through the [API](#)) or `tracker` (if the assignment was made through the hardware/driver key).
- `hardware_key` - hardware key used to change employee.

## errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.

- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

Last update: December 4, 2023







# Engine immobilizer

API calls to read the state of immobilizer and to set the new state. Engine immobilizer is an electronic security device fitted to a motor vehicle that prevents the engine from running unless it must run. This prevents the vehicle from being "hot wired" after entry has been achieved and thus reduces motor vehicle theft. This API call allows manipulating with immobilizer state.

## API actions

API base path: `/tracker/engine_immobilizer`.

### read

Requests to read the state of engine immobilizer.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/engine_immobilizer/read' \
  -H 'Content-Type: application/json' \
  -d '{"tracker_id": 123456, "hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/engine_immobilizer/read?
tracker_id=123456&hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{
  "success": true,
```

```
"enabled": true
}
```

- `enabled` - boolean. `true` if engine immobilizer enabled.

## errors

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 214 – Requested operation or parameters are not supported by the device - if device does not support alarm mode.

## set

Requests to change the engine immobilizer state of the device. The device must be online.

**required sub-user rights:** `tracker_set_output`.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
enabled	<code>true</code> if immobilizer should be enabled.	boolean	true/ false

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/engine_immobilizer/set' \
  -H 'Content-Type: application/json' \
  -d '{"tracker_id": 123456, "enabled": true, "hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/engine_immobilizer/set?
tracker_id=123456&enabled=true&hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{ "success": true }
```

## errors

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 213 – Cannot perform action: the device is offline - if corresponding tracker is not connected to the server.
- 214 – Requested operation or parameters are not supported by the device - if device does not support alarm mode.
- 219 – Not allowed for clones of the device - if tracker is clone.

Last update: December 26, 2022





# Group

Contains group object structure and API calls to interact with them. Tracker group used to organize trackers in user interface. Currently, its function is purely visual.

Group object structure:

```
{
  "id": 167,
  "title": "Main office",
  "color": "FF6DDC"
}
```

- `id` - int. Group ID. Used to reference group in objects and API calls. Read-only, assigned automatically by the server.
- `title` - string. User-specified group title, 1 to 60 printable characters, e.g. "Employees".
- `color` - string. Group color in web format (without #), e.g. "FF6DDC". Determines the color of tracker markers on the map.

## API actions

API base path: `/tracker/group`.

### assign

Assigns multiple trackers to the specified group.

**required sub-user rights:** `admin` (available only to master users).

### parameters

name	description	type	format
id	Group ID, or 0 if trackers should be removed from any group.	int	167

name	description	type	format
trackers	Array of IDs of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int array	[ 999199, 999919]

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/group/assign' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "trackers": [999199, 991999], "id": 167}'
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if no group found with the specified ID (or group belongs to another user).
- 217 - List contains nonexistent entities – if one or more of tracker IDs belong to nonexistent tracker (or to a tracker belonging to different user).

## create

Creates a new empty group.

**required sub-user rights:** `admin` (available only to master users).

## parameters

name	description	type	format
title	Ser-specified group title, 1 to 60 printable characters.	string	"Employees"
color	Group color.	string	"FF6DDC"

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/group/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "title": "Employees", "color": "FF6DDC"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/group/create?
hash=a6aa75587e5c59c32d347da438505fc3&title=Employees&color=FF6DDC
```

## response

```
{
  "success": true,
  "id": 222
}
```

- `id` - int. An ID of created group, e.g. 222.

## errors

[General](#) types only.

## delete

Deletes group with the specified ID. The group must belong to authorized user. All trackers from this group will be assigned to default group (0).

**required sub-user rights:** `admin` (available only to master users).

## parameters

name	description	type	format
id	ID of group to delete.	int	167



## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/group/delete' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "id": 167}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/group/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&id=167
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if no group found with the specified ID (or group belongs to another user).

## list

Gets all user tracker groups. There is always "default" unnamed group with ID = 0. It cannot be modified, deleted, and is not returned by this API call.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/group/list' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/group/list?  
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{  
  "success": true,  
  "list": [  
    {  
      "title": "test",  
      "color": "FF6DDC",  
      "id": 129301  
    }  
  ]  
}
```

```
]
}
```

## errors

[General](#) types only.

## update

Updates specified tracker group. Group must belong to the authorized user.

**required sub-user rights:** `admin` (available only to master users).

## parameters

name	description	type	format
id	ID of group to update.	int	167
title	Ser-specified group title, 1 to 60 printable characters.	string	"Employees"
color	Group color.	string	"FF6DDC"

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/group/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "id": 167,
    "title": "Employees", "color": "FF6DDC"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/group/update?
hash=a6aa75587e5c59c32d347da438505fc3&id=167&title=Employees&color=FF6
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if no group found with the specified ID (or group belongs to another user).

Last update: December 26, 2022





# LED

API calls to get and update LED state of the tracker. LED switch should be available for the device.

## API actions

API base path: `/tracker/led`.

### read

Gets LED status for the specified tracker.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/led/read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":  
265489}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/led/read?  
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

#### response

```
{  
  "success": true,  
  "value": true  
}
```

- `value` - boolean. LED status, `true` - ON, `false` - OFF.

## errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 214 – Requested operation or parameters are not supported by the device.

## update

Switches LED state for a specified tracker.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199
value	The new LED state, <code>true</code> – ON, <code>false</code> – OFF.	boolean	true/ false

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/led/update' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":  
265489, "value": true}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/led/update?  
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489&value=true
```

## response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.

- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 214 – Requested operation or parameters are not supported by the device.

Last update: December 26, 2022







# Geo Links info

API calls for working with Geo Links. These are special sessions to share the location of mobile objects. This is a new API replacing [Weblocator](#).

## API actions

API path: `/tracker/location/link`.

### create

Creates new Geo Link.

**required sub-user rights:** `weblocator_session_create`.

### parameters

name	description	type	restrictions
lifetime	Start and end of the session.	JSON object	Optional.
description	Link's description.	string	Only printable characters. Max length: 255.
trackers	List of tracker IDs with parameters for each tracker.	array of JSON objects	Allowed length 1 to 100.
params	Link parameters.	JSON object	

### LIFETIME OBJECT

```
{
  "from": "2024-01-29 01:00:00", // optional
  "to": "2024-01-30 01:00:00" // optional
}
```

## TRACKER OBJECT

```
{
  "alias": "John Doe", // optional
  "tracker_id": 14,
  "params": {
    "object_data": ["speed", "address"] // speed, address,
    movement_status, connection_status, driver_name, driver_phone,
    vehicle_label, vehicle_reg_number
  }
}
```

## PARAMS OBJECT

```
{
  "bounding_zone_ids": [123,...], // 0..100 zone IDs
  "bounding_mode": "inside", // or outside, optional when no
  bounding zones
  "place_ids": [234,...], // 0..100 place IDs
  "shorten_url": false, // optional, false by default
  "display_options": {
    "map": "roadmap", // or satellite, hybrid, or any other
  available map
    "autoscale": false, // optional, true by default
    "show_icons": false, // optional, false by default
    "show_driver_info": false, // optional, false by default
    "show_vehicle_info": false, // optional, false by default
    "trace_duration": 30 // in seconds, optional, 5 by default
  }
}
```

## example

### cURL

```
curl -X POST "https://api.navixy.com/v2/tracker/location/link/create" \
  -H "Content-Type: application/json" \
  --data-binary @- << EOF
{
  "hash": "a6aa75587e5c59c32d347da438505fc3",
  "lifetime": {
    "from": "2024-01-29 01:00:00",
    "to": "2024-01-30 01:00:00"
  },
  "description": "One tracker link",
  "trackers": [
    {
      "alias": "John Doe",
      "tracker_id": 14,
      "params": {
        "object_data": ["speed", "address"]
      }
    }
  ],
  "params": {
    "bounding_zone_ids": [123, 234],
    "bounding_mode": "inside",
    "place_ids": [987, 654],
    "display_options": {
      "map": "roadmap",
      "autoscale": false,
      "show_icons": false,
      "show_driver_info": false,
      "show_vehicle_info": false,
      "trace_duration": 30
    }
  }
}
EOF
```

## response

```
{
  "success": true,
  "value": 104
}
```

## errors

- 13 – Operation not permitted – if a user has insufficient rights.
- 204 – Entity not found – if one or more of zones or places are not found.
- 217 – List contains nonexistent entities – if one or more of tracker IDs belong to nonexistent tracker (or to a tracker belonging to different user).

- 236 – Feature unavailable due to tariff restrictions – if there is at least one tracker without `weblocator` tariff feature.
- 268 – Link cannot be created due to quota violation.

## update

Updates Geo Link.

### parameters

name	description	type	restrictions
id	Session ID.	int	
lifetime	Optional. Start and end of the session.	JSON object	Optional.
description	Link's description.	string	Only printable characters. Max length: 255.
trackers	List of tracker IDs with parameters for each tracker.	array of JSON objects	Allowed length 1 to 100.
params	Link parameters.	JSON object	

## example

### cURL

```
curl -X POST "https://api.navixy.com/v2/tracker/location/link/
update" \
-H "Content-Type: application/json" \
--data-binary @- << EOF
{
  "hash": "a6aa75587e5c59c32d347da438505fc3",
  "id": 104,
  "lifetime": {
    "from": "2024-01-29 01:00:00",
    "to": "2024-01-30 01:00:00"
  },
  "description": "One tracker link",
  "trackers": [
    {
      "alias": "John Doe",
      "tracker_id": 14,
      "params": {
        "object_data": ["speed", "address"]
      }
    }
  ],
  "params": {
    "bounding_zone_ids": [123, 456],
    "bounding_mode": "inside",
    "place_ids": [987, 654],
    "shorten_url": false,
    "display_options": {
      "map": "roadmap",
      "autoscale": false,
      "show_icons": false,
      "show_driver_info": false,
      "show_vehicle_info": false,
      "trace_duration": 30
    }
  }
}
EOF
```

## response

```
{
  "success": true
}
```

## errors

- 13 – Operation not permitted – if a user has insufficient rights.
- 201 – Not found in the database – if link with such an ID does not exist or does not belong to current user.

- 204 – Entity not found – if one or more of zones or places are not found.
- 217 – List contains nonexistent entities – if one or more of tracker IDs belong to nonexistent tracker (or to a tracker belonging to different user).
- 236 – Feature unavailable due to tariff restrictions – if there is at least one tracker without `weblocator` tariff feature.

## status/change

Lets to activate and deactivate a link.

### parameters

name	description	type
id	Session ID.	int
is_active	If <code>false</code> , a link is deactivated	boolean

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/location/link/status/change' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "id": 104, "is_active": false}'
```

### response

```
{
  "success": true
}
```

### errors

- 201 – Not found in the database – if link with such an ID does not exist or does not belong to current user.

## read

Returns a link with a specified ID.



parameters

name	description	type
id	Session ID.	int

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/location/link/read' \
-H 'Content-Type: application/json' \
-d '{"hash":"a6aa75587e5c59c32d347da438505fc3","id":103}'
```

HTTP GET

```
https://api.navixy.com/v2/tracker/location/link/read?
hash=a6aa75587e5c59c32d347da438505fc3&id=103
```

response

```
{
  "success": true,
  "value": {
    "create_date": "2024-01-29 03:00:00",
    "creator_id": 3,
    "description": "Another one tracker",
    "enabled": true,
    "hash": "700d4a5400000000600d4a5400000103",
    "id": 103,
    "lifetime": {
      "from": "2024-01-30 00:00:00",
      "to": "2024-01-30 03:00:00"
    },
  },
  "params": {
    "bounding_mode": "outside",
    "bounding_zone_ids": [
      51
    ],
    "display_options": {
      "autoscale": false,
      "map": "osm",
      "show_driver_info": false,
      "show_icons": false,
      "show_vehicle_info": false,
      "trace_duration": 0
    },
    "place_ids": null
  },
  "trackers": [
    {
      "alias": "Jane Doe",

```

```

    "params": {
      "object_data": []
    },
    "tracker_id": 15
  }
]
}

```

## errors

- 201 – Not found in the database – if link with such an ID does not exist or does not belong to current user.

## list

Returns a list of a user's links.

## parameters

name	description	type
filter	Optional. Filter for all fields. If used with conditions, both filter and conditions must match for every returned links.	string
conditions	Optional. Search conditions to apply to list. Array of search conditions, see <a href="#">Search conditions</a> . Possible fields listed below.	array of objects
offset	Optional. Offset, default is 0.	int
limit	Optional. Limit, default is 10,000.	int
sort	Optional. Each option is a pair of field name and sorting direction, e.g. [ "creator=asc", "id=desc" ]. Possible fields listed below.	string array

## CONDITION FIELDS

- `trackers` – labels of all trackers
- `aliases` – aliases of all trackers
- `creator` – full name of creator (only for master user)
- `description`

## SORT FIELDS

- `id`
- `create_date`
- `expire_date`
- `trackers` – labels of all trackers
- `aliases` – aliases of all trackers
- `status` – enabled < inactive < expired < disabled
- `creator` – full name of creator (only for master user)
- `description`
- If no `sort` param is specified, then `sort` option will be "id=asc".

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/location/link/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "offset": 0, "limit": 1000}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/location/link/list?
hash=a6aa75587e5c59c32d347da438505fc3&offset=0&limit=1000
```

## response

```
{
  "success": true,
  "list": [
    {
      "create_date": "2024-01-29 03:00:00",
      "creator_id": 3,
      "description": "Another one tracker",
      "enabled": true,
      "hash": "700d4a5400000000600d4a5400000103",
      "id": 103,
      "lifetime": {
        "from": "2024-01-30 00:00:00",
        "to": "2024-01-30 03:00:00"
      },
      "params": {
        "bounding_mode": "outside",
        "bounding_zone_ids": [51],
        "display_options": {
          "autoscale": false,
          "map": "osm",
          "show_driver_info": false,

```

```

        "show_icons": false,
        "show_vehicle_info": false,
        "trace_duration": 0
    },
    "place_ids": null
},
"trackers": [
    {
        "alias": "Jane Doe",
        "params": {
            "object_data": []
        },
        "tracker_id": 15
    }
]
}
]
}

```

## delete

Deletes a link with a specified ID.

### parameters

name	description	type
id	Session ID.	int

### example

#### cURL

```

curl -X POST 'https://api.navixy.com/v2/tracker/location/link/delete' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "id": 103}'

```

### response

```

{
  "success": true
}

```

### errors

- 201 – Not found in the database – if link with such an ID does not exist or does not belong to current user.

Last update: April 8, 2024





# Mobile app register

## Deprecated

This API action deprecated and should not be used.

API call to register a mobile application. Use [tracker/register](#) with `plugin_id` 35.

## API actions

API base path: `/tracker/mobile`.

### register

Registers new mobile client application.

**required sub-user rights:** `tracker_register`.

#### parameters

Part of parameters are registration plugin-specific. See "Registration plugins" section.

Common parameters are:

name	description	type	format
label	User-defined label for this tracker. Must consist of printable characters and have length between 1 and 60.	string	"Courier"
group_id	Tracker group id, 0 if tracker does not belong to any group. The specified group	int	0



name	description	type	format
	must exist. See <a href="#">group/list</a> .		
device_id	<b>Must</b> be specified if device model uses fixed device id. See <a href="#">tracker/list_models</a> .	string	"4568005588562"
send_register_commands	Indicates send or not to send activation commands to device (via SMS or GPRS channel). If parameter is not specified or equals <code>null</code> will be used the platform settings. Default: <code>null</code> .	boolean	true/false

## response

```
{
  "success": true,
  "value": {<tracker>}
}
```

For `tracker` object structure, see [tracker/](#).

## errors

- 13 – Operation not permitted – if user has insufficient rights.
- 204 – Entity not found - if specified group does not exist.
- 221 – Device limit exceeded - if device limit set for the user's dealer has been exceeded.
- 224 – Device ID already in use - if specified device ID already registered in the system.
- 225 – Not allowed for this legal type - if tariff of the new device is not compatible with user's legal type.

Last update: December 26, 2022





# Output control

API calls for output control. Some device models work with `set_all` and some with `set` calls.

## API actions

API base path: `/tracker/output`.

### set\_all

Request to change the states of all digital outputs of the device. The device must be online.

**required sub-user rights:** `tracker_set_output`.

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199
outputs	Array of desired states of all digital outputs, e.g. <code>[true, true, false]</code> means output 1 is on, output 2 is on, output 3 is off.	array of boolean	<code>[true, true, false]</code>

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/output/set_all' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489, "outputs": [true, true, false]}'
```

### response

```
{ "success": true }
```

## errors

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 213 – Cannot perform action: the device is offline - if corresponding tracker is not connected to the server.
- 214 – Requested operation or parameters are not supported by the device - if device does not support batch mode, or has a different number of outputs.
- 219 – Not allowed for clones of the device - if tracker is clone.

## set

Request to change the state of the specified digital output of the device. The device must be online.

**required sub-user rights:** `tracker_set_output`.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199
output	The number of the output to control, starting from 1.	int	1
enable	<code>true</code> if the requested output should be enabled, or <code>false</code> if it should be disabled.	boolean	true/ false

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/output/set' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", tracker_id:  
265489, "output": 1, "enable": true}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/output/set?  
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489&output=1&enabl
```

## response

```
{ "success": true }
```

## errors

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 213 – Cannot perform action: the device is offline - if corresponding tracker is not connected to the server.
- 214 – Requested operation or parameters are not supported by the device - if device does not support controlling single output, does not have specified digital output, or the specified output reserved to "engine block" feature. In this case, output cannot be controlled by this command for safety reasons.
- 219 – Not allowed for clones of the device - if tracker is clone.

Last update: December 26, 2022







# Sensor readings

API call to get last values for all metering sensors and state values. Includes CAN, OBD, and fuel.

Described getting data from sensors in our [guides](#).

## API actions

API base path: `/tracker/readings`.

### list

Gets last values for all metering sensors, state values and counters.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199
sensor_type	Optional. If specified, state values and counters will be omitted. Used to filter sensors by type.	string <a href="#">metering sensor type</a> or <a href="#">virtual sensor type</a>	"fuel"
include_components	Optional. Default is <code>true</code> . If set to <code>false</code> , parts of composite sensors will be excluded.	boolean	true

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/readings/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/readings/list?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

## response

```
{
  "success": true,
  "inputs": [
    {
      "value": 5.66,
      "label": "label",
      "units": "litres",
      "name": "fuel_level",
      "type": "fuel",
      "units_type": "custom",
      "update_time": "2023-06-28 06:05:59"
    }
  ],
  "states": [
    {
      "field": "obd_mil_status",
      "value": 12345.23,
      "update_time": "2023-06-28 06:05:59"
    }
  ],
  "virtual_sensors": [
    {
      "label": "Virtual Ignition",
      "value": "On",
      "type": "virtual_ignition",
      "update_time": "2023-06-28 06:05:59"
    },
    {
      "label": "Hood state",
      "value": "Closed",
      "type": "state",
      "update_time": "2023-06-28 06:05:59"
    }
  ],
  "counters": [
    {
      "type": "odometer",
      "value": 3232.9923342688653,
      "update_time": "2023-06-28 06:05:59"
    }
  ]
}
```

```
]
}
```

- `inputs` - an array of JSON objects containing information about the tracker sensors readings.
  - `value` - float. The value of the sensor.
  - `label` - string. The label of the sensor.
  - `units` - string. The units in which the sensor value is measured.
  - `name` - string. The name of the sensor.
  - `type` - [metering sensor type](#). The type of the sensor.
  - `units_type` - string. The type of the units in which the sensor value is measured.
  - `update_time` - date/time. The time when the sensor value was updated.
- `states` - an array of JSON objects containing information about the tracker state readings.
  - `field` - string. The field name of the state.
  - `value` - can be string, int, float, boolean, or null. The value of the field.
  - `update_time` - date/time. The time when the field value was updated.
- `virtualSensors` - an array of JSON objects containing information about the tracker virtual sensors readings.
  - `value` - float. The value of the virtual sensor.
  - `label` - string. The label of the virtual sensor.
  - `type` - [virtual sensor type](#). The type of the virtual sensor.
  - `update_time` - date/time. The time when the virtual sensor value was updated.
- `counters` - an array of JSON objects containing information about the tracker counter readings.
  - `type` - string. The type of the counter.
  - `value` - float. The value of the counter.
  - `update_time` - date/time. The time when the counter value was updated.

## errors

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

## batch\_list

Gets last values for all metering sensors, state values and counters on multiple trackers.

### parameters

name	description	type	format
trackers	An array of tracker IDs (aka "object_id"). Trackers must belong to authorized user.	int	[999199,991999]
sensor_type	Optional. If specified, state values and counters will be omitted. Used to filter sensors by type.	string <a href="#">metering sensor type</a> or <a href="#">virtual sensor type</a>	"fuel"
include_components	Optional. Default is <code>true</code> . If set to <code>false</code> , parts of composite sensors will be excluded.	boolean	true

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/readings/  
batch_list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "trackers":  
[10181215,10038816]}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tracker/readings/batch_list?  
hash=a6aa75587e5c59c32d347da438505fc3&trackers=[10181215,10038816]
```

### response

```
{  
  "result": {
```

```
"10181215": {
  "inputs": [
    {
      "label": "Tank 1",
      "units": "",
      "name": "lls_level_1",
      "type": "fuel",
      "min_value": 0.0,
      "max_value": 480.0,
      "value": 225.71,
      "units_type": "litre",
      "converted_units_type": null,
      "converted_value": null,
      "update_time": "2023-06-28 06:13:09"
    },
    {
      "label": "Tank 2",
      "units": "",
      "name": "lls_level_6",
      "type": "fuel",
      "min_value": 0.0,
      "max_value": 300.0,
      "value": 113.52,
      "units_type": "litre",
      "converted_units_type": null,
      "converted_value": null,
      "update_time": "2023-05-11 00:35:16"
    },
    {
      "label": "Fuel",
      "units": "",
      "name": "composite",
      "type": "fuel",
      "min_value": 0.0,
      "max_value": 700.0,
      "value": 175.31,
      "units_type": "litre",
      "converted_units_type": null,
      "converted_value": null,
      "update_time": "2023-05-11 00:35:26"
    }
  ],
  "states": [
    {
      "field": "input_status",
      "value": 0,
      "update_time": "2023-06-28 06:13:09"
    },
    {
      "field": "movement_state",
      "value": "parked",
      "update_time": "2023-06-28 06:13:09"
    },
    {
      "field": "tcp_status",
      "value": 2,
      "update_time": "2023-06-28 06:13:57"
    }
  ]
}
```

```

    },
    {
      "field": "output_status",
      "value": 0,
      "update_time": "2023-06-28 06:13:09"
    }
  ],
  "counters": [
    {
      "type": "odometer",
      "value": 3232.9923342688653,
      "update_time": "2023-06-28 06:05:59"
    }
  ]
},
"10038816": {
  "inputs": [],
  "states": [
    {
      "field": "input_status",
      "value": 0,
      "update_time": "2023-06-28 06:13:23"
    },
    {
      "field": "movement_state",
      "value": "parked",
      "update_time": "2023-06-28 06:13:23"
    },
    {
      "field": "output_status",
      "value": 0,
      "update_time": "2023-06-28 06:13:23"
    },
    {
      "field": "tcp_status",
      "value": 2,
      "update_time": "2023-06-28 06:14:07"
    }
  ],
  "counters": [
    {
      "type": "odometer",
      "value": 20854.422727641213,
      "update_time": "2023-06-28 06:12:23"
    }
  ]
}
},
"success": true
}

```

- `inputs` - an array of JSON objects containing information about the tracker sensors readings.
  - `value` - float. The value of the sensor.
  - `label` - string. The label of the sensor.

- `units` - string. The units in which the sensor value is measured.
- `name` - string. The name of the sensor.
- `type` - [metering sensor type](#). The type of the sensor.
- `units_type` - string. The type of the units in which the sensor value is measured.
- `update_time` - date/time. The time when the sensor value was updated.
- `min_value` - float. The minimum value of the sensor.
- `max_value` - float. The maximum value of the sensor.
- `converted_units_type` - string. The type of the units in which the sensor value is converted.
- `converted_value` - float. The converted value of the sensor reading.
- `states` - an array of JSON objects containing information about the tracker state readings.
  - `field` - string. The field name of the state.
  - `value` - can be string, int, float, boolean, or null. The value of the field.
  - `update_time` - date/time. The time when the field value was updated.
- `counters` - an array of JSON objects containing information about the tracker counter readings.
  - `type` - string. The type of the counter.
  - `value` - float. The value of the counter.
  - `update_time` - date/time. The time when the counter value was updated.

## errors

- 217 - List contains nonexistent entities.

Last update: April 24, 2024







# Retranslator

Contains tracker retranslator binding object and API calls to bind/unbind it to tracker or get already binded one.

## Tracker retranslator binding object

```
{
  "retranslator_id": 4548,
  "fake_device_id": "AI568T"
}
```

- `retranslator_id` - int. An ID of the [retranslator](#).
- `fake_device_id` - string. Optional. If this field is set retranslator use it instead of real device ID to forward data.

## API actions

API base path: `/tracker/retranslator`.

### bind

Creates or updates binding.

**required sub-user rights:** `admin` (available only to master users).

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199
retranslator_id	Retranslator ID.	int	123
fake_device_id		string	"AI568T"

name	description	type	format
	Optional. If set the retranslator will use this value instead of real device ID to forward data.		

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/retranslator/bind' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489, "retranslator_id": 123}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/retranslator/bind?hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489&retranslator_i
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found - either tracker or retranslator are not found by provided ID
- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions or some other reason.
- 219 - Not allowed for clones of the device – if tracker is a clone.
- 236 - Feature unavailable due to tariff restrictions – if there are no trackers with "retranslation" tariff feature available.
- 242 - There were errors during content validation – if `fake_device_id` is invalid for the retranslator protocol.

## list

List tracker retranslators bound to tracker with ID= `tracker_id`.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/retranslator/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/retranslator/list?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

## response

```
{
  "success": true,
  "list": [{
    "retranslator_id": 4548,
    "fake_device_id": "AI568T"
  }]
}
```

## errors

- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions, or some other reason.

## unbind

Unbinds a tracker from retranslator.

**required sub-user rights:** `admin` (available only to master users).

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199
retranslator_id	Retranslator ID.	int	123

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/retranslator/unbind' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id": 265489, "retranslator_id": 123}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/retranslator/unbind?hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489&retranslator_i
```

## response

```
{ "success": true }
```

## errors

- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions, or some other reason.
- 219 - Not allowed for clones of the device – if tracker is a clone.

Last update: December 26, 2022







# Trusted number

API calls to interact with list of trusted numbers for trackers.

## API actions

API base path: `/tracker/trusted_number`.

### list

Gets list of trusted numbers for the specified tracker.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/trusted_number/
list' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/trusted_number/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

#### response

```
{
  "success": true,
  "list": ["496156680000", "496156680001"]
}
```

- `list` - List of strings containing trusted phone numbers in the international format without "+" sign.

## errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

## update

Replaces the list of trusted numbers for a specified tracker with the new one.

**required sub-user rights:** `tracker_update`.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	999199
list	Array of phone numbers (10-15 digits) represented as strings.	string array	[ "496156680001", "496156680000" ]

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/trusted_number/
update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
265489, "list": ["496156680001", "496156680000"]}'
```

## response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.

- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

Last update: December 26, 2022





# Unconfirmed commands

API calls for to interact with unconfirmed SMS commands in the queue of the specified tracker.

## API actions

API path: `/tracker/command/unconfirmed.`

### count

Gets number of commands in queue for the specified tracker.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/command/unconfirmed/count' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/command/unconfirmed/count?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456
```

#### response

```
{
  "success": true,
  "count": 0
}
```

- `count` - int. Number of unconfirmed commands in a queue.

## errors

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

## reset

Removes all pending SMS commands from the queue for the specified tracker.

**required sub-user rights:** `tracker_update`.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/command/unconfirmed/reset' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/command/unconfirmed/reset?  
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456
```

## response

```
{ "success": true }
```

## errors

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

Last update: December 26, 2022







# Rule

Contains API calls to interact with tracker's rules. Rules used to set up conditions according to which the system logs the events and sends notifications to user.

Described creation and using rules step-by-step in our [instructions](#).

## Rule object

```
{
  "id": 668054,
  "name": "Lock is opened/closed",
  "type": "locking_unlocking",
  "description": "This rule was created automatically",
  "zone_ids": [ 18928 ],
  "trackers": [ 10029750, 10030168, 10031971 ],
  "primary_text": "Lock is opened",
  "secondary_text": "Lock is closed",
  "param": 0,
  "alerts": {
    "sms_phones": [],
    "phones": [],
    "emails": [],
    "push_enabled": true
  },
  "suspended": false,
  "auto_created": true,
  "schedule": [{
    "type": "weekly",
    "from": {
      "weekday": 1,
      "time": "00:00:00"
    },
    "to": {
      "weekday": 7,
      "time": "23:59:59"
    },
    "interval_id": 48732
  }],
  "extended_params": {
    "emergency": false,
    "zone_limit_inverted": false,
    "private_rule": true
  }
}
```

- `id` - int. An ID of a rule.
- `name` - string. Name of a rule.

- `type` - enum. One of pre-defined types of rules. See [rule types](#).
- `description` - string. Rule's description.
- `zone_ids` - int array. List of geofence IDs.
- `trackers` - int array. List of bound tracker IDs.
- `primary_text` - string. Primary text of rule notification.
- `secondary_text` - string. Secondary text of rule notification.
- `param` - int. A common parameter. See [rule types](#).
- `alerts` - object with destinations for notifications.
  - `sms_phones` - string array. Phones for SMS notifications.
  - `phones` - string array. Phones for voice calls.
  - `emails` - string array. Emails for notifications.
  - `push_enabled` - boolean. If `true` push notifications available.
  - `emergency` - boolean. If `true` notifications will be marked as emergency with color and sound.
- `suspended` - boolean. `true` if the rule suspended.
- `auto_created` - optional, boolean. `true` means that the rule created automatically.
- `shedule` - optional object. The rule will work in specified period.
- `extended_params` - optional. An object specified for concrete rule type. See [rule types](#).
- **`schedule_interval`** is one of:
  - **`weekly_schedule_interval`**

```
{
  "type": "weekly",
  "from": {
    "weekday": 1,
    "time": "00:00:00"
  },
  "to": {
    "weekday": 7,
    "time": "23:59:59"
  },
  "interval_id": 1
}
```

**\* `fixed_schedule_interval`**

```
{
  "type": "fixed",
  "from": "2014-07-09 07:50:58",
  "to": "2014-07-10 07:50:58",
}
```

```
"interval_id": 3
}
```

- `date/time` and `local_time` types described at the [data types description section](#).

## API actions

API base path: `/tracker/rule`.

### bind

Binds rule with `rule_id` to trackers list.

**required sub-user rights:** `tracker_rule_update`.

#### parameters

name	description	type
<code>rule_id</code>	ID of a rule.	int
<code>trackers</code>	IDs of trackers. Trackers which do not exist, owned by other user or deleted ignored without errors.	int array

#### example

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/rule/bind' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "rule_id": 123, "trackers": [265489]}'
```

#### response

```
{ "success": true }
```

#### errors

- 201 - Not found in the database – if rule with `rule_id` does not exist or owned by other user.

## create

Creates rule and scheduled intervals.

**required sub-user rights:** `tracker_rule_update`.

### parameters

Presented parameters are common for all rule types. However, there are specific parameters `primary_text` and `secondary_text` that are described for every rule type if exist in [rule types](#).

name	description	type
name	The name of created rule.	string
description	Rule's description.	string
trackers	List of tracker IDs belong to user for which the rule will work.	int array
zone_ids	List of zones to bind where the rule will work. Leave it empty if rule should work everywhere. Parameter <code>zone_ids</code> is not allowed for rule <code>offline</code> and can't be empty for <code>route</code> and <code>inoutzone</code> rule types.	int array
type	One of pre-defined types of rules. See <a href="#">rule types</a> .	enum
param	A common parameter that responsible for integer conditions. See <a href="#">rule types</a> .	int
alerts	An object with destinations for notifications. Described <a href="#">above</a> .	JSON object
suspended	Starts or stops tracking the rule. <code>true</code> if the rule suspended.	boolean
schedule	An optional object. Configures the time - when the rule works. Described <a href="#">above</a> .	JSON object
extended_params	An optional object. Specified for concrete rule type. See <a href="#">rule types</a> .	JSON object

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/rule/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "rule":
{"description": "", "type": "work_status_change", "primary_text":
"status changed", "secondary_text": "", "alerts": {"push_enabled":
true, "emails": ["example@gmail.com"], "emergency": false,
"sms_phones": ["745494878945"], "phones": []}, "suspended": "",
"name": "Status changing", "trackers": [123456],
"extended_params": {"emergency": false, "zone_limit_inverted":
false, "append_zone_title": "", "status_ids":
[319281,319282,319283]}, "param": "", "schedule": [{"from":
{"weekday": 1, "time": "00:00:00"}, "to": {"weekday": 7, "time":
"23:59:59"}, "type": "weekly"}], "zone_ids": [], "group_id": 1}}'
```

## response

```
{
  "success": true,
  "id": 123
}
```

- `id` - int. An ID of created rule.

## errors

- 204 - Entity not found – when associated zone is not exist.

## delete

Deletes rule with `rule_id` and all related objects from the database.

**required sub-user rights:** `tracker_rule_update`.

## parameters

name	description	type
<code>rule_id</code>	ID of a rule.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/rule/delete' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "rule_id": 123}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/rule/delete?
hash=a6aa75587e5c59c32d347da438505fc3&rule_id=123
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if rule with `rule_id` does not exist or owned by other user.

## list

List tracker rules bound to tracker with an ID= `tracker_id` or all users' tracker rules if `tracker_id` not passed.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/rule/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/rule/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [{
    "id": 667281,
    "name": "Case intrusion",
    "type": "case_intrusion",
    "description": "This rule was created automatically",
    "zone_id": 0,
    "trackers": [10029448, 10030168],
```



```

    "primary_text": "Case is opened",
    "secondary_text": "Case is closed",
    "param": 0,
    "alerts": {
      "sms_phones": [],
      "phones": [],
      "emails": [],
      "push_enabled": true
    },
    "suspended": false,
    "auto_created": true,
    "schedule": [{
      "type": "weekly",
      "from": {
        "weekday": 1,
        "time": "00:00:00"
      },
      "to": {
        "weekday": 7,
        "time": "23:59:59"
      },
      "interval_id": 46892
    }]
  }]
}

```

- `list` - list of rules

## unbind

Unbinds trackers from rule with `rule_id`.

**required sub-user rights:** `tracker_rule_update`.

### parameters

name	description	type
<code>rule_id</code>	ID of a rule.	int
<code>trackers</code>	IDs of trackers. Trackers which do not exist, owned by other user or deleted ignored without errors.	int array

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/rule/unbind' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "rule_id": 123, "trackers": [265489]}'
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if rule with `rule_id` does not exist or owned by other user.

## update

Updates rule and scheduled intervals.

**required sub-user rights:** `tracker_rule_update`.

## parameters

Presented parameters are common for all rules, but there are specific parameters that can be found in [rule types](#).

name	description	type
id	ID of a rule. You can get IDs using the <a href="#">rule/list</a> call.	int
name	The name of created rule.	string
description	Rule's description.	string
zone_ids	List of zones to bind where the rule will work. Leave it empty if rule should work everywhere. Parameter <code>zone_ids</code> is not allowed for rule <code>offline</code> and required for <code>route</code> and <code>inoutzone</code> rule types (there can be exactly one item).	int array

name	description	type
trackers	List of tracker IDs belong to user for which the rule will work.	int array
type	One of pre-defined types of rules. See <a href="#">rule types</a> .	enum
param	A common parameter that responsible for integer conditions. See <a href="#">rule types</a> .	int
alerts	An object with destinations for notifications. Described <a href="#">above</a> .	JSON object
suspended	Starts and stops tracking the rule. <code>true</code> if the rule suspended.	boolean
schedule	An optional object. Configures the time - when the rule works. Described <a href="#">above</a> .	JSON object
extended_params	An optional object. Specified for concrete rule type. See <a href="#">rule types</a> .	JSON object

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/rule/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "rule": {
    "id": 123, "description": "", "type": "work_status_change",
    "primary_text": "status changed", "secondary_text": "", "alerts": {
      "push_enabled": true, "emails": ["example@gmail.com"],
      "emergency": false, "sms_phones": ["745494878945"], "phones": []},
    "suspended": "", "name": "Status changing", "trackers": [123456],
    "extended_params": {"emergency": false, "zone_limit_inverted":
      false, "append_zone_title": "", "status_ids":
      [319281,319282,319283]}, "param": "", "schedule": [{"from":
      {"weekday": 1, "time": "00:00:00"}, "to": {"weekday": 7, "time":
      "23:59:59"}, "type": "weekly"}], "zone_ids": [], "group_id": 1}}'
```

## response

```
{ "success": true }
```

**errors**

- 201 - Not found in the database – if rule is not exists or owned by other user.
- 204 - Entity not found – when new associated zone is not exists.

Last update: December 15, 2023





## Rule types

Rule types with all parameters to create. The rule availability depends on the device, connected and configured equipment and rule integration for it.

### Geofence entrance or exit

A rule that triggers on device entering/exiting created on platform [geofences](#).

#### parameters

name	description	type
type	<code>inoutzone</code> for this rule type.	<a href="#">enum</a>
primary_text	Text of rule notification on entering geofence. It is for <code>inzone</code> event type.	string
secondary_text	Text of rule notification on exiting geofence. It is for <code>outzone</code> event type.	string

#### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
append_zone_title	Show or not the zone labels in a notification text.	boolean

### Parking state detection

A rule that triggers on detection of parking state calculated based on [parking detection settings](#).

### parameters

name	description	type
type	<code>track_change</code> for this rule type.	enum
primary_text	Text of rule notification on parking start. It is for <code>track_end</code> event type.	string
secondary_text	Text of rule notification on parking end. It is for <code>track_start</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean

## Speeding (hardware related)

A rule that triggers on speed exceeding determined by hardware. Based on the configs on the device side.

### parameters

name	description	type
type	<code>over_speed_reported</code> for this rule type.	enum



name	description	type
primary_text	Text of rule notification when speeding detected. It is for <code>over_speed_reported</code> event type.	string

#### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Speeding (platform related)

A rule that triggers on speed exceeding determined by the platform. Based on received speed from device.

#### parameters

name	description	type
type	<code>speedup</code> for this rule type.	enum
param	Speed limit. It is for <code>speedup</code> event type.	int
primary_text	Text of rule notification when speed exceeds the specified <code>param</code> value.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

### Deviation from the route

A rule that triggers on deviations from the route. Only the [route \(sausage\) type](#) geofence may be assigned.

### parameters

name	description	type
type	<code>route</code> for this rule type.	enum
primary_text	Text of rule notification when device outs the route zone. It is for <code>outroute</code> event type.	string

### extended parameters

name	description	type
allow_exit_at_endpoints	If <code>true</code> disables notifications on deviations from the start and end points of a route.	boolean
emergency	If <code>true</code> enables emergency notification.	boolean

name	description	type
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
append_zone_title	Show or not the zone labels in a notification text.	boolean

## Driving time

A rule that triggers when your employee drives more than allowed. The driving time is calculated based on [parking detection settings](#).

### parameters

name	description	type
type	<code>excessive_driving</code> for this rule type.	<a href="#">enum</a>
primary_text	Text of rule notification when driving time exceeded. It is for <code>excessive_driving_start</code> event type.	string
secondary_text	Text of rule notification on driving time exceeding end. It is for <code>excessive_driving_end</code> event type.	string

### extended parameters

name	description	type
max_driving_time	Allowed driving time. How much time your employee can drive a car	int
min_parking_time	Minimum parking time to reset the timer. How much time your employee must wait until he can continue driving	int
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule		boolean

name	description	type
	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Parking time

A rule that triggers when your employee standstill more than allowed. The parking time is calculated based on [parking detection settings](#).

### parameters

name	description	type
type	<code>excessive_parking</code> for this rule type.	enum
primary_text	Text of rule notification when parking time exceeded. It is for <code>excessive_parking</code> event type.	string
secondary_text	Text of rule notification on parking time exceeding end. It is for <code>excessive_parking_finished</code> event type.	string

### extended parameters

name	description	type
max_parking_duration	Allowed parking time. How much time a car can standstill	int
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean

name	description	type
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Task performance

A rule that triggers when assigned to a tracker [task](#) changes its status.

### parameters

name	description	type
type	<code>task_status_change</code> for this rule type.	<a href="#">enum</a>
primary_text	Text of rule notification when task changes its status to a chosen one or form is submitted or resubmitted.	string

### extended parameters

name	description	type
statuses	List of tracked statuses. Possible statuses are "arrived", "done", "delayed", "failed".	string array
on_form_submission	If <code>true</code> form submission will track.	boolean
on_repeated_form_submission	If <code>true</code> form resubmission will track.	boolean
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule		boolean

name	description	type
	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Work status change

A rule that triggers when tracker [work status](#) changes. Choose specific status IDs from a currently assigned to tracker [status listing](#).

### parameters

name	description	type
type	<code>work_status_change</code> for this rule type.	<a href="#">enum</a>
primary_text	Text of rule notification when work status changes to a chosen one. It is for <code>work_status_change</code> event type.	string

### extended parameters

name	description	type
status_ids	List of tracked status IDs. Choose specific status IDs from a currently assigned to tracker <a href="#">status listing</a> .	int array
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule		boolean

name	description	type
	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Excessive idling (hardware related)

A rule that triggers on excessive idling registered by hardware. Based on the configs on the device side.

### parameters

name	description	type
type	<code>idling</code> for this rule type.	enum
primary_text	Text of rule notification when excessive idling detected by device.	string
secondary_text	Text of rule notification when excessive idling end detected by a device.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean

name	description	type
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Excessive idling (platform related)

A rule that triggers on excessive idling registered by the platform. The idling time is calculated based on [parking detection settings](#) and ignition state.

### parameters

name	description	type
type	<code>idling_soft</code> for this rule type.	enum
primary_text	Text of rule notification when excessive idling detected by platform. It is for <code>idle_start</code> event type.	string
secondary_text	Text of rule notification when excessive idling end detected by platform. It is for <code>idle_end</code> event type.	string
param	Idle duration to send notification.	int

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title		boolean



name	description	type
	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	

## Fuel level change

Rule triggered by a drastic change in fuel level. A drastic change is when the fuel level changes faster than the accuracy of the sensor in a span of ten minutes.

### parameters

name	description	type
type	<code>fuel_level_leap</code> for this rule type.	enum
primary_text	Text of rule notification on drastically fuel level increase. It is for <code>fueling</code> event type.	string
secondary_text	Text of rule notification on drastically fuel level decrease. It is for <code>drain</code> event type.	string

### extended parameters

name	description	type
sensor_id	ID of tracked sensor. Should be a fuel level sensor. Only specified if <code>tracker_params</code> is not specified.	int
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title		boolean

name	description	type
	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	
tracker_params	An optional object. Specifies a list of sensors to be tracked in the rule, including for different trackers.	JSON object

#### tracker\_params

name	description	type
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int
sensor_id	ID of tracked sensor. Should be a fuel level sensor.	int

```
{
  "tracker_params": [
    {
      "tracker_id": 10038820,
      "sensor_id": 279421
    },
    {
      "tracker_id": 10038821,
      "sensor_id": 279422
    }
  ]
}
```

## Harsh driving

A rule that triggers on harsh driving. Based on the configs on the device side.

#### parameters

name	description	type
type	<code>harsh_driving</code> for this rule type.	enum

name	description	type
primary_text	Text of rule notification when device detects harsh driving. It is for <code>harsh_driving</code> event type.	string

#### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Advanced driver assistance systems (ADAS)

A rule that triggers on warnings from driver-assistance systems (ADAS). Detected by camera and based on the configs on the device side.

#### parameters

name	description	type
type	<code>driver_assistance</code> for this rule type.	enum
primary_text	Text of rule notification when device detects some of chosen ADAS events.	string

## extended parameters

name	description	type
lane_departure_enabled	If <code>true</code> lane departure tracked. It is for <code>lane_departure</code> event type.	boolean
forward_collision_enabled	If <code>true</code> forward collision tracked. It is for <code>forward_collision_warning</code> event type.	boolean
headway_warning_enabled	If <code>true</code> headway warning tracked. It is for <code>headway_warning</code> event type.	boolean
peds_in_danger_zone_enabled	If <code>true</code> peds in danger zone tracked. It is for <code>peds_in_danger_zone</code> event type.	boolean
peds_collision_warning_enabled	If <code>true</code> peds collision warning works. It is for <code>peds_collision_warning</code> event type.	boolean
traffic_sign_recognition_enabled	If <code>true</code> traffic sign recognition works. It is for <code>tsr_warning</code> event type.	boolean
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Auto geofencing (unauthorized movement detected by location change)

A rule that triggers on auto geofencing. When a car's ignition is off, and it outs the automatically created radius around it.

### parameters

name	description	type
type	<code>auto_geofence</code> for this rule type.	enum
primary_text	Text of rule notification when device outs automatically created geofence around it. It is for <code>auto_geofence_out</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Autocontrol related rules

Autocontrol related tracked rules like alarm, battery, doors and others. Based on the configs on the device side.

## parameters

name	description	type
type	<code>autocontrol</code> for this rule type.	enum
primary_text	Text of rule notification when device determines one of chosen autocontrol related rules.	string

## extended parameters

name	description	type
alarmcontrol	Activation of car alarms. It is for <code>alarmcontrol</code> event type. Described below.	JSON object
battery_off	Disabling of external power supply. It is for <code>battery_off</code> event type. Described below.	JSON object
door_alarm	Opening doors/trunk. It is for <code>door_alarm</code> event type. Described below.	JSON object
hood_alarm	Opening hood. It is for <code>hood_alarm</code> event type. Described below.	JSON object
ignition	Ignition. It is for <code>ignition</code> event type. Described below.	JSON object
parking	Unauthorized movement. It is for <code>parking</code> event type. Described below.	JSON object
gsm_damp	GSM-signal dumping (low signal level). It is for <code>gsm_damp</code> event type. Described below.	JSON object
security_control	Switching ON/OFF security mode. It is for <code>security_control</code> event type. Described below.	JSON object
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule		boolean

name	description	type
	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

### Map of sub-rules settings

```
{
  "alarmcontrol": {
    "enabled": true,
    "sms": false,
    "call": false,
    "email": true,
    "push": true,
    "always_notify": false
  },
  "battery_off": {
    "enabled": true,
    "sms": true,
    "call": false,
    "email": true,
    "push": true
  },
  "door_alarm": {
    "enabled": true,
    "sms": false,
    "call": false,
    "email": true,
    "push": true
  },
  "hood_alarm": {
    "enabled": true,
    "sms": false,
    "call": false,
    "email": true,
    "push": true
  },
  "ignition": {
    "enabled": true,
    "sms": false,
    "call": false,
    "email": true,
    "push": true
  }
}
```

```

    },
    "parking": {
      "enabled": true,
      "sms": false,
      "call": false,
      "email": true,
      "push": true
    },
    "gsm_damp": {
      "enabled": true,
      "sms": false,
      "call": false,
      "email": true,
      "push": true
    },
    "security_control": {
      "enabled": true,
      "sms": false,
      "call": false,
      "email": true,
      "push": true
    }
  }
}

```

## Car crash

A rule that triggers when device's sensors detect car crash. Based on the configs on the device side.

### parameters

name	description	type
type	<code>crash_alarm</code> for this rule type.	enum
primary_text	Text of rule notification when device determines crash by its accelerometer. It is for <code>crash_alarm</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean



name	description	type
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Cruise control switched ON/OFF

A rule that triggers when a device provides cruise control switching event. Based on the configs on the device side.

### parameters

name	description	type
type	<code>cruise_control</code> for this rule type.	enum
primary_text	Text of rule notification when cruise control switch on. It is for <code>cruise_control_on</code> event type.	string
secondary_text	Text of rule notification when cruise control switch off. It is for <code>cruise_control_off</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title		boolean

name	description	type
	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	

## Distance between objects

A rule that triggers a change in distance between objects. The distance is measured by the last valid GPS coordinates between chosen objects.

### parameters

name	description	type
type	<code>distance_control</code> for this rule type.	enum
primary_text	Text of rule notification when distance is breached. It is for <code>distance_breached</code> event type.	string
secondary_text	Text of rule notification when distance is restored. It is for <code>distance_restored</code> event type.	string

### extended parameters

name	description	type
observed_trackers	List of observed tracker IDs.	int array
control_type	Type of distance control. One of <code>["moving_away", "approaching"]</code> .	enum
control_distance_meters	Distance for control in meters.	int
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean

name	description	type
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Driver absence

A rule that triggers when driver leaves or enters cabin. Detected by camera and based on the configs on the device side.

### parameters

name	description	type
type	<code>driver_enter_absence</code> for this rule type.	enum
primary_text	Text of rule notification when driver leaves a cabin. It is for <code>driver_absence</code> event type.	string
secondary_text	Text of rule notification when driver enters a cabin. It is for <code>driver_enter</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title		boolean

name	description	type
	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	

## Driver change

A rule that triggers on driver change automatically by the key or manually in widget with driver from a [drivers list](#).

### parameters

name	description	type
type	<code>driver_change</code> for this rule type.	<a href="#">enum</a>
primary_text	Text of rule notification when a new driver assigned to a device. It is for <code>driver_changed</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Driver distraction

A rule that triggers when driver distracts from the road. Detected by camera and based on the configs on the device side.

### parameters

name	description	type
type	<code>driver_distraction</code> for this rule type.	enum
primary_text	Text of rule notification when driver distraction detected. It is for <code>driver_distraction_started</code> event type.	string
secondary_text	Text of rule notification when driver distraction ends. It is for <code>driver_distraction_finished</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Fall detection

A rule that triggers when g-sensor or accelerometer detects falling.

### parameters

name	description	type
type	<code>g_sensor</code> for this rule type.	enum
primary_text	Text of rule notification when g-sensor detects falling. It is for <code>g_sensor</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Fatigue driving

A rule that triggers on fatigue driving. Detected by camera and based on the configs on the device side.

### parameters

name	description	type
type	<code>fatigue_driving</code> for this rule type.	enum
primary_text	Text of rule notification when fatigue driving is detected. It is for <code>fatigue_driving</code> event type.	string

name	description	type
secondary_text	Text of rule notification when fatigue driving ends. It is for <code>fatigue_driving_finished</code> event type.	string

#### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Identification via RFID/iButton/Camera

A rule that triggers on a driver identification with help of RFID, iButton or Camera. Based on the configs on the device side.

#### parameters

name	description	type
type	<code>driver_identification</code> for this rule type.	enum
primary_text	Text of rule notification when the driver tag has been identified. It is for <code>driver_identified</code> event type.	string
secondary_text	Text of rule notification when the driver tag was not identified. It is for <code>driver_not_identified</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## No movement

A rule that triggers when the device does not detect motion for longer than the time set in its settings. Based on the configs on the device side.

### parameters

name	description	type
type	<code>no_movement</code> for this rule type.	enum
primary_text	Text of rule notification when a device does not detect motion for longer than the time set in its settings. It is for <code>no_movement</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean



name	description	type
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Pressing SOS button

A rule that triggers on SOS button pressing. Based on the configs on the device side.

### parameters

name	description	type
type	<code>sos</code> for this rule type.	enum
primary_text	Text of rule notification when SOS button pressed. It is for <code>sos</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Social distancing monitoring

A rule that triggers on social distancing violation. Similar to distance between objects but related based on the configs on the device side.

### parameters

name	description	type
type	<code>proximity_violation</code> for this rule type.	enum
primary_text	Text of rule notification when safety distance breached. It is for <code>proximity_violation_start</code> event type.	string
secondary_text	Text of rule notification when safety distance restored. It is for <code>proximity_violation_end</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Unauthorized movement (determined by accelerometer)

A rule that triggers on unauthorized movement detected by accelerometer when ignition is off. Based on the configs on the device side.

### parameters

name	description	type
type	<code>parking</code> for this rule type.	enum
primary_text	Text of rule notification when movement detected by device's accelerometer. It is for <code>parking</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Backup battery low

A rule that triggers on backup battery low. Based on the configs on the device side.

### parameters

name	description	type
type	<code>backup_battery_low</code> for this rule type.	enum
primary_text	Text of rule notification when backup battery charge is low. It is for <code>backup_battery_low</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Bracelet sensor

A rule that triggers on bracelet sensor opening/closing. Based on the configs on the device side.

### parameters

name	description	type
type	<code>bracelet</code> for this rule type.	enum
primary_text	Text of rule notification when bracelet opened. It is for <code>bracelet_open</code> event type.	string
secondary_text	Text of rule notification when bracelet closed. It is for <code>bracelet_close</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean

name	description	type
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Call button pressed

A rule that triggers on call button pressing. Based on the configs on the device side.

### parameters

name	description	type
type	<code>call_button_pressed</code> for this rule type.	enum
primary_text	Text of rule notification when call button pressed. It is for <code>call_button_pressed</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title		boolean

name	description	type
	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	

## Car alarm triggered

A rule that triggers on car alarm. Based on the configs on the device side.

### parameters

name	description	type
type	<code>alarmcontrol</code> for this rule type.	enum
primary_text	Text of rule notification when car alarm triggers. It is for <code>alarmcontrol</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Case intrusion

A rule that triggers on case intrusion. Based on the configs on the device side.

## parameters

name	description	type
type	<code>case_intrusion</code> for this rule type.	enum
primary_text	Text of rule notification when device determines case intrusion. It is for <code>case_opened</code> event type.	string

## extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Check engine (MIL)

A rule that triggers on check engine (MIL) events. Based on the configs on the device side.

## parameters

name	description	type
type	<code>check_engine_light</code> for this rule type.	enum
primary_text	Text of rule notification when check engine (MIL) detected by a device. It is for <code>check_engine_light</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Connection/disconnection to the OBDII port

A rule that triggers on connection/disconnection to the OBD2 port. Based on the configs on the device side.

### parameters

name	description	type
type	<code>obd_plug_unplug</code> for this rule type.	enum
primary_text	Text of rule notification when device connected to OBDII port. It is for <code>obd_plug_in</code> event type.	string
secondary_text	Text of rule notification when device disconnected from OBDII port. It is for <code>obd_unplug</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean



name	description	type
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Door opening in alarm mode

A rule that triggers on door opening in alarm mode. Based on the configs on the device side.

### parameters

name	description	type
type	<code>door_alarm</code> for this rule type.	enum
primary_text	Text of rule notification when door opens in alarm mode. It is for <code>door_alarm</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title		boolean

name	description	type
	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	

## External device connection

A rule that triggers on connection/disconnection of an external device. Based on the configs on the device side.

### parameters

name	description	type
type	<code>external_device_connection</code> for this rule type.	enum
primary_text	Text of rule notification when external device connected to tracker. It is for <code>external_device_connected</code> event type.	string
secondary_text	Text of rule notification when external device disconnected from tracker. It is for <code>external_device_disconnected</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## External power cut

A rule that triggers when device disconnects from car's battery. Based on the configs on the device side.

### parameters

name	description	type
type	<code>battery_off</code> for this rule type.	enum
primary_text	Text of rule notification when external power disconnects. It is for <code>battery_off</code> event type.	string
secondary_text	Text of rule notification when external power connects. It is for <code>battery_on</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## GPS antenna disconnected

A rule that triggers on GPS antenna disconnect. Based on the configs on the device side.

### parameters

name	description	type
type	<code>antenna_disconnect</code> for this rule type.	enum
primary_text	Text of rule notification when device determines GPS antenna disconnection. It is for <code>antenna_disconnect</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## GPS jamming (signal dump)

A rule that triggers when device determines GPS jamming. Based on the configs on the device side.

### parameters

name	description	type
type	<code>gps_damp</code> for this rule type.	enum
primary_text	Text of rule notification when device determines GPS jamming. It is for <code>gps_damp</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## GPS signal lost/recover

A rule that triggers on GPS signal lost/recover. Based on the configs on the device side.

### parameters

name	description	type
type	<code>gps_lost_recover</code> for this rule type.	enum
primary_text	Text of rule notification when GPS signal lost. It is for <code>gps_lost</code> event type.	string
secondary_text	Text of rule notification when GPS signal recovers. It is for <code>gps_recover</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule		boolean

name	description	type
	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## GSM jamming (signal dump)

A rule that triggers on GSM jamming. Based on the configs on the device side.

### parameters

name	description	type
type	<code>gsm_damp</code> for this rule type.	enum
primary_text	Text of rule notification when device determines GSM jamming. It is for <code>gsm_damp</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Hood opening in alarm mode

A rule that triggers on hood opening in alarm mode. Based on the configs on the device side.

### parameters

name	description	type
type	<code>hood_alarm</code> for this rule type.	enum
primary_text	Text of rule notification when hood opens in alarm mode. It is for <code>hood_alarm</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Ignition start in alarm mode

A rule that triggers on ignition start in alarm mode. This rule is not related to usual ignition status change. Based on the configs on the device side.

## parameters

name	description	type
type	<code>ignition</code> for this rule type.	enum
primary_text	Text of rule notification when ignition starts in alarm mode. It is for <code>ignition</code> event type.	string

## extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Light sensor

A rule that triggers on when light sensor detects bright/dark environment. Based on the configs on the device side.

## parameters

name	description	type
type	<code>light_sensor</code> for this rule type.	enum
primary_text	Text of rule notification when environment bright. It is for <code>light_sensor_bright</code> event type.	string



name	description	type
secondary_text	Text of rule notification when environment dark. It is for <code>light_sensor_dark</code> event type.	string

#### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Location report on demand

A rule that triggers on location requests.

#### parameters

name	description	type
type	<code>location_response</code> for this rule type.	enum
primary_text	Text of rule notification when location is requested manually from device. It is for <code>location_response</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Locking/unlocking (padlock)

A rule that triggers on locking/unlocking(padlock). Based on the configs on the device side.

### parameters

name	description	type
type	<code>locking_unlocking</code> for this rule type.	enum
primary_text	Text of rule notification when lock opens. It is for <code>lock_opened</code> event type.	string
secondary_text	Text of rule notification when lock closes. It is for <code>lock_closed</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean

name	description	type
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Low battery

A rule that triggers on low internal battery. Based on the device's battery voltage and value specified for the model on the platform.

### parameters

name	description	type
type	<code>lowpower</code> for this rule type.	enum
primary_text	Text of rule notification when device's battery charge is low. It is for <code>lowpower</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title		boolean

name	description	type
	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	

## Padlock tampering

A rule that triggers on padlock tampering. Based on the configs on the device side.

### parameters

name	description	type
type	<code>strap_bolt</code> for this rule type.	enum
primary_text	Text of rule notification when padlock has been forced. It is for <code>strap_bolt_cut</code> event type.	string
secondary_text	Text of rule notification when padlock has been installed. It is for <code>strap_bolt_ins</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Tracker detach from the objects

A rule that triggers when someone detach tracker from the object. Based on the configs on the device side.

### parameters

name	description	type
type	<code>detach</code> for this rule type.	enum
primary_text	Text of rule notification when device determines detach from the object. It is for <code>detach</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Tracker switch ON/OFF

A rule that triggers on tracker switch ON/OFF. Based on the configs on the device side.

### parameters

name	description	type
type	<code>on_off</code> for this rule type.	enum

name	description	type
primary_text	Text of rule notification when tracker switched off. It is for <code>poweroff</code> event type.	string
secondary_text	Text of rule notification when tracker switched on. It is for <code>poweron</code> event type.	string

#### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Tracker switched OFF or lost connection

A rule that triggers when tracker loses connection with the server - gets red offline status and keeps it for X minutes.

#### parameters

name	description	type
type	<code>offline</code> for this rule type.	enum
primary_text	Text of rule notification when tracker switched off or lost connection. It is for <code>gps_lost</code> event type.	string
secondary_text		string

name	description	type
	Text of rule notification when tracker switched on or connection restored. It is for <code>gps_recover</code> event type.	
param	Offline time to notification in minutes.	int

#### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean

## Tracker switched ON

A rule that triggers on tracker switch ON. Based on the configs on the device side.

#### parameters

name	description	type
type	<code>poweron</code> for this rule type.	enum
primary_text	Text of rule notification when tracker switches on. It is for <code>poweroff</code> event type.	string

#### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted		boolean

name	description	type
	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Vibration sensor

A rule that triggers when vibration sensor determines vibration. Based on the configs on the device side.

### parameters

name	description	type
type	<code>vibration</code> for this rule type.	enum
primary_text	Text of rule notification when vibration starts. It is for <code>vibration_start</code> event type.	string
secondary_text	Text of rule notification when vibration ends. It is for <code>vibration_end</code> event type.	string

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title		boolean



name	description	type
	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	

## Inputs triggering.

A rule that triggers when the input state changes.

### parameters

name	description	type
type	<code>input_change</code> for this rule type. Both events for switch on/off will have <code>input_change</code> event type.	enum
primary_text	Text of rule notification when input switches on.	string
secondary_text	Text of rule notification when input switches off.	string
param	Discrete input number.	int

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Outputs triggering

A rule that triggers when the output state changes.

### parameters

name	description	type
type	<code>output_change</code> for this rule type. Both events for switch on/off will have <code>output_change</code> event type.	enum
primary_text	Text of rule notification when output switches on.	string
secondary_text	Text of rule notification when output switches off.	string
param	Output number.	int

### extended parameters

name	description	type
emergency	If <code>true</code> enables emergency notification.	boolean
private_rule	Affects only sub users. If <code>true</code> then the rule and notifications are visible only to sub user.	boolean
zone_limit_inverted	The rule tracked inside of zones if <code>false</code> or outside if <code>true</code> . Default is: <code>false</code> .	boolean
append_zone_title	Show or not the zone labels in a notification text. Must be <code>null</code> or <code>false</code> if the <code>zone_limit_inverted</code> param set to <code>true</code> .	boolean

## Parameter in range

A rule that triggers when value of a chosen measurement sensor gets into or out of specified range. One rule per one sensor and device.

## parameters

name	description	type
type	<code>sensor_range</code> for this rule type.	enum
primary_text	Text of rule notification when sensor value goes out range. It is for <code>sensor_outrange</code> event type.	string
secondary_text	Text of rule notification when sensor value goes into range. It is for <code>sensor_inrange</code> event type.	string

## extended parameters

name	description	type
sensor_id	ID of a tracked sensor. Only specified if <code>tracker_params</code> is not specified.	int
threshold	A threshold for a sensor. If the parameter is omitted or null, the default value 0.03 is used. Ignored if <code>tracker_params</code> is specified.	double
min	A minimum range value. Ignored if <code>tracker_params</code> is specified.	double
max	A maximum range value. Ignored if <code>tracker_params</code> is specified.	double
tracker_params	An optional object. Specifies a list of parameters to be tracked in the rule, including for different trackers.	JSON object

## tracker\_params

name	description	type
sensor_id	ID of a tracked sensor.	int
threshold		double

name	description	type
	A threshold for a sensor. If the parameter is omitted or null, the default value 0.03 is used.	
min	A minimum range value.	double
max	A maximum range value.	double

Example:

```
{
  "tracker_params": [{
    "tracker_id": 10181445,
    "trigger_value": "1",
    "state_field": "ble_magnet_sensor_3"
  }, {
    "tracker_id": 10181446,
    "trigger_value": "1",
    "virtual_sensor_id": 21212
  }
]
```

## State field value

A rule that triggers when specified value of a chosen state field sensor detected.

### parameters

name	description	type
type	<code>state_field_control</code> for this rule type.	enum
primary_text	Text of rule notification when state field determines chosen value. It is for <code>state_field_control</code> event type.	string

### extended parameters

name	description	type
allow_repeat		bool

name	description	type
	Allows notification repeating even if state field value doesn't change.	
repeat_delay_seconds	How many seconds must pass with the same value before notification will be generated again.	int
trigger_value	Expected value to trigger the rule. Only specified if <code>tracker_params</code> is not specified.	string
state_field	State field code. Only specified if <code>virtual_sensor_id</code> and <code>tracker_params</code> are not specified.	enum
virtual_sensor_id	ID of virtual sensor. Only specified if <code>state_field</code> and <code>tracker_params</code> are not specified.	int
tracker_params	An optional object. Specifies a list of parameters to be tracked in the rule, including for different trackers.	JSON object

### tracker\_params

name	description	type
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int
trigger_value	Expected value to trigger the rule.	string
state_field	State field code. Only specified if <code>virtual_sensor_id</code> is not specified.	enum
virtual_sensor_id	ID of virtual sensor. Only specified if <code>state_field</code> is not specified.	int

```
{
  "tracker_params": [{
    "tracker_id": 10181445,
```

```
    "trigger_value": "1",  
    "state_field": "ble_magnet_sensor_3"  
  }, {  
    "tracker_id": 10181446,  
    "trigger_value": "1",  
    "virtual_sensor_id": 21212  
  }  
]  
}
```

Last update: October 4, 2023







# Sensor actions

Contains API calls to interact with sensors.

## Sensor sub-types

### Metering sensor

```
{
  "type": "metering",
  "id": 860250,
  "sensor_type": "temperature",
  "name": "OBD Coolant temperature",
  "input_name": "obd_coolant_t",
  "divider": 1.0,
  "accuracy": 0.0,
  "units": "",
  "units_type": "celsius",
  "parameters": {
    "parent_ids": [123042, 123566],
    "volume": 0.7,
    "min": 0.0,
    "max": 12.0,
    "max_lowering_by_time": 120.0,
    "max_lowering_by_mileage": 120.0,
    "ignore_drains_in_move": true,
    "ignore_refuels_in_move": false,
    "refuel_gap_minutes": 11
  }
}
```

- `type` - string. Always "metering".
- `id` - int. Sensor's id.
- `sensor_type` - [metering sensor type](#). Type of the sensor.
- `name` - string, max size 100. A name of sensor.
- `input_name` - string, max size 64.
- `divider` - double.
- `accuracy` - double. The minimum= 0.0 , maximum= 100.0 with step 0.25 .
- `units` - string.
- `units_type` - [enum](#). Units type for a sensor.

- `parameters` - optional object with additional parameters.
  - `parent_ids` - optional array of `parent_ids` for composite sensor.
  - `volume` - double. Optional. Volume for composite sensor.
  - `parent_ids` - optional. int array. Array of `parent_ids` for composite sensor.
  - `volume` - optional. Double. Volume for composite sensor.
  - `min` - optional. Double. Min acceptable raw value for a sensor.
  - `max` - optional. Double. Max acceptable raw value for a sensor.
  - `max_lowering_by_time` - optional. Double. Max legal value lowering per hour.
  - `max_lowering_by_mileage` - optional. Double. Max legal value lowering per 100 km.
  - `ignore_drains_in_move` - optional. Boolean. Default is false. If true, the fuel drains will not be detected during movement.
  - `ignore_refuels_in_move` - optional. Boolean. Default is false. If true, the refuels will not be detected during movement.
  - `refuel_gap_minutes` - optional. Integer. Default is 5. Time in minutes after the start of the movement, refuels will be detected during movement.

### Metering sensor type values

- `fuel`
- `temperature`
- `rpm`
- `custom`
- `fuel_consumption`
- `instant_consumption`
- `power`
- `speed`
- `flow_meter`
- `acceleration`

### Discrete input

```
{
  "type": "discrete",
  "id": 888951,
  "sensor_type": "ignition",
```

```

    "name": "Ignition",
    "input_number": 4
  }

```

- `type` - string. Always "discrete".
- `id` - int. An ID of a sensor.
- `sensor_type` - [discrete sensor type](#). Type of the sensor.
- `name` - string, max size 100.
- `input_number` - int, [1..8]. Assigned input number.

### Discrete sensor type values

- `ignition`
- `sos_button`
- `power`
- `engine`
- `car_alarm`
- `door`
- `charge`
- `detach`
- `custom`

### Virtual sensor

```

{
  "type": "virtual",
  "id": 1700049,
  "sensor_type": "virtual_ignition",
  "name": "Virtual Ignition",
  "input_name": "board_voltage",
  "parameters": {
    "calc_method": "in_range",
    "range_from": 13.4,
    "value_titles": [{
      "value": "0",
      "title": "Off"
    }, {
      "value": "1",
      "title": "On"
    }]
  }
}

```

```
}  
}
```

- `type` - string. Always "virtual".
- `id` - int. Sensor's id.
- `sensor_type` - [virtual sensor type](#). Type of the sensor. "virtual\_ignition" for virtual ignition or "state" for others.
- `name` - string, max size 100. A name of sensor.
- `input_name` - string, max size 64. A source input field name (identifier).
- `parameters` - optional object with additional parameters.
- `calc_method` - [enum](#). A method of sensor value calculation. One of this: "in\_range", "identity", "bit\_index".
- `range_from` - double. Low bound of range. It is used only with "in\_range" calc method.
- `range_to` - double. High bound of range. It is used only with "in\_range" calc method.
- `bit_index` - int, [1..N]. A bit index in input field source value. It is used only with "bit\_index" calc method.
- `value_titles` - mapping for bind special titles for sensor values, if it is necessary.
  - `value` - string, max size 64. Sensor value.
  - `title` - string, max size 64. Title for the sensor value.

### Virtual sensor type values

- `state`
- `virtual_ignition`

Some requirements:

- There can be only one virtual sensor with type `virtual_ignition` for tracker.
- One or both field `range_from` and `range_to` must be present for the calc method "in\_range".
- Field `bit_index` must be present for the calc method "bit\_index".
- There can be no more than 100 value titles.
- All values must be unique within `value_titles`.

Described work with virtual sensors in our [instructions](#).

## API actions

API base path: `/tracker/sensor`.

### batch\_list

List tracker sensors bound to trackers with specified identifiers (parameter `trackers`).

There exist a similar method for working with a single tracker - [list](#).

#### parameters

Name	Description	Type
trackers	Set of tracker identifiers. Each of the relevant trackers must be accessible to the authorized user and not be blocked. Number of trackers (length of array) is limited to a maximum of 500 (this number may be changed in future).	int array

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/sensor/batch_list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "trackers": [204104, 181451]}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/sensor/batch_list?
hash=a6aa75587e5c59c32d347da438505fc3&trackers=[204104, 181451]
```

#### response

Contains a map, where keys are IDs from **trackers** parameter and values are lists of [sensor](#) objects.

```
{
  "success": true,
  "result": {
    "11": [
      {
        "id": 1,
        "type": "discrete",
        "sensor_type": "fuel",
        "name": "Main tank",
        "input_name": "fuel_level",
        "group_type": null,

```

```

        "divider": 1,
        "accuracy": 0.0,
        "units": null,
        "units_type": "litre"
    }
]
}
}

```

## errors

- 217 - List contains nonexistent entities - if one of `trackers` either does not exist or is blocked.
- 221 - Device limit exceeded - if too many IDs were passed in `trackers` parameter.

## create

Creates a sensor.

**required sub-user rights:** `tracker_update`.

## parameters

name	description	type
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int
sensor	<a href="#">Sensor object</a> .	JSON object

## examples

### cURL

```

curl -X POST 'https://api.navixy.com/v2/tracker/sensor/create' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "sensor": {"type": "metering", "id": 860250, "sensor_type": "temperature", "name": "OBD Coolant temperature", "input_name": "obd_coolant_t", "divider": 1.0, "accuracy": 0.0, "units": "", "units_type": "celsius"}}'

```

## response

```

{
    "success": true,

```

```
"id": 937
}
```

- `id` - int. An ID of created sensor.

## errors

- 232 - Input already in use – if given input number (for discrete input) or input name (for metering sensor) already in use.
- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions, or some other reason.
- 219 - Not allowed for clones of the device – if tracker is clone.
- 270 - Too many sensors of same type - the number of tracker's sensors, having same `sensor_type` is limited.

## delete

Deletes a sensor with `sensor_id` from the database.

**required sub-user rights:** `tracker_update`.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
sensor_id	Sensor ID.	int	234567

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/sensor/delete' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "sensor_id": 23456}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/sensor/delete?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456&sensor_id=2345
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database - if sensor with a sensor\_id is not exists or owned by other user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 219 – Not allowed for clones of the device - if tracker is clone.

## list

List tracker sensors bound to tracker with specified ID ( `tracker_id` parameter).

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/sensor/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":  
123456}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/sensor/list?  
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456
```

## response

```
{  
  "success": true,  
  "list": [{  
    "type": "metering",  
    "id": 860250,  
    "sensor_type": "temperature",  
    "name": "OBD Coolant temperature",  
    "input_name": "obd_coolant_t",  
    "divider": 1.0,  
    "accuracy": 0.0,
```



```
    "units": "",
    "units_type": "celsius"
  }
}
```

- `list` - list of sensor objects. See [sensor](#) object description.

## errors

- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions, or some other reason.

## update

Updates sensor.

**required sub-user rights:** `tracker_update`.

## parameters

name	description	type
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int
sensor	<a href="#">Sensor object</a> .	JSON object

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/sensor/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "sensor": {"type": "metering", "id": 860250, "sensor_type": "temperature", "name": "OBD Coolant temperature", "input_name": "obd_coolant_t", "divider": 1.0, "accuracy": 0.0, "units": "", "units_type": "celsius"}}'
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if sensor not exists or owned by other user.
- 232 - Input already in use – if given input number (for discrete input) or input name (for metering sensor) already in use.
- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions, or some other reason.
- 219 - Not allowed for clones of the device – if tracker is clone.

## batch\_copy

Copies sensors from one tracker to another.

### Important

This operation will delete sensors of target trackers, and some sensor data could be lost!

**required sub-user rights:** `tracker_update`.

## parameters

name	description	type	format
base_tracker_id	ID of the base tracker (aka "object_id") from which you want to copy sensors. Tracker must belong to authorized user and not be blocked.	int	123456
trackers	ID of trackers. Target trackers for copying sensors.	[int]	[ 12345, 54321 ]

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/sensor/batch_copy' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
    "base_tracker_id": 123456, "trackers": [56789, 54321]}'
```

## response

```
{
  "success": true
}
```

## errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 272 – Trackers must have same models - if base tracker and one of target trackers has a different model.

## data/read

Gets all `metering` or `virtual` sensor readings with values and time per requested period. It can't be used with discrete sensor.

## parameters

name	description	type	format
tracker_id	ID of the base tracker (aka "object_id") from which you want to read sensor's data. Tracker must belong to authorized user and not be blocked.	int	123456
sensor_id	Sensor ID.	int	234567
from	Start date and time for searching.	date/ time	"2022-02-28 00:00:00"
to	End date and time for searching. Must be after <code>from</code> date. Maximum period is <code>maxReportTimeSpan</code> , default 30 days.	date/ time	"2022-03-28 23:59:00"
raw_data	If <code>true</code> then the response will contain raw data without any calibration and multiplication. Affects only <code>metering</code> sensors. Default value is false for backward compatibility.	boolean	false

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/sensor/data/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "sensor_id": 1456789, "from": "2022-02-28 00:00:00", "to": "2022-03-28 23:59:00"}'
```

## response

```
{
  "success": true,
  "list": [
    {
      "value": 100500,
      "get_time": "2022-02-28 00:00:00"
    },
    {
      "value": 100501,
      "get_time": "2022-02-28 00:00:30"
    }
  ]
}
```

- `value` - a value of sensor data. It can be double, int or string depending on the sensor type.
- `get_time` - time when value was received.

## errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 211 – Requested time span is too big - if interval between "from" and "to" is too big. Maximum period is `maxReportTimeSpan`.
- 228 – Not supported by the sensor - if sensor is not a metering or virtual sensor.

Last update: April 1, 2024





# Sensor calibration data

Contains API calls to read and set sensor calibration data which is used for calibration received data from sensors to the convenient format. For example, analog fuel sensor provides Volts that should be calibrated to Liters.

## API actions

API path: `/tracker/sensor/calibration_data`.

### read

Gets calibration data for sensor.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
sensor_id	ID of the sensor.	int	12345

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/sensor/calibration_data/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "sensor_id": 12345}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/sensor/calibration_data/read?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456&sensor_id=1234
```

#### response

```
{
  "success": true,
```

```
"value": [{ "in":0.0, "out":0.0}, {"in":0.7, "out":60.0}]
}
```

- `value` - list of objects containing calibration data.

## errors

- 201 – Not found in the database (if there is no tracker with such ID belonging to authorized user).
- 228 – Not supported by the sensor (if sensor doesn't support calibration).

## update

Replaces the calibration data for a sensor.

**required sub-user rights:** `tracker_update`.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
sensor_id	ID of the sensor.	int	12345
data	Array of calibration data objects.	array of JSON object	[{"in":0.0, "out":0.0}, {"in":0.7, "out":60.0}]

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/sensor/calibration_data/update' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "sensor_id": 12345, "data": [{"in":0.0, "out":0.0}, {"in":0.7, "out":60.0}]}'
```

## response



```
{ "success": true }
```

#### errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 228 – Not supported by the sensor - if sensor doesn't support calibration.
- 228 – Not supported by the sensor - if sensor doesn't support calibration.
- 219 – Not allowed for clones of the device - if tracker is clone.

## upload\_omnicomm

Replaces the calibration data for a sensor from Omnicomm LLS monitor's XML configuration file. If XML file contains information about multiple sensors, user must specify which sensor number to use.

**required sub-user rights:** `tracker_update`.

**MUST** be a POST multipart request (multipart/form-data), with one of the parts being an XML file upload (with the name "file").

#### parameters

name	description	type
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int
sensor_id	ID of the sensor.	int
file	A file upload containing LLS monitor XML file.	file upload

#### response

```
{ "success": true }
```

#### errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.

- 228 – Not supported by the sensor - if sensor doesn't support calibration.
- 219 – Not allowed for clones of the device - if tracker is clone.
- 233 – No data file - if file part is missing.
- 234 – Invalid data format - if supplied file is not a valid LLS monitor XML file.
- 235 – Missing calibration data - if there is no calibration data for the specified sensor number.

Last update: December 26, 2022





# Input name

API base path: `/tracker/sensor/input_name`.

API call to get all sensor inputs and state fields existing in the system and their descriptions.

## API actions

### list

This will provide descriptions of all sensor inputs and state fields present in the system. These descriptions will be given in the language according to the user's locale.

### parameters

Only API key `hash`.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/sensor/input_name/
list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/tracker/sensor/input_name/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

### response

For every input following properties returned: `input_name` and `description`.

`input_name` is an enum member, same as in [sensor object](#).

`description` is made in current user's language (according to [locale settings](#)).

## Response

```
{
  "success": true,
  "list": [
    {
      "input_name": "acceleration",
      "description": "Acceleration"
    },
    {
      "input_name": "analog_1",
      "description": "Analog sensor #1"
    },
    {
      "input_name": "analog_2",
      "description": "Analog sensor #2"
    },
    {
      "input_name": "analog_3",
      "description": "Analog sensor #3"
    },
    {
      "input_name": "analog_4",
      "description": "Analog sensor #4"
    },
    {
      "input_name": "analog_5",
      "description": "Analog sensor #5"
    },
    {
      "input_name": "analog_6",
      "description": "Analog sensor #6"
    },
    {
      "input_name": "analog_7",
      "description": "Analog sensor #7"
    },
    {
      "input_name": "analog_8",
      "description": "Analog sensor #8"
    },
    {
      "input_name": "axis_x",
      "description": "Acceleration by X axis"
    },
    {
      "input_name": "axis_y",
      "description": "Acceleration by Y axis"
    },
    {
      "input_name": "axis_z",
      "description": "Acceleration by Z axis"
    },
    {
      "input_name": "battery_current",
      "description": "Battery current"
    },
    {
      "input_name": "battery_level",
```

```
        "description": "Battery level"
    },
    {
        "input_name": "battery_voltage",
        "description": "Battery voltage"
    },
    {
        "input_name": "ble_battery_level_1",
        "description": "BLE: Battery level #1"
    },
    {
        "input_name": "ble_battery_level_10",
        "description": "BLE: Battery level #10"
    },
    {
        "input_name": "ble_battery_level_11",
        "description": "BLE: Battery level #11"
    },
    {
        "input_name": "ble_battery_level_12",
        "description": "BLE: Battery level #12"
    },
    {
        "input_name": "ble_battery_level_13",
        "description": "BLE: Battery level #13"
    },
    {
        "input_name": "ble_battery_level_14",
        "description": "BLE: Battery level #14"
    },
    {
        "input_name": "ble_battery_level_15",
        "description": "BLE: Battery level #15"
    },
    {
        "input_name": "ble_battery_level_16",
        "description": "BLE: Battery level #16"
    },
    {
        "input_name": "ble_battery_level_2",
        "description": "BLE: Battery level #2"
    },
    {
        "input_name": "ble_battery_level_3",
        "description": "BLE: Battery level #3"
    },
    {
        "input_name": "ble_battery_level_4",
        "description": "BLE: Battery level #4"
    },
    {
        "input_name": "ble_battery_level_5",
        "description": "BLE: Battery level #5"
    },
    {
        "input_name": "ble_battery_level_6",
        "description": "BLE: Battery level #6"
    },
    {
        "input_name": "ble_battery_level_7",
        "description": "BLE: Battery level #7"
    },
    {
        "input_name": "ble_battery_level_8",
        "description": "BLE: Battery level #8"
    },
    {
        "input_name": "ble_battery_level_9",
        "description": "BLE: Battery level #9"
    }
]
```

```
    },
    {
      "input_name": "ble_battery_level_8",
      "description": "BLE: Battery level #8"
    },
    {
      "input_name": "ble_battery_level_9",
      "description": "BLE: Battery level #9"
    },
    {
      "input_name": "ble_battery_voltage_1",
      "description": "BLE: Battery voltage #1"
    },
    {
      "input_name": "ble_battery_voltage_10",
      "description": "BLE: Battery voltage #10"
    },
    {
      "input_name": "ble_battery_voltage_11",
      "description": "BLE: Battery voltage #11"
    },
    {
      "input_name": "ble_battery_voltage_12",
      "description": "BLE: Battery voltage #12"
    },
    {
      "input_name": "ble_battery_voltage_13",
      "description": "BLE: Battery voltage #13"
    },
    {
      "input_name": "ble_battery_voltage_14",
      "description": "BLE: Battery voltage #14"
    },
    {
      "input_name": "ble_battery_voltage_15",
      "description": "BLE: Battery voltage #15"
    },
    {
      "input_name": "ble_battery_voltage_16",
      "description": "BLE: Battery voltage #16"
    },
    {
      "input_name": "ble_battery_voltage_2",
      "description": "BLE: Battery voltage #2"
    },
    {
      "input_name": "ble_battery_voltage_3",
      "description": "BLE: Battery voltage #3"
    },
    {
      "input_name": "ble_battery_voltage_4",
      "description": "BLE: Battery voltage #4"
    },
    {
      "input_name": "ble_battery_voltage_5",
      "description": "BLE: Battery voltage #5"
    },
    {
      "input_name": "ble_battery_voltage_6",
      "description": "BLE: Battery voltage #6"
    },
  ],
```



```
{
  "input_name": "ble_battery_voltage_7",
  "description": "BLE: Battery voltage #7"
},
{
  "input_name": "ble_battery_voltage_8",
  "description": "BLE: Battery voltage #8"
},
{
  "input_name": "ble_battery_voltage_9",
  "description": "BLE: Battery voltage #9"
},
{
  "input_name": "ble_humidity_1",
  "description": "BLE: Humidity sensor #1"
},
{
  "input_name": "ble_humidity_10",
  "description": "BLE: Humidity sensor #10"
},
{
  "input_name": "ble_humidity_11",
  "description": "BLE: Humidity sensor #11"
},
{
  "input_name": "ble_humidity_12",
  "description": "BLE: Humidity sensor #12"
},
{
  "input_name": "ble_humidity_13",
  "description": "BLE: Humidity sensor #13"
},
{
  "input_name": "ble_humidity_14",
  "description": "BLE: Humidity sensor #14"
},
{
  "input_name": "ble_humidity_15",
  "description": "BLE: Humidity sensor #15"
},
{
  "input_name": "ble_humidity_16",
  "description": "BLE: Humidity sensor #16"
},
{
  "input_name": "ble_humidity_2",
  "description": "BLE: Humidity sensor #2"
},
{
  "input_name": "ble_humidity_3",
  "description": "BLE: Humidity sensor #3"
},
{
  "input_name": "ble_humidity_4",
  "description": "BLE: Humidity sensor #4"
},
{
  "input_name": "ble_humidity_5",
  "description": "BLE: Humidity sensor #5"
},
{
```

```
    "input_name": "ble_humidity_6",
    "description": "BLE: Humidity sensor #6"
  },
  {
    "input_name": "ble_humidity_7",
    "description": "BLE: Humidity sensor #7"
  },
  {
    "input_name": "ble_humidity_8",
    "description": "BLE: Humidity sensor #8"
  },
  {
    "input_name": "ble_humidity_9",
    "description": "BLE: Humidity sensor #9"
  },
  {
    "input_name": "ble_lls_level_1",
    "description": "BLE: LLS #1 level"
  },
  {
    "input_name": "ble_lls_level_10",
    "description": "BLE: LLS #10 level"
  },
  {
    "input_name": "ble_lls_level_2",
    "description": "BLE: LLS #2 level"
  },
  {
    "input_name": "ble_lls_level_3",
    "description": "BLE: LLS #3 level"
  },
  {
    "input_name": "ble_lls_level_4",
    "description": "BLE: LLS #4 level"
  },
  {
    "input_name": "ble_lls_level_5",
    "description": "BLE: LLS #5 level"
  },
  {
    "input_name": "ble_lls_level_6",
    "description": "BLE: LLS #6 level"
  },
  {
    "input_name": "ble_lls_level_7",
    "description": "BLE: LLS #7 level"
  },
  {
    "input_name": "ble_lls_level_8",
    "description": "BLE: LLS #8 level"
  },
  {
    "input_name": "ble_lls_level_9",
    "description": "BLE: LLS #9 level"
  },
  {
    "input_name": "ble_lls_temperature_1",
    "description": "BLE: LLS #1 temperature"
  },
  {
    "input_name": "ble_lls_temperature_10",
```

```
        "description": "BLE: LLS #10 temperature"
    },
    {
        "input_name": "ble_lls_temperature_2",
        "description": "BLE: LLS #2 temperature"
    },
    {
        "input_name": "ble_lls_temperature_3",
        "description": "BLE: LLS #3 temperature"
    },
    {
        "input_name": "ble_lls_temperature_4",
        "description": "BLE: LLS #4 temperature"
    },
    {
        "input_name": "ble_lls_temperature_5",
        "description": "BLE: LLS #5 temperature"
    },
    {
        "input_name": "ble_lls_temperature_6",
        "description": "BLE: LLS #6 temperature"
    },
    {
        "input_name": "ble_lls_temperature_7",
        "description": "BLE: LLS #7 temperature"
    },
    {
        "input_name": "ble_lls_temperature_8",
        "description": "BLE: LLS #8 temperature"
    },
    {
        "input_name": "ble_lls_temperature_9",
        "description": "BLE: LLS #9 temperature"
    },
    {
        "input_name": "ble_luminosity_1",
        "description": "BLE: Luminosity #1"
    },
    {
        "input_name": "ble_luminosity_10",
        "description": "BLE: Luminosity #10"
    },
    {
        "input_name": "ble_luminosity_11",
        "description": "BLE: Luminosity #11"
    },
    {
        "input_name": "ble_luminosity_12",
        "description": "BLE: Luminosity #12"
    },
    {
        "input_name": "ble_luminosity_13",
        "description": "BLE: Luminosity #13"
    },
    {
        "input_name": "ble_luminosity_14",
        "description": "BLE: Luminosity #14"
    },
    {
        "input_name": "ble_luminosity_15",
        "description": "BLE: Luminosity #15"
    }
```

```
},
{
  "input_name": "ble_luminosity_16",
  "description": "BLE: Luminosity #16"
},
{
  "input_name": "ble_luminosity_2",
  "description": "BLE: Luminosity #2"
},
{
  "input_name": "ble_luminosity_3",
  "description": "BLE: Luminosity #3"
},
{
  "input_name": "ble_luminosity_4",
  "description": "BLE: Luminosity #4"
},
{
  "input_name": "ble_luminosity_5",
  "description": "BLE: Luminosity #5"
},
{
  "input_name": "ble_luminosity_6",
  "description": "BLE: Luminosity #6"
},
{
  "input_name": "ble_luminosity_7",
  "description": "BLE: Luminosity #7"
},
{
  "input_name": "ble_luminosity_8",
  "description": "BLE: Luminosity #8"
},
{
  "input_name": "ble_luminosity_9",
  "description": "BLE: Luminosity #9"
},
{
  "input_name": "ble_signal_strength_1",
  "description": "BLE: Signal strength #1"
},
{
  "input_name": "ble_signal_strength_10",
  "description": "BLE: Signal strength #10"
},
{
  "input_name": "ble_signal_strength_11",
  "description": "BLE: Signal strength #11"
},
{
  "input_name": "ble_signal_strength_12",
  "description": "BLE: Signal strength #12"
},
{
  "input_name": "ble_signal_strength_13",
  "description": "BLE: Signal strength #13"
},
{
  "input_name": "ble_signal_strength_14",
  "description": "BLE: Signal strength #14"
},
},
```

```
{
  "input_name": "ble_signal_strength_15",
  "description": "BLE: Signal strength #15"
},
{
  "input_name": "ble_signal_strength_16",
  "description": "BLE: Signal strength #16"
},
{
  "input_name": "ble_signal_strength_2",
  "description": "BLE: Signal strength #2"
},
{
  "input_name": "ble_signal_strength_3",
  "description": "BLE: Signal strength #3"
},
{
  "input_name": "ble_signal_strength_4",
  "description": "BLE: Signal strength #4"
},
{
  "input_name": "ble_signal_strength_5",
  "description": "BLE: Signal strength #5"
},
{
  "input_name": "ble_signal_strength_6",
  "description": "BLE: Signal strength #6"
},
{
  "input_name": "ble_signal_strength_7",
  "description": "BLE: Signal strength #7"
},
{
  "input_name": "ble_signal_strength_8",
  "description": "BLE: Signal strength #8"
},
{
  "input_name": "ble_signal_strength_9",
  "description": "BLE: Signal strength #9"
},
{
  "input_name": "ble_temp_sensor_1",
  "description": "BLE: temperature sensor #1"
},
{
  "input_name": "ble_temp_sensor_10",
  "description": "BLE: temperature sensor #10"
},
{
  "input_name": "ble_temp_sensor_11",
  "description": "BLE: temperature sensor #11"
},
{
  "input_name": "ble_temp_sensor_12",
  "description": "BLE: temperature sensor #12"
},
{
  "input_name": "ble_temp_sensor_13",
  "description": "BLE: temperature sensor #13"
},
{
  "input_name": "ble_temp_sensor_2",
  "description": "BLE: temperature sensor #2"
},
{
  "input_name": "ble_temp_sensor_3",
  "description": "BLE: temperature sensor #3"
},
{
  "input_name": "ble_temp_sensor_4",
  "description": "BLE: temperature sensor #4"
},
{
  "input_name": "ble_temp_sensor_5",
  "description": "BLE: temperature sensor #5"
},
{
  "input_name": "ble_temp_sensor_6",
  "description": "BLE: temperature sensor #6"
},
{
  "input_name": "ble_temp_sensor_7",
  "description": "BLE: temperature sensor #7"
},
{
  "input_name": "ble_temp_sensor_8",
  "description": "BLE: temperature sensor #8"
},
{
  "input_name": "ble_temp_sensor_9",
  "description": "BLE: temperature sensor #9"
}
}
```

```
    "input_name": "ble_temp_sensor_14",
    "description": "BLE: temperature sensor #14"
  },
  {
    "input_name": "ble_temp_sensor_15",
    "description": "BLE: temperature sensor #15"
  },
  {
    "input_name": "ble_temp_sensor_16",
    "description": "BLE: temperature sensor #16"
  },
  {
    "input_name": "ble_temp_sensor_2",
    "description": "BLE: temperature sensor #2"
  },
  {
    "input_name": "ble_temp_sensor_3",
    "description": "BLE: temperature sensor #3"
  },
  {
    "input_name": "ble_temp_sensor_4",
    "description": "BLE: temperature sensor #4"
  },
  {
    "input_name": "ble_temp_sensor_5",
    "description": "BLE: temperature sensor #5"
  },
  {
    "input_name": "ble_temp_sensor_6",
    "description": "BLE: temperature sensor #6"
  },
  {
    "input_name": "ble_temp_sensor_7",
    "description": "BLE: temperature sensor #7"
  },
  {
    "input_name": "ble_temp_sensor_8",
    "description": "BLE: temperature sensor #8"
  },
  {
    "input_name": "ble_temp_sensor_9",
    "description": "BLE: temperature sensor #9"
  },
  {
    "input_name": "ble_tire_pressure_1",
    "description": "BLE: Tire pressure #1"
  },
  {
    "input_name": "ble_tire_pressure_10",
    "description": "BLE: Tire pressure #10"
  },
  {
    "input_name": "ble_tire_pressure_11",
    "description": "BLE: Tire pressure #11"
  },
  {
    "input_name": "ble_tire_pressure_12",
    "description": "BLE: Tire pressure #12"
  },
  {
    "input_name": "ble_tire_pressure_13",
```

```
    "description": "BLE: Tire pressure #13"
  },
  {
    "input_name": "ble_tire_pressure_14",
    "description": "BLE: Tire pressure #14"
  },
  {
    "input_name": "ble_tire_pressure_15",
    "description": "BLE: Tire pressure #15"
  },
  {
    "input_name": "ble_tire_pressure_16",
    "description": "BLE: Tire pressure #16"
  },
  {
    "input_name": "ble_tire_pressure_17",
    "description": "BLE: Tire pressure #17"
  },
  {
    "input_name": "ble_tire_pressure_18",
    "description": "BLE: Tire pressure #18"
  },
  {
    "input_name": "ble_tire_pressure_19",
    "description": "BLE: Tire pressure #19"
  },
  {
    "input_name": "ble_tire_pressure_2",
    "description": "BLE: Tire pressure #2"
  },
  {
    "input_name": "ble_tire_pressure_20",
    "description": "BLE: Tire pressure #20"
  },
  {
    "input_name": "ble_tire_pressure_21",
    "description": "BLE: Tire pressure #21"
  },
  {
    "input_name": "ble_tire_pressure_22",
    "description": "BLE: Tire pressure #22"
  },
  {
    "input_name": "ble_tire_pressure_23",
    "description": "BLE: Tire pressure #23"
  },
  {
    "input_name": "ble_tire_pressure_24",
    "description": "BLE: Tire pressure #24"
  },
  {
    "input_name": "ble_tire_pressure_25",
    "description": "BLE: Tire pressure #25"
  },
  {
    "input_name": "ble_tire_pressure_26",
    "description": "BLE: Tire pressure #26"
  },
  {
    "input_name": "ble_tire_pressure_27",
    "description": "BLE: Tire pressure #27"
```

```
},
{
  "input_name": "ble_tire_pressure_28",
  "description": "BLE: Tire pressure #28"
},
{
  "input_name": "ble_tire_pressure_29",
  "description": "BLE: Tire pressure #29"
},
{
  "input_name": "ble_tire_pressure_3",
  "description": "BLE: Tire pressure #3"
},
{
  "input_name": "ble_tire_pressure_30",
  "description": "BLE: Tire pressure #30"
},
{
  "input_name": "ble_tire_pressure_4",
  "description": "BLE: Tire pressure #4"
},
{
  "input_name": "ble_tire_pressure_5",
  "description": "BLE: Tire pressure #5"
},
{
  "input_name": "ble_tire_pressure_6",
  "description": "BLE: Tire pressure #6"
},
{
  "input_name": "ble_tire_pressure_7",
  "description": "BLE: Tire pressure #7"
},
{
  "input_name": "ble_tire_pressure_8",
  "description": "BLE: Tire pressure #8"
},
{
  "input_name": "ble_tire_pressure_9",
  "description": "BLE: Tire pressure #9"
},
{
  "input_name": "ble_tire_temperature_1",
  "description": "BLE: Tire air temperature #1"
},
{
  "input_name": "ble_tire_temperature_10",
  "description": "BLE: Tire air temperature #10"
},
{
  "input_name": "ble_tire_temperature_11",
  "description": "BLE: Tire air temperature #11"
},
{
  "input_name": "ble_tire_temperature_12",
  "description": "BLE: Tire air temperature #12"
},
{
  "input_name": "ble_tire_temperature_13",
  "description": "BLE: Tire air temperature #13"
},
},
```



```
{
  "input_name": "ble_tire_temperature_14",
  "description": "BLE: Tire air temperature #14"
},
{
  "input_name": "ble_tire_temperature_15",
  "description": "BLE: Tire air temperature #15"
},
{
  "input_name": "ble_tire_temperature_16",
  "description": "BLE: Tire air temperature #16"
},
{
  "input_name": "ble_tire_temperature_17",
  "description": "BLE: Tire air temperature #17"
},
{
  "input_name": "ble_tire_temperature_18",
  "description": "BLE: Tire air temperature #18"
},
{
  "input_name": "ble_tire_temperature_19",
  "description": "BLE: Tire air temperature #19"
},
{
  "input_name": "ble_tire_temperature_2",
  "description": "BLE: Tire air temperature #2"
},
{
  "input_name": "ble_tire_temperature_20",
  "description": "BLE: Tire air temperature #20"
},
{
  "input_name": "ble_tire_temperature_21",
  "description": "BLE: Tire air temperature #21"
},
{
  "input_name": "ble_tire_temperature_22",
  "description": "BLE: Tire air temperature #22"
},
{
  "input_name": "ble_tire_temperature_23",
  "description": "BLE: Tire air temperature #23"
},
{
  "input_name": "ble_tire_temperature_24",
  "description": "BLE: Tire air temperature #24"
},
{
  "input_name": "ble_tire_temperature_25",
  "description": "BLE: Tire air temperature #25"
},
{
  "input_name": "ble_tire_temperature_26",
  "description": "BLE: Tire air temperature #26"
},
{
  "input_name": "ble_tire_temperature_27",
  "description": "BLE: Tire air temperature #27"
}
```

```
    "input_name": "ble_tire_temperature_28",
    "description": "BLE: Tire air temperature #28"
  },
  {
    "input_name": "ble_tire_temperature_29",
    "description": "BLE: Tire air temperature #29"
  },
  {
    "input_name": "ble_tire_temperature_3",
    "description": "BLE: Tire air temperature #3"
  },
  {
    "input_name": "ble_tire_temperature_30",
    "description": "BLE: Tire air temperature #30"
  },
  {
    "input_name": "ble_tire_temperature_4",
    "description": "BLE: Tire air temperature #4"
  },
  {
    "input_name": "ble_tire_temperature_5",
    "description": "BLE: Tire air temperature #5"
  },
  {
    "input_name": "ble_tire_temperature_6",
    "description": "BLE: Tire air temperature #6"
  },
  {
    "input_name": "ble_tire_temperature_7",
    "description": "BLE: Tire air temperature #7"
  },
  {
    "input_name": "ble_tire_temperature_8",
    "description": "BLE: Tire air temperature #8"
  },
  {
    "input_name": "ble_tire_temperature_9",
    "description": "BLE: Tire air temperature #9"
  },
  {
    "input_name": "ble_user_data_1",
    "description": "BLE custom user data #1"
  },
  {
    "input_name": "ble_user_data_10",
    "description": "BLE custom user data #10"
  },
  {
    "input_name": "ble_user_data_11",
    "description": "BLE custom user data #11"
  },
  {
    "input_name": "ble_user_data_12",
    "description": "BLE custom user data #12"
  },
  {
    "input_name": "ble_user_data_13",
    "description": "BLE custom user data #13"
  },
  {
    "input_name": "ble_user_data_14",
```

```
    "description": "BLE custom user data #14"
  },
  {
    "input_name": "ble_user_data_15",
    "description": "BLE custom user data #15"
  },
  {
    "input_name": "ble_user_data_16",
    "description": "BLE custom user data #16"
  },
  {
    "input_name": "ble_user_data_17",
    "description": "BLE custom user data #17"
  },
  {
    "input_name": "ble_user_data_18",
    "description": "BLE custom user data #18"
  },
  {
    "input_name": "ble_user_data_19",
    "description": "BLE custom user data #19"
  },
  {
    "input_name": "ble_user_data_2",
    "description": "BLE custom user data #2"
  },
  {
    "input_name": "ble_user_data_20",
    "description": "BLE custom user data #20"
  },
  {
    "input_name": "ble_user_data_21",
    "description": "BLE custom user data #21"
  },
  {
    "input_name": "ble_user_data_22",
    "description": "BLE custom user data #22"
  },
  {
    "input_name": "ble_user_data_23",
    "description": "BLE custom user data #23"
  },
  {
    "input_name": "ble_user_data_24",
    "description": "BLE custom user data #24"
  },
  {
    "input_name": "ble_user_data_25",
    "description": "BLE custom user data #25"
  },
  {
    "input_name": "ble_user_data_26",
    "description": "BLE custom user data #26"
  },
  {
    "input_name": "ble_user_data_27",
    "description": "BLE custom user data #27"
  },
  {
    "input_name": "ble_user_data_28",
    "description": "BLE custom user data #28"
```

```
},
{
  "input_name": "ble_user_data_29",
  "description": "BLE custom user data #29"
},
{
  "input_name": "ble_user_data_3",
  "description": "BLE custom user data #3"
},
{
  "input_name": "ble_user_data_30",
  "description": "BLE custom user data #30"
},
{
  "input_name": "ble_user_data_31",
  "description": "BLE custom user data #31"
},
{
  "input_name": "ble_user_data_32",
  "description": "BLE custom user data #32"
},
{
  "input_name": "ble_user_data_33",
  "description": "BLE custom user data #33"
},
{
  "input_name": "ble_user_data_34",
  "description": "BLE custom user data #34"
},
{
  "input_name": "ble_user_data_35",
  "description": "BLE custom user data #35"
},
{
  "input_name": "ble_user_data_36",
  "description": "BLE custom user data #36"
},
{
  "input_name": "ble_user_data_37",
  "description": "BLE custom user data #37"
},
{
  "input_name": "ble_user_data_38",
  "description": "BLE custom user data #38"
},
{
  "input_name": "ble_user_data_39",
  "description": "BLE custom user data #39"
},
{
  "input_name": "ble_user_data_4",
  "description": "BLE custom user data #4"
},
{
  "input_name": "ble_user_data_40",
  "description": "BLE custom user data #40"
},
{
  "input_name": "ble_user_data_41",
  "description": "BLE custom user data #41"
},
},
```

```
{
  "input_name": "ble_user_data_42",
  "description": "BLE custom user data #42"
},
{
  "input_name": "ble_user_data_43",
  "description": "BLE custom user data #43"
},
{
  "input_name": "ble_user_data_44",
  "description": "BLE custom user data #44"
},
{
  "input_name": "ble_user_data_45",
  "description": "BLE custom user data #45"
},
{
  "input_name": "ble_user_data_46",
  "description": "BLE custom user data #46"
},
{
  "input_name": "ble_user_data_47",
  "description": "BLE custom user data #47"
},
{
  "input_name": "ble_user_data_48",
  "description": "BLE custom user data #48"
},
{
  "input_name": "ble_user_data_49",
  "description": "BLE custom user data #49"
},
{
  "input_name": "ble_user_data_5",
  "description": "BLE custom user data #5"
},
{
  "input_name": "ble_user_data_50",
  "description": "BLE custom user data #50"
},
{
  "input_name": "ble_user_data_6",
  "description": "BLE custom user data #6"
},
{
  "input_name": "ble_user_data_7",
  "description": "BLE custom user data #7"
},
{
  "input_name": "ble_user_data_8",
  "description": "BLE custom user data #8"
},
{
  "input_name": "ble_user_data_9",
  "description": "BLE custom user data #9"
},
{
  "input_name": "board_voltage",
  "description": "Board voltage"
},
{
```

```
    "input_name": "braking",
    "description": "Braking"
  },
  {
    "input_name": "can_adblue_level",
    "description": "CAN: Level of AdBlue fluid"
  },
  {
    "input_name": "can_axle_load_1",
    "description": "CAN: Axle #1 load"
  },
  {
    "input_name": "can_axle_load_10",
    "description": "CAN: Axle #10 load"
  },
  {
    "input_name": "can_axle_load_11",
    "description": "CAN: Axle #11 load"
  },
  {
    "input_name": "can_axle_load_12",
    "description": "CAN: Axle #12 load"
  },
  {
    "input_name": "can_axle_load_13",
    "description": "CAN: Axle #13 load"
  },
  {
    "input_name": "can_axle_load_14",
    "description": "CAN: Axle #14 load"
  },
  {
    "input_name": "can_axle_load_15",
    "description": "CAN: Axle #15 load"
  },
  {
    "input_name": "can_axle_load_2",
    "description": "CAN: Axle #2 load"
  },
  {
    "input_name": "can_axle_load_3",
    "description": "CAN: Axle #3 load"
  },
  {
    "input_name": "can_axle_load_4",
    "description": "CAN: Axle #4 load"
  },
  {
    "input_name": "can_axle_load_5",
    "description": "CAN: Axle #5 load"
  },
  {
    "input_name": "can_axle_load_6",
    "description": "CAN: Axle #6 load"
  },
  {
    "input_name": "can_axle_load_7",
    "description": "CAN: Axle #7 load"
  },
  {
    "input_name": "can_axle_load_8",
```

```

        "description": "CAN: Axle #8 load"
    },
    {
        "input_name": "can_axle_load_9",
        "description": "CAN: Axle #9 load"
    },
    {
        "input_name": "can_consumption",
        "description": "CAN: Total fuel used"
    },
    {
        "input_name": "can_consumption_relative",
        "description": "CAN: Relative fuel consumption"
    },
    {
        "input_name": "can_coolant_t",
        "description": "CAN: Coolant temperature"
    },
    {
        "input_name": "can_engine_hours",
        "description": "CAN: Engine hours"
    },
    {
        "input_name": "can_engine_hours_relative",
        "description": "CAN: Relative engine hours"
    },
    {
        "input_name": "can_engine_load",
        "description": "CAN: Engine load"
    },
    {
        "input_name": "can_engine_revolutions",
        "description": "CAN: Total engine revolutions"
    },
    {
        "input_name": "can_engine_temp",
        "description": "CAN: Engine temp."
    },
    {
        "input_name": "can_fuel",
        "description": "CAN: Fuel level"
    },
    {
        "input_name": "can_fuel_2",
        "description": "CAN: Fuel level #2"
    },
    {
        "input_name": "can_fuel_economy",
        "description": "CAN: Fuel consumption"
    },
    {
        "input_name": "can_fuel_litres",
        "description": "CAN: Fuel (litres)"
    },
    {
        "input_name": "can_fuel_rate",
        "description": "CAN: Instant fuel rate"
    },
    {
        "input_name": "can_intake_air_t",
        "description": "CAN: Intake air temp."
    }

```

```

    },
    {
        "input_name": "can_mileage",
        "description": "CAN: Mileage"
    },
    {
        "input_name": "can_mileage_relative",
        "description": "CAN: Relative mileage"
    },
    {
        "input_name": "can_pto_duration",
        "description": "CAN: Total duration of PTO when stand still"
    },
    {
        "input_name": "can_pto_fuel_used",
        "description": "CAN: Total fuel used during PTO when stand
still"
    },
    {
        "input_name": "can_r_prefix",
        "description": "CAN: R-prefix"
    },
    {
        "input_name": "can_rpm",
        "description": "CAN: RPM"
    },
    {
        "input_name": "can_speed",
        "description": "CAN: Speed"
    },
    {
        "input_name": "can_throttle",
        "description": "CAN: Throttle"
    },
    {
        "input_name": "composite",
        "description": "input.composite"
    },
    {
        "input_name": "cornering",
        "description": "Cornering"
    },
    {
        "input_name": "ext_battery_voltage",
        "description": "External battery voltage"
    },
    {
        "input_name": "ext_temp_sensor_1",
        "description": "External temperature #1"
    },
    {
        "input_name": "ext_temp_sensor_10",
        "description": "External temperature #10"
    },
    {
        "input_name": "ext_temp_sensor_2",
        "description": "External temperature #2"
    },
    {
        "input_name": "ext_temp_sensor_3",
        "description": "External temperature #3"
    }

```



```
},
{
  "input_name": "ext_temp_sensor_4",
  "description": "External temperature #4"
},
{
  "input_name": "ext_temp_sensor_5",
  "description": "External temperature #5"
},
{
  "input_name": "ext_temp_sensor_6",
  "description": "External temperature #6"
},
{
  "input_name": "ext_temp_sensor_7",
  "description": "External temperature #7"
},
{
  "input_name": "ext_temp_sensor_8",
  "description": "External temperature #8"
},
{
  "input_name": "ext_temp_sensor_9",
  "description": "External temperature #9"
},
{
  "input_name": "freq_1",
  "description": "Frequency sensor #1"
},
{
  "input_name": "freq_2",
  "description": "Frequency sensor #2"
},
{
  "input_name": "freq_3",
  "description": "Frequency sensor #3"
},
{
  "input_name": "freq_4",
  "description": "Frequency sensor #4"
},
{
  "input_name": "freq_5",
  "description": "Frequency sensor #5"
},
{
  "input_name": "freq_6",
  "description": "Frequency sensor #6"
},
{
  "input_name": "freq_7",
  "description": "Frequency sensor #7"
},
{
  "input_name": "freq_8",
  "description": "Frequency sensor #8"
},
{
  "input_name": "fuel_consumption",
  "description": "Fuel consumption"
},
}
```

```
{
  "input_name": "fuel_frequency",
  "description": "LLS: Frequency"
},
{
  "input_name": "fuel_level",
  "description": "LLS: Level"
},
{
  "input_name": "fuel_temperature",
  "description": "LLS: Temperature"
},
{
  "input_name": "humidity_1",
  "description": "Relative humidity sensor #1"
},
{
  "input_name": "humidity_2",
  "description": "Relative humidity sensor #2"
},
{
  "input_name": "humidity_3",
  "description": "Relative humidity sensor #3"
},
{
  "input_name": "humidity_4",
  "description": "Relative humidity sensor #4"
},
{
  "input_name": "hw_mileage",
  "description": "Mileage"
},
{
  "input_name": "impulse_counter_1",
  "description": "Impulse counter #1"
},
{
  "input_name": "impulse_counter_2",
  "description": "Impulse counter #2"
},
{
  "input_name": "impulse_counter_3",
  "description": "Impulse counter #3"
},
{
  "input_name": "impulse_counter_4",
  "description": "Impulse counter #4"
},
{
  "input_name": "impulse_counter_5",
  "description": "Impulse counter #5"
},
{
  "input_name": "impulse_counter_6",
  "description": "Impulse counter #6"
},
{
  "input_name": "impulse_counter_7",
  "description": "Impulse counter #7"
},
{
```

```
    "input_name": "impulse_counter_8",
    "description": "Impulse counter #8"
  },
  {
    "input_name": "input_status",
    "description": "input.input_status"
  },
  {
    "input_name": "lls_level_1",
    "description": "LLS #1: Level"
  },
  {
    "input_name": "lls_level_10",
    "description": "LLS #10: Level"
  },
  {
    "input_name": "lls_level_11",
    "description": "LLS #11: Level"
  },
  {
    "input_name": "lls_level_12",
    "description": "LLS #12: Level"
  },
  {
    "input_name": "lls_level_13",
    "description": "LLS #13: Level"
  },
  {
    "input_name": "lls_level_14",
    "description": "LLS #14: Level"
  },
  {
    "input_name": "lls_level_15",
    "description": "LLS #15: Level"
  },
  {
    "input_name": "lls_level_16",
    "description": "LLS #16: Level"
  },
  {
    "input_name": "lls_level_2",
    "description": "LLS #2: Level"
  },
  {
    "input_name": "lls_level_3",
    "description": "LLS #3: Level"
  },
  {
    "input_name": "lls_level_4",
    "description": "LLS #4: Level"
  },
  {
    "input_name": "lls_level_5",
    "description": "LLS #5: Level"
  },
  {
    "input_name": "lls_level_6",
    "description": "LLS #6: Level"
  },
  {
    "input_name": "lls_level_7",
```

```
    "description": "LLS #7: Level"
  },
  {
    "input_name": "lls_level_8",
    "description": "LLS #8: Level"
  },
  {
    "input_name": "lls_level_9",
    "description": "LLS #9: Level"
  },
  {
    "input_name": "lls_level_raw_1",
    "description": "LLS #1: Level (raw)"
  },
  {
    "input_name": "lls_level_raw_2",
    "description": "LLS #2: Level (raw)"
  },
  {
    "input_name": "lls_level_raw_3",
    "description": "LLS #3: Level (raw)"
  },
  {
    "input_name": "lls_level_raw_4",
    "description": "LLS #4: Level (raw)"
  },
  {
    "input_name": "lls_level_raw_5",
    "description": "LLS #5: Level (raw)"
  },
  {
    "input_name": "lls_level_raw_6",
    "description": "LLS #6: Level (raw)"
  },
  {
    "input_name": "lls_level_raw_7",
    "description": "LLS #7: Level (raw)"
  },
  {
    "input_name": "lls_level_raw_8",
    "description": "LLS #8: Level (raw)"
  },
  {
    "input_name": "lls_temperature_1",
    "description": "LLS #1: Temperature"
  },
  {
    "input_name": "lls_temperature_10",
    "description": "LLS #10: Temperature"
  },
  {
    "input_name": "lls_temperature_11",
    "description": "LLS #11: Temperature"
  },
  {
    "input_name": "lls_temperature_12",
    "description": "LLS #12: Temperature"
  },
  {
    "input_name": "lls_temperature_13",
    "description": "LLS #13: Temperature"
  }
```

```
},
{
  "input_name": "lls_temperature_14",
  "description": "LLS #14: Temperature"
},
{
  "input_name": "lls_temperature_15",
  "description": "LLS #15: Temperature"
},
{
  "input_name": "lls_temperature_16",
  "description": "LLS #16: Temperature"
},
{
  "input_name": "lls_temperature_2",
  "description": "LLS #2: Temperature"
},
{
  "input_name": "lls_temperature_3",
  "description": "LLS #3: Temperature"
},
{
  "input_name": "lls_temperature_4",
  "description": "LLS #4: Temperature"
},
{
  "input_name": "lls_temperature_5",
  "description": "LLS #5: Temperature"
},
{
  "input_name": "lls_temperature_6",
  "description": "LLS #6: Temperature"
},
{
  "input_name": "lls_temperature_7",
  "description": "LLS #7: Temperature"
},
{
  "input_name": "lls_temperature_8",
  "description": "LLS #8: Temperature"
},
{
  "input_name": "lls_temperature_9",
  "description": "LLS #9: Temperature"
},
{
  "input_name": "obd_absolute_load_value",
  "description": "Absolute load value"
},
{
  "input_name": "obd_consumption",
  "description": "OBD: Fuel consumption"
},
{
  "input_name": "obd_control_module_voltage",
  "description": "Control module voltage"
},
{
  "input_name": "obd_coolant_t",
  "description": "OBD: Coolant temperature"
},
}
```

```
{
  "input_name": "obd_custom_fuel_litres",
  "description": "OBD: Real Fuel (litres)"
},
{
  "input_name": "obd_custom_odometer",
  "description": "OBD: Odometer"
},
{
  "input_name": "obd_engine_load",
  "description": "OBD: Engine load"
},
{
  "input_name": "obd_fuel",
  "description": "OBD: Fuel"
},
{
  "input_name": "obd_intake_air_pressure",
  "description": "OBD: Intake air pressure"
},
{
  "input_name": "obd_intake_air_t",
  "description": "OBD: Intake air temp."
},
{
  "input_name": "obd_mil_run_time",
  "description": "Time run with MIL on"
},
{
  "input_name": "obd_oil_temperature",
  "description": "OBD: Oil temperature"
},
{
  "input_name": "obd_rpm",
  "description": "OBD: RPM"
},
{
  "input_name": "obd_speed",
  "description": "OBD: Speed"
},
{
  "input_name": "obd_throttle",
  "description": "OBD: Throttle"
},
{
  "input_name": "obd_time_since_engine_start",
  "description": "Run time since engine start"
},
{
  "input_name": "passengers_entered_1",
  "description": "Passenger counter #1: Entry"
},
{
  "input_name": "passengers_entered_2",
  "description": "Passenger counter #2: Entry"
},
{
  "input_name": "passengers_entered_3",
  "description": "Passenger counter #3: Entry"
},
{
```

```

    "input_name": "passengers_entered_4",
    "description": "Passenger counter #4: Entry"
  },
  {
    "input_name": "passengers_entered_5",
    "description": "Passenger counter #5: Entry"
  },
  {
    "input_name": "passengers_entered_6",
    "description": "Passenger counter #6: Entry"
  },
  {
    "input_name": "passengers_entered_7",
    "description": "Passenger counter #7: Entry"
  },
  {
    "input_name": "passengers_entered_8",
    "description": "Passenger counter #8: Entry"
  },
  {
    "input_name": "passengers_exit_1",
    "description": "Passenger counter #1: Exit"
  },
  {
    "input_name": "passengers_exit_2",
    "description": "Passenger counter #2: Exit"
  },
  {
    "input_name": "passengers_exit_3",
    "description": "Passenger counter #3: Exit"
  },
  {
    "input_name": "passengers_exit_4",
    "description": "Passenger counter #4: Exit"
  },
  {
    "input_name": "passengers_exit_5",
    "description": "Passenger counter #5: Exit"
  },
  {
    "input_name": "passengers_exit_6",
    "description": "Passenger counter #6: Exit"
  },
  {
    "input_name": "passengers_exit_7",
    "description": "Passenger counter #7: Exit"
  },
  {
    "input_name": "passengers_exit_8",
    "description": "Passenger counter #8: Exit"
  },
  {
    "input_name": "physiologic_blood_pressure_dt",
    "description": "Diastolic blood pressure"
  },
  {
    "input_name": "physiologic_blood_pressure_st",
    "description": "Systolic blood pressure"
  },
  {
    "input_name": "physiologic_heart_rate",

```

```
        "description": "Heart rate"
    },
    {
        "input_name": "raw_can_1",
        "description": "CAN: Raw data #1"
    },
    {
        "input_name": "raw_can_10",
        "description": "CAN: Raw data #10"
    },
    {
        "input_name": "raw_can_11",
        "description": "CAN: Raw data #11"
    },
    {
        "input_name": "raw_can_12",
        "description": "CAN: Raw data #12"
    },
    {
        "input_name": "raw_can_13",
        "description": "CAN: Raw data #13"
    },
    {
        "input_name": "raw_can_14",
        "description": "CAN: Raw data #14"
    },
    {
        "input_name": "raw_can_15",
        "description": "CAN: Raw data #15"
    },
    {
        "input_name": "raw_can_16",
        "description": "CAN: Raw data #16"
    },
    {
        "input_name": "raw_can_2",
        "description": "CAN: Raw data #2"
    },
    {
        "input_name": "raw_can_3",
        "description": "CAN: Raw data #3"
    },
    {
        "input_name": "raw_can_4",
        "description": "CAN: Raw data #4"
    },
    {
        "input_name": "raw_can_5",
        "description": "CAN: Raw data #5"
    },
    {
        "input_name": "raw_can_6",
        "description": "CAN: Raw data #6"
    },
    {
        "input_name": "raw_can_7",
        "description": "CAN: Raw data #7"
    },
    {
        "input_name": "raw_can_8",
        "description": "CAN: Raw data #8"
```



```
},
{
  "input_name": "raw_can_9",
  "description": "CAN: Raw data #9"
},
{
  "input_name": "rs232_1",
  "description": "RS-232 #1"
},
{
  "input_name": "rs232_2",
  "description": "RS-232 #2"
},
{
  "input_name": "rs232_3",
  "description": "RS-232 #3"
},
{
  "input_name": "rs232_4",
  "description": "RS-232 #4"
},
{
  "input_name": "rs232_5",
  "description": "RS-232 #5"
},
{
  "input_name": "rs232_6",
  "description": "RS-232 #6"
},
{
  "input_name": "tacho_mileage",
  "description": "TACH0: Mileage"
},
{
  "input_name": "tacho_speed",
  "description": "TACH0: Speed"
},
{
  "input_name": "temp_sensor",
  "description": "Temperature"
},
{
  "input_name": "tire_pressure_1",
  "description": "Tire pressure #1"
},
{
  "input_name": "tire_pressure_10",
  "description": "Tire pressure #10"
},
{
  "input_name": "tire_pressure_11",
  "description": "Tire pressure #11"
},
{
  "input_name": "tire_pressure_12",
  "description": "Tire pressure #12"
},
{
  "input_name": "tire_pressure_13",
  "description": "Tire pressure #13"
},
}
```

```
{
  "input_name": "tire_pressure_14",
  "description": "Tire pressure #14"
},
{
  "input_name": "tire_pressure_15",
  "description": "Tire pressure #15"
},
{
  "input_name": "tire_pressure_16",
  "description": "Tire pressure #16"
},
{
  "input_name": "tire_pressure_17",
  "description": "Tire pressure #17"
},
{
  "input_name": "tire_pressure_18",
  "description": "Tire pressure #18"
},
{
  "input_name": "tire_pressure_19",
  "description": "Tire pressure #19"
},
{
  "input_name": "tire_pressure_2",
  "description": "Tire pressure #2"
},
{
  "input_name": "tire_pressure_20",
  "description": "Tire pressure #20"
},
{
  "input_name": "tire_pressure_21",
  "description": "Tire pressure #21"
},
{
  "input_name": "tire_pressure_22",
  "description": "Tire pressure #22"
},
{
  "input_name": "tire_pressure_23",
  "description": "Tire pressure #23"
},
{
  "input_name": "tire_pressure_24",
  "description": "Tire pressure #24"
},
{
  "input_name": "tire_pressure_25",
  "description": "Tire pressure #25"
},
{
  "input_name": "tire_pressure_26",
  "description": "Tire pressure #26"
},
{
  "input_name": "tire_pressure_27",
  "description": "Tire pressure #27"
},
{
```

```
    "input_name": "tire_pressure_28",
    "description": "Tire pressure #28"
  },
  {
    "input_name": "tire_pressure_29",
    "description": "Tire pressure #29"
  },
  {
    "input_name": "tire_pressure_3",
    "description": "Tire pressure #3"
  },
  {
    "input_name": "tire_pressure_30",
    "description": "Tire pressure #30"
  },
  {
    "input_name": "tire_pressure_4",
    "description": "Tire pressure #4"
  },
  {
    "input_name": "tire_pressure_5",
    "description": "Tire pressure #5"
  },
  {
    "input_name": "tire_pressure_6",
    "description": "Tire pressure #6"
  },
  {
    "input_name": "tire_pressure_7",
    "description": "Tire pressure #7"
  },
  {
    "input_name": "tire_pressure_8",
    "description": "Tire pressure #8"
  },
  {
    "input_name": "tire_pressure_9",
    "description": "Tire pressure #9"
  },
  {
    "input_name": "tire_temperature_1",
    "description": "Tire air temperature #1"
  },
  {
    "input_name": "tire_temperature_10",
    "description": "Tire air temperature #10"
  },
  {
    "input_name": "tire_temperature_11",
    "description": "Tire air temperature #11"
  },
  {
    "input_name": "tire_temperature_12",
    "description": "Tire air temperature #12"
  },
  {
    "input_name": "tire_temperature_13",
    "description": "Tire air temperature #13"
  },
  {
    "input_name": "tire_temperature_14",
```

```
    "description": "Tire air temperature #14"
  },
  {
    "input_name": "tire_temperature_15",
    "description": "Tire air temperature #15"
  },
  {
    "input_name": "tire_temperature_16",
    "description": "Tire air temperature #16"
  },
  {
    "input_name": "tire_temperature_17",
    "description": "Tire air temperature #17"
  },
  {
    "input_name": "tire_temperature_18",
    "description": "Tire air temperature #18"
  },
  {
    "input_name": "tire_temperature_19",
    "description": "Tire air temperature #19"
  },
  {
    "input_name": "tire_temperature_2",
    "description": "Tire air temperature #2"
  },
  {
    "input_name": "tire_temperature_20",
    "description": "Tire air temperature #20"
  },
  {
    "input_name": "tire_temperature_21",
    "description": "Tire air temperature #21"
  },
  {
    "input_name": "tire_temperature_22",
    "description": "Tire air temperature #22"
  },
  {
    "input_name": "tire_temperature_23",
    "description": "Tire air temperature #23"
  },
  {
    "input_name": "tire_temperature_24",
    "description": "Tire air temperature #24"
  },
  {
    "input_name": "tire_temperature_25",
    "description": "Tire air temperature #25"
  },
  {
    "input_name": "tire_temperature_26",
    "description": "Tire air temperature #26"
  },
  {
    "input_name": "tire_temperature_27",
    "description": "Tire air temperature #27"
  },
  {
    "input_name": "tire_temperature_28",
    "description": "Tire air temperature #28"
```

```
},
{
  "input_name": "tire_temperature_29",
  "description": "Tire air temperature #29"
},
{
  "input_name": "tire_temperature_3",
  "description": "Tire air temperature #3"
},
{
  "input_name": "tire_temperature_30",
  "description": "Tire air temperature #30"
},
{
  "input_name": "tire_temperature_4",
  "description": "Tire air temperature #4"
},
{
  "input_name": "tire_temperature_5",
  "description": "Tire air temperature #5"
},
{
  "input_name": "tire_temperature_6",
  "description": "Tire air temperature #6"
},
{
  "input_name": "tire_temperature_7",
  "description": "Tire air temperature #7"
},
{
  "input_name": "tire_temperature_8",
  "description": "Tire air temperature #8"
},
{
  "input_name": "tire_temperature_9",
  "description": "Tire air temperature #9"
},
{
  "input_name": "uds_adblue_tanklevel_absolut",
  "description": "UDS: Level of AdBlue fluid"
},
{
  "input_name": "uds_adblue_tanklevel_percent",
  "description": "UDS: Level of AdBlue fluid (percent)"
},
{
  "input_name": "uds_ambient_air_temp",
  "description": "UDS: Ambient air temperature"
},
{
  "input_name": "uds_battery_state_percent",
  "description": "UDS: Battery level"
},
{
  "input_name": "uds_battery_voltage",
  "description": "UDS: Battery voltage"
},
{
  "input_name": "uds_consumption",
  "description": "UDS: Fuel consumption"
},
}
```

```

{
  "input_name": "uds_consumption_average",
  "description": "UDS: Average fuel consumption"
},
{
  "input_name": "uds_consumption_average_high",
  "description": "UDS: Average high fuel consumption"
},
{
  "input_name": "uds_consumption_average_low",
  "description": "UDS: Average low fuel consumption"
},
{
  "input_name": "uds_consumption_since_reset",
  "description": "UDS: Fuel consumption since reset"
},
{
  "input_name": "uds_eco_co2_score",
  "description": "UDS: Environmental score"
},
{
  "input_name": "uds_engine_coolant_temp",
  "description": "UDS: Coolant temperature"
},
{
  "input_name": "uds_engine_oil_temperature",
  "description": "UDS: Oil temperature"
},
{
  "input_name": "uds_fuel_tank_level_absolute",
  "description": "UDS: Fuel level"
},
{
  "input_name": "uds_fuel_tank_level_percent",
  "description": "UDS: Fuel level (percent)"
},
{
  "input_name": "uds_odometer",
  "description": "UDS: Odometer"
},
{
  "input_name": "uds_rpm",
  "description": "UDS: RPM"
},
{
  "input_name": "uds_service_days_since_last",
  "description": "UDS: Days since last service"
},
{
  "input_name": "uds_service_distance_snc_lst",
  "description": "UDS: Km since last service"
},
{
  "input_name": "uds_service_interval_days",
  "description": "UDS: Days till next service"
},
{
  "input_name": "uds_service_interval_distance",
  "description": "UDS: Distance to drive till next service"
},
{

```

```

    "input_name": "uds_service_max_days",
    "description": "UDS: Max days of service interval"
  },
  {
    "input_name": "uds_service_max_distance",
    "description": "UDS: Max km of service interval"
  },
  {
    "input_name": "uds_speed",
    "description": "UDS: Speed"
  },
  {
    "input_name": "uds_steer_angle",
    "description": "UDS: Steer angle"
  },
  {
    "input_name": "uds_tank_level_cng_percent",
    "description": "UDS: Cng fuel tank level (percent)"
  },
  {
    "input_name": "uds_throttle",
    "description": "UDS: Throttle"
  },
  {
    "input_name": "uds_time_since_engine_start",
    "description": "UDS: Run time since engine start"
  },
  {
    "input_name": "uds_tire_pressure_front_left",
    "description": "UDS: Tire pressure front left"
  },
  {
    "input_name": "uds_tire_pressure_front_right",
    "description": "UDS: Tire pressure front right"
  },
  {
    "input_name": "uds_tire_pressure_rear_left",
    "description": "UDS: Tire pressure rear left"
  },
  {
    "input_name": "uds_tire_pressure_rear_right",
    "description": "UDS: Tire pressure rear right"
  },
  {
    "input_name": "user_data_1",
    "description": "Custom user data #1"
  },
  {
    "input_name": "user_data_10",
    "description": "Custom user data #10"
  },
  {
    "input_name": "user_data_11",
    "description": "Custom user data #11"
  },
  {
    "input_name": "user_data_12",
    "description": "Custom user data #12"
  },
  {
    "input_name": "user_data_13",

```

```
    "description": "Custom user data #13"
  },
  {
    "input_name": "user_data_14",
    "description": "Custom user data #14"
  },
  {
    "input_name": "user_data_15",
    "description": "Custom user data #15"
  },
  {
    "input_name": "user_data_16",
    "description": "Custom user data #16"
  },
  {
    "input_name": "user_data_17",
    "description": "Custom user data #17"
  },
  {
    "input_name": "user_data_18",
    "description": "Custom user data #18"
  },
  {
    "input_name": "user_data_19",
    "description": "Custom user data #19"
  },
  {
    "input_name": "user_data_2",
    "description": "Custom user data #2"
  },
  {
    "input_name": "user_data_20",
    "description": "Custom user data #20"
  },
  {
    "input_name": "user_data_21",
    "description": "Custom user data #21"
  },
  {
    "input_name": "user_data_22",
    "description": "Custom user data #22"
  },
  {
    "input_name": "user_data_23",
    "description": "Custom user data #23"
  },
  {
    "input_name": "user_data_24",
    "description": "Custom user data #24"
  },
  {
    "input_name": "user_data_25",
    "description": "Custom user data #25"
  },
  {
    "input_name": "user_data_26",
    "description": "Custom user data #26"
  },
  {
    "input_name": "user_data_27",
    "description": "Custom user data #27"
```



```

    },
    {
      "input_name": "user_data_28",
      "description": "Custom user data #28"
    },
    {
      "input_name": "user_data_29",
      "description": "Custom user data #29"
    },
    {
      "input_name": "user_data_3",
      "description": "Custom user data #3"
    },
    {
      "input_name": "user_data_30",
      "description": "Custom user data #30"
    },
    {
      "input_name": "user_data_31",
      "description": "Custom user data #31"
    },
    {
      "input_name": "user_data_32",
      "description": "Custom user data #32"
    },
    {
      "input_name": "user_data_4",
      "description": "Custom user data #4"
    },
    {
      "input_name": "user_data_5",
      "description": "Custom user data #5"
    },
    {
      "input_name": "user_data_6",
      "description": "Custom user data #6"
    },
    {
      "input_name": "user_data_7",
      "description": "Custom user data #7"
    },
    {
      "input_name": "user_data_8",
      "description": "Custom user data #8"
    },
    {
      "input_name": "user_data_9",
      "description": "Custom user data #9"
    }
  ]
}

```

## errors

General types only.

Last update: November 22, 2023





# Tracker settings actions

Contains API calls to get and change tracker's label and group.

## API actions

API base path: `/tracker/settings`.

### read

Gets base settings for the specified tracker.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/settings/read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/settings/read?  
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456
```

#### response

```
{  
  "success": true,  
  "settings": {  
    "label": "Courier",  
    "group_id": 1  
  }  
}
```

- `label` - string. User-defined label for this tracker, e.g. "Courier".

- `group_id` - int. Tracker group id. 0 if tracker does not belong to any group.

#### errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

## update

Updates the settings of the specified tracker.

**required sub-user rights:** `tracker_update`.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
group_id	Tracker group ID. 0 if tracker does not belong to any group. The specified group must exist.	int	1
label	User-defined label for this tracker, e.g. "Courier". Must consist of printable characters and have length between 1 and 60. Cannot contain <code>&lt;</code> and <code>&gt;</code> symbols.	string	"Courier"

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/settings/update' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":  
123456, "group_id": 1, "label": "Courier"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/settings/update?  
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456&group_id=1&lab
```

#### response

```
{ "success": true }
```

**errors**

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 204 – Entity not found - if there is no group with the specified group id.

Last update: March 28, 2023







# LBS settings

Contains API calls for reading and changing LBS settings. It is responsible for the LBS detection radius portlet in devices and settings tab in the UI. LBS (Location-based service) technology allows to determine the tracker's location without using standard location services such as GPS, GLONASS, Galileo or Beidou. LBS locates the position using cellular base stations or Wi-Fi access points.

## API actions

API base path: `/tracker/settings/lbs`.

### read

Gets LBS settings for the specified tracker.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/settings/lbs/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/settings/lbs/read?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456
```

#### response

```
{
  "success": true,
```

```
"max_radius": 300
}
```

- `max_radius` - int. Max allowed radius for LBS points in meters. Min=0, max=10000.

#### errors

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

## update

Updates LBS settings for the specified tracker.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
max_radius	Max allowed radius for LBS points in meters. Min=0, max=10000.	int	1000

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/settings/lbs/update' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456, "max_radius": 1000}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/settings/lbs/update?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456&max_radius=100
```

#### response

```
{ "success": true }
```

**errors**

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

Last update: December 26, 2022





# Tracking mode

API calls for reading and setting data transmission and operating modes of devices. It is responsible for the tracking mode portlet in devices and settings tab in the UI. The list of settings can vary depending on model of the used tracker, the principle of its work and its functionality.

## API actions

API base path: `/tracker/settings/tracking`.

### read

Gets tracking settings for the specified tracker.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/settings/tracking/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/settings/tracking/read?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456
```

#### response

Returned fields may differ from model to model. See tracking profiles for more information.

```
{
  "success": true,
  "value" : {<tracking settings>}
}
```

## errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 214 – Requested operation or parameters are not supported by the device - if device model has no tracking settings at all.

## update

Sends new tracking settings to the specified tracker.

**required sub-user rights:** `tracker_configure`.

## parameters

name	description	type
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int
tracking_settings	Set of fields which differ from model to model. See <a href="#">tracking profiles</a> for more information.	JSON object

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/settings/tracking/
update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
123456, "tracking_settings": {"tracking_angle": 30,
"tracking_distance": 100, "tracking_interval": 60,
"on_stop_tracking_interval": 180, "sleep_mode": "disabled",
"stop_detection": "ignition"}}'
```



## response

Returned fields may differ from model to model. See tracking profiles for more information.

```
{ "success": true }
```

## errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 214 – Requested operation or parameters are not supported by the device - if device model has no tracking settings at all.
- 219 – Not allowed for clones of the device - if specified tracker is clone of another tracker.

Last update: December 26, 2022





# Tracking profiles

Contains tracking profiles of all device models with description.

## albatross\_s6

Tracking profile for Albatross S6.

```
{
  "tracking_interval": 30,
  "tracking_distance": 100
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=65535.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=65535.

## albatross\_s8\_5

Tracking profile for Albatross S8.5.

```
{
  "tracking_interval": 30,
  "psm_interval": 60000,
  "psm_mode": 0
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=65535.
- `psm_interval` - optional int. Duration in seconds for the device to stay in the deep sleep mode. Min=600, max=65535.
- `psm_mode` - int. Define the sleep level. Min=0, max=1.

## apkcom

Tracking profile for АПК КОМ ASC-2 GLONASS/GPS, АПК КОМ ASC-6 GLONASS/GPS, АПК КОМ ASC-7, АПК КОМ ASC-8.

---

```
{
  "tracking_angle": 30,
  "tracking_interval": 30,
  "tracking_distance": 100
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=5, max=180.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=300.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=5000.

## arknav\_x8

Tracking profile for Arknav RX8.

```
{
  "tracking_angle": 30,
  "tracking_interval": 60,
  "tracking_distance": 150
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=180.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=65534.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=65534.

## arnavi2

Arnavi 2 tracking profile.

```
{
  "tracking_angle": 30,
  "tracking_distance": 100,
  "min_tracking_interval": 30,
  "max_tracking_interval": 300,
  "speed_change": 50,
  "freeze_by_speed": false,
  "freeze_by_motion": true,
}
```

```
"freeze_by_ignition": false
}
```

\* `tracking_angle` – int. Degrees 10-255, the device will send tracking data when course changing more than defined value. \* `tracking_distance` – int. Distance in meters 50-65535, e.g. 100 means that the device will send data every 100 meters. \* `min_tracking_interval` – int. Min interval in seconds 30-255, e.g. 30 means that the device will send tracking data no more frequently than every 30 seconds. \* `max_tracking_interval` – int. Max interval in seconds 30-65535, e.g. 30 means that the device will send tracking data not less frequently than every 30 seconds. \* `speed_change` – int. Kph 3-255, the device will send tracking data when speed changing more than defined value. \* `freeze_by_speed` – boolean. Freeze coordinates when speed is less than 2kph. \* `freeze_by_motion` – boolean. Freeze coordinates when motion sensor detects no motion. \* `freeze_by_ignition` – boolean. Freeze coordinates when ignition is OFF.

## arnavi4

Tracking profile for Arnavi 4, Arnavi 5, Arnavi Integral, Arnavi Integral-2, Arnavi Integral-3.

```
{
  "max_tracking_interval": 60,
  "min_tracking_interval": 5,
  "speed_change": 10,
  "tracking_angle": 30,
  "tracking_distance": 150
}
```

- `max_tracking_interval` – int. Max interval in seconds 30-65535, e.g. 30 means that the device will send tracking data not less frequently than every 30 seconds.
- `min_tracking_interval` – int. Min interval in seconds 30-255, e.g. 30 means that the device will send tracking data no more frequently than every 30 seconds.
- `speed_change` – int. Kph 3-255, the device will send tracking data when speed changing more than defined value.
- `tracking_angle` – int. Degrees 10-255, the device will send tracking data when course changing more than defined value.
- `tracking_distance` – int. Distance in meters 50-65535, e.g. 100 means that the device will send data every 100 meters.

## atlanta

Tracking profile for Atlanta L-100, Atlanta O-300, Atlanta PT-100, Atlanta W-track, Atlanta WP-30C.

```
{
  "tracking_distance": 150,
  "tracking_interval": 60
}
```

- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=65534.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=18000.

## atlanta\_pt100

Tracking profile for Atlanta PT-100.

```
{
  "tracking_interval": 300
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=300, max=18000.

## atrack

ATrack tracking profile.

```
{
  "control_mode": "acc",
  "tracking_interval": 30,
  "tracking_distance": 150,
  "tracking_angle": 30,
  "psm_mode": 0,
  "psm_interval": 30,
  "on_stop_tracking_interval": 1
}
```

\* `control_mode` - optional [enum](#). Mode of tracking by the ACC or engine status. Can be "acc" | "engine\_status". \* `tracking_interval` - optional int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=65535x10, default=300. \* `tracking_distance` - optional int. Distance in meters,

e.g. 100 means that the device will send data every 100 meters. Min=50, max=65535, default=100. \* `tracking_angle` - optional int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=80, default=10. \* `psm_mode` - optional int. Define the sleep level, 0 – no sleeping, 1- light sleep (GPS Off, GPRS On, G-sensor On), 2- deep sleep (GPS Off, GPRS Off, G-sensor On). Min=0, max=2, default=0. \* `psm_interval` - optional int. Duration in seconds for the device to stay in the deep sleep mode. Min=30, max=65535x60, default=90x60. \* `on_stop_tracking_interval` - int. Minimum time in seconds that must elapse before reporting next position while the ACC is in Off status. "acc" in control\_mode must be set in order to use this time interval. Min=1, max=65535x10, default=15x60.

## autofon

Autofon profile.

```
{
  "type": "interval",
  "tracking_interval": 30,
  "online_on_ext_power": true,
  "timer1_time" : "2020-09-16 03:17:26",
  "timer1_interval": 15,
  "timer2_time" : "2020-09-18 03:17:26",
  "timer2_interval": 30
}
```

- `type` - [enum](#). Tracking type "interval" or "power\_save".
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=240.
- `online_on_ext_power` - boolean. Connect to server when external power connected.
- `timer1_time` - date/time. Date/time for timer1 for checking incoming SMS commands.
- `timer1_interval` - int. Interval to wakeup for timer1, minutes, min=15.
- `timer2_time` - date/time. Date/time for timer2 for sending location.
- `timer2_interval` - int. Interval to wakeup for timer1, minutes, min=15.

## autoleaders\_st901

Tracking profile for Auto Leaders ST-901, Auto Leaders ST-901M.



```
{
  "psm_interval": 60,
  "psm_mode": 0,
  "tracking_interval": 30
}
```

- `psm_interval` - int. Duration in seconds for the device to stay in the deep sleep mode. Min=30, max=18000.
- `psm_mode` - int. Define the sleep level. Min=0, max=1.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=18000.

## autoseeker\_at17

Tracking profile for Autoseeker AT-17.

```
{
  "tracking_interval": 30
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=1, max=18000.

## avlsat\_neos

Tracking profile for AVLSAT NEO-S.

```
{
  "tracking_interval": 60
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=60, max=599940.

## bitrek310

Tracking profile for BI 310 CICADA, NaviTrek 310 Cicada.

```
{
  "psm_interval": 12000,
  "psm_mode": 0,
}
```

```
"tracking_interval": 720
}
```

- `psm_interval` - int. Duration in seconds for the device to stay in the deep sleep mode. Min=300, max=86400.
- `psm_mode` - int. Define the sleep level. Min=0, max=1.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=720, max=21600.

## bofan\_pt521

Tracking profile for Bofan PT502, Bofan PT521.

```
{
  "tracking_angle": 30,
  "tracking_distance": 100,
  "tracking_interval": 60,
  "type": "interval"
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=90.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=5000.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=1200.
- `type` - enum. Can be "interval" | "distance" | "power\_save" | "distance\_interval\_angle" | "interval\_angle" | "intelligent".

## box

Tracking profile for BOX-tracker, BOXtracker 2, Galileosky Boxfinder v1.0.

```
{
  "tracking_angle": 30,
  "tracking_interval": 120
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=359.

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=4294968.

## boxfinder

Tracking profile for Galileosky Boxfinder v1.0.

```
{  
  "shock_value": 1.5,  
  "sleep_timeout": 180  
}
```

- `shock_value` - double. Can be min=0.5, max=4 g.
- `sleep_timeout` - int. Can be min=1, max=1440 minutes.

## bsj

Tracking profile for BSJ KM-01/02, Gosafe G1C.

```
{  
  "tracking_angle": 30,  
  "tracking_interval": 150  
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=5, max=180.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.

## c2stek\_fl

Tracking profile for C2STEK FL10, C2STEK FL2000G.

```
{  
  "tracking_angle": 30,  
  "tracking_distance": 300,  
  "tracking_interval": 120  
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=0, max=360.

- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=0, max=9999.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=0, max=9999.

## calamp

Tracking profile for CalAmp ATU-620, CalAmp LMU-1100, CalAmp LMU-1200, CalAmp LMU-200, CalAmp LMU-2030, CalAmp LMU-2600, CalAmp LMU-2630, CalAmp LMU-2720, CalAmp LMU-300, CalAmp LMU-3030, CalAmp LMU-3640, CalAmp LMU-400, CalAmp LMU-4200, CalAmp LMU-4230, CalAmp LMU-4520, CalAmp LMU-5530, CalAmp LMU-700, CalAmp LMU-800, CalAmp LMU-900, CalAmp TTU-1200, CalAmp TTU-2830, CalAmp TTU-700.

```
{
  "psm_interval": 600,
  "tracking_angle": 30,
  "tracking_distance": 200,
  "tracking_interval": 60
}
```

- `psm_interval` - int. Duration in seconds for the device to stay in the deep sleep mode. Min=30, max=86400.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=5, max=180.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=5000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.

## cantrack\_t80

Tracking profile for Cantrack T80.

```
{
  "tracking_interval": 10
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=1000.

## careu

Tracking profile for CAREU U1 Lite Plus, CAREU U1 Plus, CAREU UT1, CAREU UW1, CAREU Ucan, CAREU Ueco, CAREU Ugo, IntelliTrac A1, Intellitrac S1.

```
{
  "tracking_angle": 45,
  "tracking_distance": 50,
  "tracking_interval": 20
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=5, max=180.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=25, max=50000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=65535.

## cargo

Tracking profile for Cargo Light 2, Cargo Mini 2, Cargo Pro 2.

```
{
  "psm_interval": 600,
  "tracking_angle": 60,
  "tracking_distance": 100,
  "tracking_interval": 30
}
```

- `psm_interval` - int. Duration in seconds for the device to stay in the deep sleep mode. Min=30, max=86400.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=5, max=180.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=5000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.

## carscop\_cctr800

Tracking profile for Carscop CCTR-808S, Carscop CCTR-809.

```
{
  "psm_interval": 3600,
  "psm_mode": 1,
  "tracking_interval": 30
}
```

- `psm_interval` - int. Duration in seconds for the device to stay in the deep sleep mode. Min=3600, max=432000.
- `psm_mode` - int. Define the sleep level. Min=0, max=1.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=999.

## carscop\_cctr830

Tracking profile for Carscop CCTR-830, Toptracking CCTR-830G.

```
{
  "tracking_interval": 40
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=999.

## castel\_idd

Tracking profile for Sinocastel IDD-213.

```
{
  "sleep_report_interval": 120,
  "tracking_angle": 20,
  "tracking_distance": 500,
  "tracking_interval": 200,
  "upload_records_count": 1
}
```

- `sleep_report_interval` - int. Interval in minutes, e.g. 10 means that the device will send tracking data every 10 minutes in a sleep mode. Min=10, max=1440.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=90.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=5000.

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=600.
- `upload_records_count` - int. Count of uploaded records. Min =1, max=10.

## castel\_interval

Tracking profile for Sinocastel MPIP-620, Sinocastel PT-690, Sinocastel PT-718S.

```
{
  "tracking_interval": 60
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=18000.

## cguard

cGuard tracking profile.

```
{
  "tracking_interval": 60,
  "tracking_distance": 100,
  "tracking_angle" : 15,
  "psm_interval": 300
}
```

\* `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=0, max=65535, default=60. \* `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=0, max=65535, default=100. \* `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=0, max=180, default=15. \* `psm_interval` - int. Duration in seconds for the device to stay in the deep sleep mode. Min=0, max=65535, default=300.

## cguard\_asset

cGuard tracking profile for asset trackers. name: 'cguard\_asset'

```
{
  "tracking_interval": 60,
  "tracking_distance": 100,
  "tracking_angle" : 45,
```

```

    "psm_interval": 300,
    "mode": "ASSET",
    "wakeup_type": "PERIODICAL",
    "wakeup_day": "EVERYDAY",
    "wakeup_time": "12:00",
    "wakeup_period": 1440,
    "moving_detection": true
}

```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=0, max=65535, default=60.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=0, max=65535, default=100.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=0, max=180, default=45.
- `psm_interval` - int. Duration in seconds for the device to stay in the deep sleep mode. Min=0, max=65535, default=300.
- `mode` - **enum**. Device working mode. `TRACKER` means that device work in the continuous mode. `ASSET` means that device work in the periodical mode and wakes up on schedule or by period.
- `wakeup_type` - **enum**. Can be "SCHEDULED" | "PERIODICAL". How device wakes up in `ASSET` mode. default="PERIODICAL".
- `wakeup_day` - **enum**. Can be "EVERYDAY" | "MONDAY" | "TUESDAY" | "WEDNESDAY" | "THURSDAY" | "FRIDAY" | "SATURDAY" | "SUNDAY", default="EVERYDAY". What day to wake up if `wakeup_type` = `SCHEDULED`.
- `wakeup_time` - string. What time in minutes to wake up if `wakeup_type` = `SCHEDULED`. Format `HH:mm`, default="12:00"
- `wakeup_period` - int. Wakeup period in minutes. Min=15, max=65535, default=1440. Required if `wakeup_type` = `PERIODICAL`
- `moving_detection` - boolean. If `true` means that device will be wakes up at the beginning of the movement. Required if `mode` == 'ASSET'

## concox\_distance\_interval

Tracking profile for Concox X3.

```

{
    "tracking_distance": 100,

```



```

    "tracking_interval": 30
}

```

- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=100, max=10000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=18000.

## concox\_gt350

Tracking profile for Concox GT350.

```

{
    "psm_interval": 600,
    "psm_mode": 1,
    "tracking_interval": 10
}

```

- `psm_interval` - int. Duration in seconds for the device to stay in the deep sleep mode. Min=600, max=432000.
- `psm_mode` - int. Define the sleep level. Min=0, max=1.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=1800.

## concox\_gt700

Tracking profile for Concox AT3, Concox AT4, Concox GT710.

```

{
    "psm_interval": 2,
    "tracking_interval": 1,
    "type": "interval",
    "wakeup_time": "10:20"
}

```

- `psm_interval` - int. Duration in hours for the device to stay in the deep sleep mode. Min=1, max=24. Valid values are 1, 2, 3, 4, 6, 8, 12, 24.
- `tracking_interval` - int. Interval in minutes. Min=1, max=30.
- `type` - [enum](#). Can be "interval" | "distance" | "power\_save" | "distance\_interval\_angle" | "interval\_angle" | "intelligent".
- `wakeup_time` - string. Format `hh:mm`.

## concox\_interval

Tracking profile for Concox GK309 , Concox GS503, Concox GT03A, Concox GT03C, Concox WeTrack Lite, Concox WeTrack2, Jimi JI09.

```
{
  "tracking_interval": 30
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=18000.

## concox\_jv200

Tracking profile for Concox JV200.

```
{
  "tracking_interval": 30
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=18000.

## concox\_qbit

Tracking profile for Concox QBIT.

```
{
  "gps_tracking_interval": 10,
  "lbs_tracking_interval": 60,
  "mode": "lbs"
}
```

- `gps_tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds in `gps` mode. Min=30, max=18000.
- `lbs_tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds in `lbs` mode. Min=30, max=18000.
- `mode` - string. Can be "lbs" | "gps".

## concoxgt02

Tracking profile for Concox GT02 / TR02.

```
{
  "tracking_interval": 30
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=18000.

## concoxgt06

Tracking profile for Concox GV20, Concox X1, Protrack VT05.

```
{
  "psm_interval": 3000,
  "tracking_angle": 120,
  "tracking_distance": 250,
  "tracking_interval": 30,
  "type": "intelligent"
}
```

- `psm_interval` - int. Duration in seconds for the device to stay in the deep sleep mode. Min=30, max=65535.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=0, max=180.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=10000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=65535.
- `type` - [enum](#). Can be "interval" | "distance" | "power\_save" | "distance\_interval\_angle" | "interval\_angle" | "intelligent".

## default

Default tracking profile.

```
{
  "type": "interval",
  "tracking_interval": 30,
}
```

```
"tracking_distance": 100
}
```

- `type` - `enum`. Can be "interval" (send tracking data based on time intervals) or "distance" (send tracking data after passing specified distance).
- `tracking_interval` - `int`. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds.
- `tracking_distance` - `int`. Distance in meters, e.g. 100 means that the device will send data every 100 meters.

## default\_angle

Default profile with optional angle-based tracking.

```
{
  "type": "distance",
  "tracking_interval": 30,
  "tracking_distance": 100,
  "tracking_angle" : 30
}
```

- `type` - `enum`. Can be "interval" (send tracking data based on time intervals) or "distance" (send tracking data after passing specified distance).
- `tracking_interval` - `int`. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds.
- `tracking_distance` - `int`. Distance in meters, e.g. 100 means that the device will send data every 100 meters.
- `tracking_angle` - optional `int`. If specified, the device will additionally send data when it changes direction to specified angle, e.g. 30 degrees.

## default\_powersave

Default powersave profile with optional angle-based tracking.

```
{
  "type": "power_save",
  "tracking_interval": 30,
  "tracking_distance": 100,
  "tracking_angle" : 30,
  "psm_interval": 65535,
  "psm_mode": 2
}
```

- `type` - `enum`. Can be "interval" (send tracking data based on time intervals) or "distance" (send tracking data after passing specified distance).
- `tracking_interval` - optional int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds.
- `tracking_distance` - optional int. Distance in meters, e.g. 100 means that the device will send data every 100 meters.
- `tracking_angle` - optional int. If specified, the device will additionally send data when it changes direction to specified angle, e.g. 30 degrees.
- `psm_interval` - optional int. Define the time interval in seconds (60-65535) which the unit stays in the sleeping state when type= `power_save` .
- `psm_mode` - optional int. Define the sleep level when type != `power_save` , 0 - no sleeping, 1 - light sleep(GPS Off, GPRS On, G-sensor On), 2 - deep sleep(GPS Off, GPRS Off, G-sensor On).

## defenstar\_007

Tracking profile for Defenstar DS007.

```
{
  "tracking_interval": 65534
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=60, max=65534.

## defenstar\_008

Tracking profile for Defenstar DS008, Gubloos GPS-S1.

```
{
  "tracking_interval": 1000
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=9999.

## digitalsystems\_dsf22

Tracking profile for DigitalSystems DSF22.

```
{
  "tracking_angle": 10,
  "tracking_interval": 120
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=359.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.

## distance\_interval

Tracking profile with distance and interval.

```
{
  "tracking_distance": 250,
  "tracking_interval": 3600
}
```

- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=100000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.

## distance\_interval\_angle\_psm

Tracking profile with distance, interval, angle and power save mode.

```
{
  "psm_interval": 86400,
  "tracking_angle": 10,
  "tracking_distance": 100,
  "tracking_interval": 60
}
```

- `psm_interval` - int. Duration in seconds for the device to stay in the deep sleep mode. Min=30, max=86400.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=359.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=100000.

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.

## distance\_interval\_angle

Tracking profile with distance, interval and angle.

```
{
  "tracking_interval": 30,
  "tracking_distance": 100,
  "tracking_angle" : 30
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees.

## eelink

Tracking profile for Eelink GOT08, Eelink GOT10, Eelink GPT18, Eelink TK-319, Eelink TK116, Eelink TK119.

```
{
  "tracking_interval": 30
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=18000.

## eelink\_tk116

Tracking profile for Eelink TK116.

```
{
  "tracking_interval": 3600
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=3600.

## eelink\_v2

Tracking profile for Eelink GPT18, Eelink TK-319.

```
{
  "active_tracking_interval": 30,
  "gps_working_mode": "always_on",
  "gsm_working_mode": "auto",
  "tracking_angle": 30,
  "tracking_distance": 50,
  "tracking_interval": 60
}
```

- `active_tracking_interval` - int. Active tracking interval in seconds. Min=30, max=65535.
- `gps_working_mode` - [enum](#). Can be "always\_on" | "auto".
- `gsm_working_mode` - [enum](#). Can be "always\_on" | "auto".
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=0, max=180.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=10000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=65535.

## enfora

Tracking profile for Enfora MT-GL (GSM2218), Enfora MT-Gu (GSM2338), Novatel MT4100, SkyPatrol TT8740, SkyPatrol TT8750.

```
{
  "tracking_distance": 100,
}
```



```
"tracking_interval": 60  
}
```

- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=100, max=10000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=18000.

## esino

Tracking profile for Esino ES-GP34, Esino ES-GT23.

```
{  
  "tracking_interval": 20  
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=20, max=3600.

## etrack\_tlt2h

Tracking profile for E-Track TLT-2H.

```
{  
  "tracking_interval": 600  
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=59999.

## fifotrack

Tracking profile for Fifotrack A100, fifotrack A100 FW1.15+, fifotrack A300, fifotrack A300 FW1.23+, fifotrack A600 (FW before V1.07), fifotrack A600 FW1.07+.

```
{  
  "psm_interval": 3600,  
  "psm_mode": 2,  
  "tracking_angle": 45,  
  "tracking_distance": 100,
```

```

    "tracking_interval": 60
}

```

- `psm_interval` - int. Duration in seconds for the device to stay in the deep sleep mode. Min=0, max=3932100.
- `psm_mode` - int. Define the sleep level when type != `power_save`, 0 - no sleeping, 1 - light sleep (GPS Off, GPRS On, G-sensor On), 2 - deep sleep (GPS Off, GPRS Off, G-sensor On).
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=359.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=65535.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=655350.

## genesis\_g36

Tracking profile for Sinocastel HT-770, Ezlink T28, G36, Orion 7, XiLi Technologies PT100.

```

{
    "tracking_interval": 1
}

```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=1.

## gl200

Queclink/Ruslink GL200/GL300 profile

```

{
    "type": "distance",
    "tracking_interval": 30,
    "tracking_distance": 100,
    "tracking_angle": 30,
    "psm_interval": 600,
    "movement_detection": true,
    "non_movement_duration": 420
}

```

- `type` - [enum](#). Tracking type "distance" or "interval" or "power\_save".

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees.
- `psm_interval` - int. Sending interval when the tracking type is "power\_save", seconds.
- `movement_detection` - boolean.
- `non_movement_duration` - int. In seconds.

## gl500

Queclink/Ruslink GL500 profile.

```
{
  "type": "interval",
  "tracking_interval": 1,
  "wakeup_time": "10:00",
  "psm_interval": 8
}
```

- `type` - [enum](#). Tracking type "interval" or "power save".
- `tracking_interval` - int. Interval in minutes.
- `wakeup_time` - int. Wakeup time for power\_save mode in a format "HH:mm".
- `psm_interval` - int. Update interval in power\_save mode, hours (1, 2, 3, 4, 6, 8, 12, 24).

## gt300

Queclink/Ruslink GT300 profile.

```
{
  "tracking_interval": 5,
  "start_time": "0000",
  "end_time": "2359",
  "movement_detection": true,
  "min_speed": 10,
}
```

```

    "min_distance": 20
}

```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=5, max=86400.
- `start_time` - string. Start time of scheduled fix timing report. The valid format is "HHMM" 0000-2359.
- `end_time` - string. End time of scheduled fix timing report. The valid format is "HHMM" 0000-2359.
- `movement_detection` - boolean. Enable suspend reports if the device at rest.
- `min_speed` - int. The speed threshold of movement detect, km/h 0-999.
- `min_distance` - int. The distance threshold of movement detect, meters 1-9099.

## gotoptk206\_amgps\_freko

Tracking profile for AMGPS Freko.

```

{
    "tracking_interval": 10
}

```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=3600.

## gv500

Queclink/Ruslink GV500 profile.

```

{
    "type": "interval",
    "tracking_interval": 30,
    "tracking_distance": 100,
    "tracking_angle": 30,
    "psm_mode": 1,
    "psm_interval": 600
}

```

- `type` - [enum](#). Tracking type when ignition is ON, "distance" or "interval".
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds.

- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees.
- `psm_mode` - int. Define the sleep level when type != `power_save`, 0 - no sleeping, 1 - light sleep(GPS Off, GPRS On, G-sensor On), 2 - deep sleep(GPS Off, GPRS Off, G-sensor On).
- `psm_interval` - int. Sending interval when the engine is off, seconds.

## gv55lite

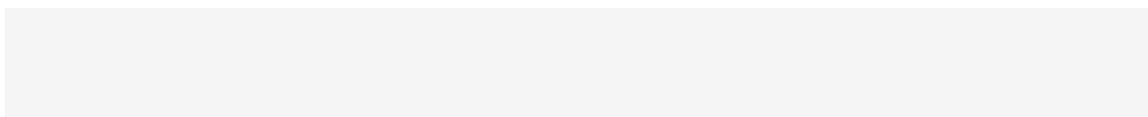
Queclink/Ruslink GV55Lite profile.

```
{
  "type": "interval",
  "tracking_interval": 30,
  "tracking_distance": 100,
  "tracking_angle" : 30,
  "psm_mode": 1,
  "psm_interval": 600
}
```

- `type` - [enum](#). Tracking type when ignition is ON, "distance" or "interval".
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees.
- `psm_mode` - int. Define the sleep level when type != `power_save`, 0 - no sleeping, 1 - light sleep(GPS Off, GPRS On, G-sensor On), 2 - deep sleep(GPS Off, GPRS Off, G-sensor On).
- `psm_interval` - int. Sending interval when the engine is off, seconds.

## gubloost1

Tracking profile for Defenstar GPS668, Gubloos GPS-T1, MiniFinder Pico.



```
{  
  "tracking_interval": 10  
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=9999.

## haicom\_hi603x

Tracking profile for Haicom HI-603X.

```
{  
  "tracking_interval": 30  
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=2592000.

## helioversal\_m1

Tracking profile for Helioversal M1.

```
{  
  "tracking_interval": 30  
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.

## hhd\_g

Tracking profile for HHD G-400, HHD G-600.

```
{  
  "tracking_interval": 20  
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=20.

## howen\_herome

Tracking profile for Hero-ME31-08, Hero-ME32-04, Hero-ME41-04.

```
{  
  "tracking_interval": 30  
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.

## hua\_sheng\_hs3000g

Tracking profile for Hua Sheng HS 3000G.

```
{  
  "psm_interval": 600,  
  "tracking_angle": 10,  
  "tracking_interval": 30  
}
```

- `psm_interval` - int. Sending interval when the engine is off, seconds. Min=60, max=86400.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=5, max=250.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=450.

## huabao

Tracking profile for Huabao HB-T10.

```
{  
  "tracking_interval": 1000  
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=9999.

## intellitrac\_x1

Tracking profile for IntelliTrac X1, IntelliTrac X1+.

```
{
  "tracking_angle": 5,
  "tracking_distance": 100,
  "tracking_interval": 30,
  "type": "interval"
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=5, max=358.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=100, max=65534.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=65534.
- `type` - [enum](#). Can be "interval" | "distance" | "power\_save" | "distance\_interval\_angle" | "interval\_angle" | "intelligent".

## interval

Tracking profile with an interval only.

```
{
  "tracking_interval": 30
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30.

## interval\_angle

Tracking profile with an interval and angle.

```
{
  "tracking_interval": 30,
  "tracking_angle" : 30
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds.



- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees.

## interval\_angle\_powersave

Tracking profile with an interval, angle and powersave mode.

```
{
  "psm_interval": 60,
  "tracking_angle": 55,
  "tracking_interval": 30
}
```

- `psm_interval` - int. Sending interval when the engine is off, seconds. Min=60, max=86400.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=5, max=355.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=900.

## interval\_powersave

Tracking profile with an interval and powersave mode.

```
{
  "psm_interval": 3000,
  "psm_mode": 1,
  "tracking_interval": 30
}
```

- `psm_interval` - int. Sending interval when the engine is off, seconds. Min=60, max=86400.
- `psm_mode` - int. Define the sleep level when type != `power_save`, 0 - no sleeping, 1 - light sleep(GPS Off, GPRS On, G-sensor On).
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=180.

## jimi\_jc100

Tracking profile for Jimi JC100.

```
{
  "tracking_interval": 30
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=18000.

## jinsheng\_js810

Tracking profile for Jin Sheng JS810, Jin Sheng JS810S.

```
{
  "tracking_interval": 30
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=65534.

## jointech\_gp

Tracking profile for Jointech GP4000, Jointech GP6000, Jointech GP6000F.

```
{
  "psm_interval": 3600,
  "psm_mode": 1,
  "tracking_angle": 45,
  "tracking_distance": 150,
  "tracking_interval": 30,
  "type": "interval"
}
```

- `psm_interval` - int. Sending interval when the engine is off, seconds. Min=300, max=65535.
- `psm_mode` - int. Define the sleep level when type != `power_save`, 1 - no sleeping, 2 - light sleep(GPS Off, GPRS On, G-sensor On), 3 - deep sleep(GPS Off, GPRS Off, G-sensor On).
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=90.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=100, max=65535.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=65535.

- `type` - `enum`. Can be "interval" | "distance" | "power\_save" | "distance\_interval\_angle" | "interval\_angle" | "intelligent".

## jointech\_jt701

Tracking profile for Jointech JT701.

```
{
  "tracking_interval": 240
}
```

- `tracking_interval` - `int`. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=60000.

## jointech\_jt703

Profile for Jointech JT703B

```
{
  "tracking_interval": 10,
  "sleep_mode": "enabled",
  "wakeup_timers": ["10:00:00", "16:00:00"],
  "sleep_time_in_minutes": 60
}
```

- `tracking_interval` - `int`. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=60000.
- `sleep_mode` - `enum`. Can be "enabled" | "disabled".
- `wakeup_timers` - optional string. Define wake-up timers when the sleep mode enabled, 1-48 timers. Local time in a standard format `HH:mm:ss`.
- `sleep_time_in_minutes` - optional `int`. Define the time interval which the unit stays in the sleeping state when wake-up timers not defined. Min=10, max=1440.

## jointech\_jt707

Tracking profile for Jointech JT707.

```
{
  "psm_interval": 150,
  "psm_mode": 0,
}
```

```

    "tracking_interval": 60
}

```

- `psm_interval` - int. Sending interval when the engine is off, seconds. Min=10, max=1440.
- `psm_mode` - int. Define the sleep level when type != `power_save`, 0 - no sleeping, 1 - light sleep(GPS Off, GPRS On, G-sensor On).
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=5, max=43200.

## keson\_ks168

Tracking profile for Keson KS168.

```

{
    "tracking_interval": 10
}

```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=65535.

## laipacs911

Tracking profile for Laipac S911 Lola, Laipac-911BL.

```

{
    "tracking_distance": 1000,
    "tracking_interval": 30,
    "type": "distance"
}

```

- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=100000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=5, max=43200.
- `type` - [enum](#). Can be "interval" | "distance" | "power\_save" | "distance\_interval\_angle" | "interval\_angle" | "intelligent".

## lk200

Tracking profile for LKGPS LK209A, LKGPS LK209B, LKGPS LK210.

```
{
  "tracking_interval": 45
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=65535.

## logosoft

Tracking profile for Logosoft Log-101.

```
{
  "type": "interval",
  "tracking_interval": 30,
  "tracking_distance": 300,
  "tracking_angle" : 10
}
```

- `type` - [enum](#). Tracking type "interval" or "distance" or "intelligent".
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=300.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10.

## m7

Profile for Navixy M7.

```
{
  "type": "interval",
  "psm_mode": 1,
  "tracking_interval": 30,
  "tracking_distance": 100,
  "tracking_angle" : 30,
  "psm_interval": 600,
  "wakeup_timer1": "10:00",
  "wakeup_timer2": "16:00",
}
```

```

    "wakeup_timer3": "22:00"
}

```

- `type` - `enum`. Can be "interval" (send tracking data based on time intervals), "distance" (send tracking data after passing specified distance).
- `psm_mode` - `int`. Power save mode, 0 - disable, 1 - powersave without timers, 2 - powersave with timers.
- `tracking_interval` - optional `int`. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds.
- `tracking_distance` - optional `int`. Distance in meters, e.g. 100 means that the device will send data every 100 meters.
- `tracking_angle` - optional `int`. If specified, the device will additionally send data when it changes direction to specified angle, e.g. 30 degrees.
- `psm_interval` - optional `int`. Define the time interval in seconds (600-3932100) which the unit stays in the sleeping state.
- `wakeup_timer` - optional string. Timer 1-3.

## maxtrack\_140

Tracking profile for Maxtrack MXT-140.

```

{
    "tracking_angle": 60,
    "tracking_distance": 500,
    "tracking_interval": 20
}

```

- `tracking_angle` - `int`. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=180.
- `tracking_distance` - `int`. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=0, max=25500.
- `tracking_interval` - `int`. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=20, max=65535.

## megastek\_gvt430

Tracking profile for Megastek GVT-430.

```

{
    "tracking_angle": 25,

```

```

    "tracking_distance": 100,
    "tracking_interval": 30
}

```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=60.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=100, max=1000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.

## megastek\_mt

Tracking profile for Megastek MT-300, Megastek MT-90s, Megastek MT100.

```

{
    "tracking_interval": 30
}

```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.

## megastek\_mt100

Tracking profile for Megastek MT100.

```

{
    "tracking_distance": 50,
    "tracking_interval": 300,
    "type": "intelligent"
}

```

- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=100000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=65535.
- `type` - [enum](#). Can be "interval" | "distance" | "power\_save" | "distance\_interval\_angle" | "interval\_angle" | "intelligent".

## meiligaovt

Tracking profile for GoTop VT360, GoTop VT380, Meiligao VT310, Meitrack VT310, RedView VT310.

```
{
  "tracking_angle": 30,
  "tracking_distance": 200,
  "tracking_interval": 10
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=0, max=359.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=5000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=655350.

## meitrack

Meitrack profile.

```
{
  "tracking_interval": 30,
  "tracking_distance": 100,
  "tracking_angle" : 30,
  "psm_mode": 0,
  "psm_interval": 3600
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees.
- `psm_mode` - optional int. Define the sleep level when type != `power_save`, `0` - no sleeping, `1` - light sleep(GPS Off, GPRS On, G-sensor On), `2` - deep sleep(GPS Off, GPRS Off, G-sensor On).
- `psm_interval` - optional int. Define the time interval in seconds which the unit stays in the sleeping state.



## meitrack\_asset

Tracking profile for Meitrack T355v2.

```
{
  "psm_interval": 3932100,
  "psm_mode": 0,
  "tracking_angle": 10,
  "tracking_distance": 50,
  "tracking_interval": 30
}
```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=0, max=3932100.
- `psm_mode` - optional int. Define the sleep level when type != `power_save`, `0` - no sleeping, `1` - light sleep(GPS Off, GPRS On, G-sensor On), `2` - deep sleep(GPS Off, GPRS Off, G-sensor On).
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=359.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=65535.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=655350.

## meitrack\_vehicle

Tracking profile for Meitrack MVT100, Meitrack MVT340, Meitrack MVT380, Meitrack MVT600, Meitrack T1, Meitrack T3, Meitrack T333, Meitrack T366G, Meitrack T366L, Meitrack T622G, Meitrack TC68S, Meitrack TC68SG.

```
{
  "on_stop_tracking_interval": 120,
  "psm_interval": 300,
  "psm_mode": 2,
  "tracking_angle": 45,
  "tracking_distance": 50,
  "tracking_interval": 30
}
```

- `on_stop_tracking_interval` - int. Tracking interval in seconds when the vehicle stopped. Min=0, max=655350.
- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=0, max=3932100.

- `psm_mode` - optional int. Define the sleep level when type != `power_save`, `0` - no sleeping, `1` - light sleep(GPS Off, GPRS On, G-sensor On), `2` - deep sleep(GPS Off, GPRS Off, G-sensor On).
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=359.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=65535.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=655350.

## meitrack\_without\_ps

Tracking profile for Meitrack P66.

```
{
  "tracking_angle": 45,
  "tracking_distance": 150,
  "tracking_interval": 60
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=359.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=65535.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=655350.

## mictrack\_mp90

Tracking profile for MicTrack MP-90.

```
{
  "tracking_angle": 20,
  "tracking_interval": 60
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=20, max=180.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=65535.

## mika\_g1

Tracking profile for MIKA G1.

```
{  
  "tracking_interval": 30  
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=10000.

## mrd\_100

Tracking profile for MRD-100.

```
{  
  "tracking_interval": 20  
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=20, max=65535.

## mwp008\_a

Tracking profile for Diwei TK116, Moralwinhk P008A, Moralwinhk P168.

```
{  
  "tracking_interval": 10  
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=65535.

## myrope\_m500

Tracking profile for MyRope M528, MyRope M588.

```
{  
  "psm_interval": 60,  
  "tracking_distance": 10,  
  "tracking_interval": 50  
}
```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=60, max=65535.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=1, max=65535.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=1, max=65535.

## navisetgt

Tracking profile for Naviset GT-10, Naviset GT-20.

```
{
  "tracking_angle": 120,
  "tracking_distance": 150,
  "tracking_interval": 240
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=5, max=180.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=255.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=300.

## noran

Tracking profile for Noran NR008, Noran NR024, Noran NR100.

```
{
  "tracking_interval": 150
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=15, max=64800.

## oigo\_ar2

Tracking profile for Oigo AR-2GM, Oigo AR-3HU.

```
{
  "psm_interval": 60,
  "tracking_angle": 45,
  "tracking_distance": 300,
  "tracking_interval": 15
}
```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=15, max=604800.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=0, max=180.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=0, max=60000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=15, max=604800.

## orange\_tk103

Tracking profile for Orange TK-103.

```
{
  "tracking_interval": 990
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=990.

## piccolo\_atx

Tracking profile for Piccolo ATX.

```
{
  "tracking_interval": 300
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=300, max=86400.

## piccolo\_distance\_interval\_angle

Tracking profile for Piccolo ATX2S, Piccolo Hybrid+, Piccolo STX, Piccolo TMX+.

```
{
  "tracking_angle": 30,
  "tracking_distance": 100,
  "tracking_interval": 30
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=30, max=150.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=100, max=10000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=65535.

## piccolo\_plus

### Profile Wireless Links for Piccolo Plus

```
{
  "sleep_mode": "disabled",
  "track_by": "interval",
  "tracking_interval": 60,
  "tracking_distance": 100,
  "track_by_angle": true,
  "tracking_angle": 30,
  "asset_moving_interval": 300,
  "asset_stopped_interval": 86400
}
```

- `sleep_mode` - **enum**. Can be "disabled" | "engine" | "asset" | "hybrid".
- `track_by` - optional **enum**. Can be "interval" | "distance". Need for disabled, engine, hybrid modes.
- `tracking_interval` - optional int. Interval in seconds, need for disabled, engine, hybrid modes. Min=60, max=86400.
- `tracking_distance` - optional int. Distance in meters, need for disabled, engine, hybrid modes. Min=100, max=10000.
- `track_by_angle` - optional boolean. Need for disabled, engine, hybrid modes.
- `tracking_angle` - optional int. If specified, the device will additionally send data when it changes direction to specified angle, e.g. 30 degrees, need for disabled, engine, hybrid modes. Min=30, max=150.
- `asset_moving_interval` - optional int. Need for asset and hybrid modes. Min=300, max=86400.

- `asset_stopped_interval` - optional int. Need for asset and hybrid modes. Min=300, max=86400.

## redview\_vt680

Tracking profile for RedView VT680.

```
{  
  "tracking_angle": 60,  
  "tracking_interval": 10  
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=30, max=270.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=655350.

## sanfone

Tracking profile for Sanfone SF100, Sanfone SF700.

```
{  
  "tracking_angle": 120,  
  "tracking_distance": 60,  
  "tracking_interval": 60  
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=360.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=30, max=60000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=999.

## satsol

Tracking profile for SAT-LITE 3, SAT-LITE 4, Sat Lite 2, Sat Pro, Super Lite.

```
{  
  "psm_interval": 30,  
  "tracking_angle": 10,  
  "tracking_distance": 50,  
}
```

```
"tracking_interval": 30
}
```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=30, max=86400.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=359.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=9999.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.

## senseitp211

```
{
  "tracking_interval": 30,
  "gps_enabled": true
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30.
- `gps_enabled` - boolean.

## sheriff\_avax12

Tracking profile for Sheriff AWAX12.

```
{
  "tracking_interval": 900
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=900, max=86400.

## sinowell\_g102

Tracking profile for Sinowell G102.

```
{
  "psm_interval": 10,
  "tracking_angle": 5,
```



```

    "tracking_distance": 50,
    "tracking_interval": 10
}

```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=10, max=65000.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=5, max=180.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=1000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=1000.

## skypatrol\_tt8750plus

Tracking profile for SkyPatrol TT8750+.

```

{
    "psm_interval": 30,
    "tracking_angle": 10,
    "tracking_distance": 100,
    "tracking_interval": 30
}

```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=30, max=18000.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=359.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=100, max=10000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=18000.

## sleep\_active

Tracking profile for СКАТ-Маяк.

```

{
    "active_time": 300,
}

```

```

    "sleep_time": 300
}

```

- `active_time` - int. Min=300, max=599940 seconds.
- `sleep_time` - int. Min=300, max=599940 seconds.

## spetrotec\_iwatcher

Tracking profile for Spetrotec i-WATCHER AVL.

```

{
    "tracking_distance": 100,
    "tracking_interval": 60,
    "type": "interval"
}

```

- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=100, max=100000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=60, max=86400.
- `type` - [enum](#). Can be "interval" | "distance" | "power\_save" | "distance\_interval\_angle" | "interval\_angle" | "intelligent".

## stab\_liner

Tracking profile for M2M-Cyber GLX, STAB Liner 102.

```

{
    "psm_interval": 3600,
    "tracking_angle": 10,
    "tracking_distance": 50,
    "tracking_interval": 30
}

```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=0, max=3600.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=0, max=180.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=100000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=0, max=3600.

## starcom\_helios

Tracking profile for Starcom Helios Advanced, Starcom Helios Hybrid, Starcom Helios TT.

```
{
  "tracking_interval": 10
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=0, max=432000.

## starline\_m17

Tracking profile for Starline M17.

```
{
  "psm_interval": 600,
  "psm_mode": 0,
  "tracking_interval": 100
}
```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=60, max=3540.
- `psm_mode` - int. Define the sleep level when type != `power_save`, 0 - no sleeping, 1 - light sleep(GPS Off, GPRS On, G-sensor On).
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=0, max=300.

## suntech\_distance\_interval\_angle

Tracking profile for Suntech ST200, Suntech ST215, Suntech ST300, Suntech ST310U, Suntech ST340LC, Suntech ST600R, Suntech ST600V, Suntech ST650.

```
{
  "tracking_angle": 30,
  "tracking_distance": 50,
  "tracking_interval": 20
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=0, max=180.

- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=60000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=20, max=60000.

## suntech\_interval

Tracking profile for Suntech ST940.

```
{
  "tracking_interval": 20
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=20, max=60000.

## syrus

Tracking profile for Syrus 2G.

```
{
  "tracking_angle": 5,
  "tracking_distance": 200,
  "tracking_interval": 30
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=5, max=90.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=100, max=5000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=9999.

## telfm

Teltonika FM tracking profile.

```
{
  "tracking_angle": 30,
  "tracking_distance": 100,
  "tracking_interval": 60,
  "on_stop_tracking_interval": 180,
}
```

```

    "sleep_mode": "disabled",
    "stop_detection": "ignition"
}

```

\* `tracking_angle` – int. Degrees 10-255, the device will send tracking data when course changing more than defined value. \* `tracking_distance` – int. Distance in meters 50-65535, e.g. 100 means that the device will send data every 100 meters. \* `tracking_interval` – int. Interval in seconds 30-255, e.g. 30 means that the device will send tracking data no more frequently than every 30 seconds. \* `on_stop_tracking_interval` – int. On stop interval in seconds 30-65535, e.g. 30 means that the device will send tracking data not less frequently than every 30 seconds. \* `sleep_mode` – [enum](#). Can be "disabled" | "soft\_sleep". \* `stop_detection` – [enum](#). Can be "ignition" | "g\_sensor" | "gps".

## telfm5x

Tracking profile for Teltonika FM5500, Teltonika FM6320, Teltonika FMB630, Teltonika FMB640.

```

{
    "sleep_mode": "disabled",
    "sleep_timeout": 300,
    "tracking_angle": 25,
    "tracking_distance": 50,
    "tracking_interval": 30
}

```

- `sleep_mode` – [enum](#). Can be "disabled" | "soft\_sleep".
- `sleep_timeout` - int. Can be min=300, max=2592000 seconds.
- `tracking_angle` – int. Degrees min=0, max=180, the device will send tracking data when course changing more than defined value.
- `tracking_distance` – int. Distance in meters min=50, max=65535, e.g. 100 means that the device will send data every 100 meters.
- `tracking_interval` – int. Interval in seconds min=30, max=2592000, e.g. 30 means that the device will send tracking data no more frequently than every 30 seconds.

## topfly

Tracking profile for TopFlyTech T8603, TopFlyTech T8608, TopFlyTech T8803, TopFlyTech T8803 Pro, TopFlyTech T8803+, TopFlyTech T8806, TopFlyTech T8806+,

TopFlyTech T8806+R, TopFlyTech T8808A, TopFlyTech T8808A+, TopFlyTech T8808B, TopFlyTech T8808B+.

```
{
  "psm_interval": 10000,
  "tracking_angle": 60,
  "tracking_distance": 50,
  "tracking_interval": 30
}
```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=0, max=65535.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=0, max=90.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=65535.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=0, max=65535.

## topshine\_distance\_interval

Tracking profile for TopShine MT02, TopShine MT08, TopShine OGT100, TopShine VT1000, TopShine VT200W, TopShine VT900.

```
{
  "tracking_distance": 50,
  "tracking_interval": 10
}
```

- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=65535.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=655350.

## topshine\_distance\_interval\_angle

Tracking profile for TopShine MT08, TopShine OGT100, TopShine VT1000.

```
{
  "tracking_angle": 15,
  "tracking_distance": 50,
}
```

```
"tracking_interval": 60
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=0, max=359.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=65535.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=655350.

## topten

Tracking profile for TopTen GT08, TopTen TK-510, TopTen TK228.

```
{
  "tracking_angle": 25,
  "tracking_interval": 30
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=0, max=359.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=655350.

## totarget

Tracking profile for TT-08, VG-eLock7A.

```
{
  "tracking_interval": 30
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=300.

## totem

Tracking profile for TotemTech AT05, TotemTech AT07.

```
{
  "psm_interval": 15000,
}
```

```

    "tracking_angle": 10,
    "tracking_distance": 60,
    "tracking_interval": 30
}

```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=10, max=18000.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=180.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=10, max=18000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=10, max=18000.

## trackertech\_msp320

Tracking profile for Tracker Technology MSP320.

```

{
    "tracking_interval": 30
}

```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.

## trackertech\_msp340

Tracking profile for Tracker Technology MSP340.

```

{
    "psm_interval": 180,
    "tracking_interval": 30
}

```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=180, max=86400.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.



## trackertech\_msp350

Tracking profile for Tracker Technology MSP350.

```
{
  "psm_interval": 2147483647,
  "psm_mode": 0,
  "tracking_distance": 50,
  "tracking_interval": 30
}
```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=60, max=2147483647.
- `psm_mode` - int. Define the sleep level when type != `power_save`, `0` - no sleeping, `1` - light sleep(GPS Off, GPRS On, G-sensor On), `2` - deep sleep(GPS Off, GPRS Off, G-sensor On), `3` - ultra deep sleep.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=50, max=100000.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=86400.

## tramigo

Profile for Tramigo models that do not support the interval in seconds

```
{
  "tracking_interval": 1,
  "tracking_distance": 0.5,
  "tracking_angle": 20,
  "on_stop_tracking_interval": 120,
  "sleep_mode": "disabled"
}
```

- `tracking_interval` - int. Interval in minutes, e.g. 30 means that the device will send tracking data every 30 minutes. Min=1, max=10080.
- `tracking_distance` - float. Distance in kilometers, e.g. 0.5 means that the device will send data every 500 meters. Min=0.5, max=20.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=20, max=180.
- `on_stop_tracking_interval` - int. Interval in minutes when not in a trip. Min=1, max=10080.
- `sleep_mode` - sting enum. Can be "disabled" | "enabled".

## tramigo\_with\_seconds

Profile for Tramigo models that do support the interval in seconds

```
{
  "tracking_interval": 30,
  "tracking_distance": 20,
  "tracking_angle": 180,
  "on_stop_tracking_interval": 100,
  "sleep_mode": "enabled"
}
```

- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=604800.
- `tracking_distance` - float. Distance in kilometers, e.g. 0.5 means that the device will send data every 500 meters. Min=0.5, max=20.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=20, max=180.
- `on_stop_tracking_interval` - int. Interval in minutes when not in a trip. Min=1, max=10080.
- `sleep_mode` - sting enum. Can be "disabled" | "enabled".

## tt1

Profile for Navixy TT-1.

```
{
  "type": "interval",
  "psm_mode": 2,
  "tracking_interval": 30,
  "tracking_distance": 100,
  "tracking_angle": 30,
  "psm_interval": 60,
  "bat_voltage": "1.5",
  "bat_psm_interval": 600
}
```

- `type` - [enum](#). Can be "interval" (send tracking data based on time intervals), "distance" (send tracking data after passing specified distance).
- `psm_mode` - int. power save mode, 0 - disable, 1 - powersave mode, 2 - Back-up Battery Power Saving Mode
- `tracking_interval` - optional int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds.

- `tracking_distance` - optional int. Distance in meters, e.g. 100 means that the device will send data every 100 meters.
- `tracking_angle` - optional int. If specified, the device will additionally send data when it changes direction to specified angle, e.g. 30 degrees.
- `psm_interval` - optional int. Define the time interval in seconds (600-3932100) which the unit stays in the sleeping state.
- `bat_voltage` - optional string. Threshold of low back-up battery voltage.
- `bat_psm_interval` - optional int. Sleeping duration when battery voltage below defined threshold, seconds.

## ulbotech\_t300

Tracking profile for IMTSA TR2-OBd, Ulbotech T361, Ulbotech T381.

```
{
  "tracking_angle": 3,
  "tracking_distance": 150,
  "tracking_interval": 30
}
```

- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=3, max=20.
- `tracking_distance` - int. Distance in meters, e.g. 100 means that the device will send data every 100 meters. Min=0, max=25500.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=900.

## vjoy

Tracking profile for Kingneed C30, Kingneed T0024 / T4024, Kingneed T0026, Kingneed T1124, Kingneed T12, Kingneed T16/T18/T19, Kingneed T300, Kingneed T3124 / T5124, Kingneed T500, Kingneed T6024, Kingneed T6124, Kingneed T630, Kingneed T8124, Kingneed TK10, Kingneed TK101, Kingneed TK20, Kingneed TK5, VJOYCAR T0026G, VJOYCAR T13G, VJOYCAR T13GSE, VJOYCAR T633G, VJOYCAR TK10SDC, VJoy T12, VJoy TK05, VJoy TK10GSE, VJoy TK10GSE Solar, VJoy TK20SE.

```
{
  "continuous_report_interval": 10,
  "motion_interval": 30,
  "psm_mode": 1,
}
```

```

    "psm_wake_up_interval": 1
}

```

- `continuous_report_interval` - int. Min=10, max=5940 seconds.
- `motion_interval` - int. Min=30, max=999 seconds.
- `psm_mode` - int. Define the sleep level when type != `power_save`, 0 - no sleeping, 1 - light sleep(GPS Off, GPRS On, G-sensor On).
- `psm_wake_up_interval` - int. Min=1, max=99 hours.

## xirgo

Tracking profile for Xirgo XT-2050C, Xirgo XT-2060G, Xirgo XT-2150C, Xirgo XT-2160G, Xirgo XT-2450V, Xirgo XT-2460G, Xirgo XT-4750C, Xirgo XT-4760G, Xirgo XT-4850C.

```

{
    "psm_interval": 2592000,
    "tracking_angle": 10,
    "tracking_distance": 1,
    "tracking_interval": 30
}

```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=60, max=2592000.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=180.
- `tracking_distance` - int. Distance in miles, e.g. 100 means that the device will send data every 100 miles. Min=1, max=100.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=2592000.

## xirgo\_48

Tracking profile for Xirgo XT-4850C.

```

{
    "psm_interval": 60,
    "tracking_angle": 10,
    "tracking_distance": 1,
}

```

```

    "tracking_interval": 30
}

```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=60, max=2592000.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=180.
- `tracking_distance` - int. Distance in miles, e.g. 100 means that the device will send data every 100 miles. Min=1, max=100.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=2592000.

## xirgo\_dist

Tracking profile for Xirgo XT-2050C, Xirgo XT-2060G, Xirgo XT-2450V, Xirgo XT-2460G, Xirgo XT-4750C, Xirgo XT-4760G.

```

{
    "psm_interval": 60,
    "tracking_angle": 10,
    "tracking_distance": 2,
    "tracking_interval": 60
}

```

- `psm_interval` - int. Define the time interval in seconds which the unit stays in the sleeping state. Min=60, max=2592000.
- `tracking_angle` - int. The device will additionally send data when it changes direction to specified angle, e.g. 30 degrees. Min=10, max=180.
- `tracking_distance` - int. Distance in miles, e.g. 100 means that the device will send data every 100 miles. Min=1, max=100.
- `tracking_interval` - int. Interval in seconds, e.g. 30 means that the device will send tracking data every 30 seconds. Min=30, max=2592000.

## yatut\_poisk

"Я ТУТ ПОИСК" tracking profile. name: 'yatut\_poisk'

```

{
    "mode": "DAILY",
    "main_wakeup_time": "12:00",
    "wakeup_period": "24",
}

```

```
    "gps_determination_period": 0  
}
```

- `mode` - [enum](#). Device's working mode. Can be "DAILY" | "TEST" | "SEARCH", default="DAILY".
- `main_wakeup_time` - string. At what time to wake up if mode == "DAILY". Format HH:mm, default="12:00"
- `wakeup_period` - [enum](#). Only values 8, 12 or 24 (hours). Default="24"
- `gps_determination_period` - int. How often to determine the position by satellites (in days). Zero (0) means on each waking up. Min=0, max=30, default=0.

Last update: October 31, 2021







# Parking detection

Contains API calls for getting and changing parking detection for the tracker. It is responsible for the parking detection function in the UI. The monitoring system automatically detects the facts of parking (states without movement), for the following purposes:

- To split a movement trajectory to separate trips - for clear illustration and easy viewing in tabular reports;
- To capture "Trip end" / "Trip start" events - with possibility of Email/SMS notification.

## API actions

API base path: `/tracker/settings/trip_detection`.

### read

Gets parking detection settings for the specified tracker.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/settings/trip_detection/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id": 123456}'
```

##### HTTP GET

```
https://api.navixy.com/v2/tracker/settings/trip_detection/read?hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456
```

## response

```
{
  "success": true,
  "min_idle_duration_minutes": 5,
  "idle_speed_threshold": 3,
  "ignition_aware": false,
  "motion_sensor_aware": false
}
```

- `min_idle_duration_minutes` - int. Number of minutes the device must be idle before a trip considered finished.
- `idle_speed_threshold` - int. Speed (km/h) below which the device marked as being idle.
- `ignition_aware` - boolean. Check ignition state to detect a trip.
- `motion_sensor_aware` - boolean. Check motion sensor state to detect a trip.

## errors

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.

## update

Updates parking detection settings for the specified tracker.

**required sub-user rights:** `tracker_update`.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
min_idle_duration_minutes	Number of minutes the device must be idle before a trip considered finished. Min=1, max=1440.	int	5

name	description	type	format
idle_speed_threshold	Speed (km/h) below which the device marked as being idle. Min=0, max=200. If 0 - will never idle.	int	3
ignition_aware	Check ignition state to detect a trip.	boolean	false
motion_sensor_aware	Check motion sensor state to detect a trip.	boolean	false

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/settings/
trip_detection/update' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
123456, "min_idle_duration_minutes": "5", "idle_speed_threshold":
"3", "ignition_aware": false, "motion_sensor_aware": false}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/settings/trip_detection/update?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456&min_idle_durat
```

## response

```
{"success": true}
```

## errors

- 204 – Entity not found (if there is no tracker with such ID belonging to authorized user).
- 208 – Device blocked (if tracker exists but was blocked due to tariff restrictions or some other reason).

Last update: December 26, 2022





# About special settings

## About special settings

Some trackers provide additional specific kind of control which is defined with `special_control` field of tracker model. This field contains `type`, which identifies a certain kind of settings. (For example "pwr\_off\_key" or "sos\_key", which you can see below) `special_control` = "none" means that tracker doesn't have specific kind of control. In other cases you can:

- **read** special settings with [api/tracker/settings/special/read](#),
- **update** special settings with [api/tracker/settings/special/update](#),
- **perform special control** with [api/tracker/send\\_command](#).

Such control assumes tracker special settings

## API actions

API base path: `/tracker/settings/special`.

### read

Gets special settings for the specified tracker.

#### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
type	Optional. Type of special object.	enum	"electronic_lock_password"

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/settings/special/
read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
123456}'
```

### HTTP GET

```
https://api.navixy.com/v2/tracker/settings/special/read?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456
```

## responses

If parameter type is present:

```
{
  "success": true,
  "value": {
    "type": "electronic_lock_password",
    "password": "4567879",
    "remember_password": false
  }
}
```

- `value` - settings object.

If parameter type omitted:

```
{
  "success": true,
  "list": [{
    "type": "electronic_lock_password",
    "password": "4567879",
    "remember_password": false
  }, {
    "type": "hhd_lock_password",
    "password": "25698545",
    "remember_password": true
  }]
}
```

- `list` - array of objects. Settings object array.

Settings object structures by type:

### electronic\_lock\_password

```
{
  "type": "electronic_lock_password",
  "password": "password",
```

```
    "remember_password": false
}
```

- password - string. Nullable.

### hhd\_lock\_password

```
{
  "type": "hhd_lock_password",
  "password": "56894567",
  "remember_password": true
}
```

- password - string. Nullable. 8 digits.

### jointech\_lock\_password

```
{
  "type": "jointech_lock_password",
  "password": "d45s6w",
  "remember_password": false
}
```

- password - string. Nullable. 6 non-space, non-comma symbols.

### vg\_lock\_password

```
{
  "type": "vg_lock_password",
  "password": "123456",
  "remember_password": true
}
```

- password - string. Nullable. 6 digits.

### autofon\_sms\_alerts

```
{
  "type": "autofon_sms_alerts",
  "low_battery_mode": "enable",
  "ext_input_mode": "disable",
  "sos_button_mode": "enable"
}
```

- low\_battery\_mode - [enum](#). Can be "enable" | "disable".
- ext\_input\_mode - [enum](#). Can be "enable" | "disable".
- sos\_button\_mode - [enum](#). Can be "enable" | "disable".

### auto\_geofence\_telfm

---



```
{
  "type": "auto_geofence_telfm",
  "mode": "enable",
  "activation_timeout": 300,
  "radius": 50
}
```

- `mode` - **enum**. Can be "enable" | "disable".
- `activation_timeout` - int. 0-65535 seconds.
- `radius` - int. 50 - 10000 meters.

### **bce\_tacho\_control**

```
{
  "type": "bce_tacho_control",
  "function": "slot1"
}
```

- `function` - **enum**. Can be "slot1" | "slot2" | "vu\_activities" | "vu\_no\_activities"

### **call\_button**

```
{
  "type": "call_button",
  "capacity": 1,
  "items": [{ "phone": "45641784111"}]
}
```

- `items` - Array of phone numbers (10-15 digits) represented as strings.
  - `phone` - string. Phone number in the international format without "+" sign.

### **call\_buttons\_v40**

```
{
  "type": "call_buttons_v40",
  "capacity": 4,
  "items": [{ "phone": "45641784111"}]
}
```

- `items` - Array of phone numbers (10-15 digits) represented as strings.
  - `phone` - string. Phone number in the international format without "+" sign.

### **careu\_psm**

```
{
  "type": "careu_psm",
  "sleep_when_ignition_off": true,
}
```

```

    "sleep_when_no_motion": true,
    "sleep_when_no_communication": true,
    "sleep_conditions_duration": 1,
    "deep_sleep_conditions_duration": 300,
    "wake_up_interval": 30,
    "wake_up_from_dsm_interval": 2
}

```

- `sleep_when_ignition_off` - boolean.
- `sleep_when_no_motion` - boolean.
- `sleep_when_no_communication` - boolean.
- `sleep_conditions_duration` – int. Delay between the moment when conditions met and sleep mode activation in minutes. Can be 1-255.
- `deep_sleep_conditions_duration` – int. Delay between sleep mode activation and deep sleep mode activation in minutes. Can be 0-65535.
- `wake_up_interval` – int. Delay before waking up from sleep mode in minutes. Can be 0-65535.
- `wake_up_from_dsm_interval` – int. Delay before waking up from deep sleep mode in hours. Can be 0-255.
- 0 in these fields means don't switch.

## castel\_alarms

```

{
  "type": "castel_alarms",
  "acceleration": {
    "report": true,
    "beep": true,
    "threshold": 0.4
  },
  "deceleration": {
    "report": false,
    "beep": false,
    "threshold": 0.7
  },
  "crash": {
    "report": true,
    "beep": true,
    "threshold": 1.0
  },
  "sharp_turn": {
    "report": true,
    "beep": true,
    "threshold": 0.3
  }
}

```

- `report` - boolean. If `true` will send notification to server upon an event.

- `beep` - boolean. If `true` will sound upon an event.
- `threshold` - double. Normal values range where event does not occur. Each unit equals 1 g.
  - `acceleration` - 0.2 - 0.8.
  - `deceleration` - 0.3 - 1.0.
  - `crash` - 1.0 - 2.0.
  - `sharp_turn` - 0.3 - 0.9.

### castel\_obd

```
{
  "type": "castel_obd",
  "enable_pid_reports": true,
  "pid_data_records_per_message": 1,
  "pid_data_collect_interval": 30
}
```

- `enable_pid_reports` - boolean.
- `pid_data_records_per_message` - int. Count of records per one message. Can be 1 - 20.
- `pid_data_collect_interval` - int. Data collect interval in seconds. Can be 30 - 600.

### charging\_gmt100

```
{
  "type": "charging_gmt100",
  "mode": "on_need"
}
```

- `mode` - [enum](#). Can be "on\_need" | "ign\_on\_only" | "ign\_on" | "low\_charge".

### ddd\_emails

```
{
  "type": "ddd_emails",
  "emails": ["test@email.com", "example@email.com"]
}
```

- `emails` - string array. Valid emails. Maximum size 5.

### digital\_password

```
{
  "type": "digital_password",
```

```
    "password": "123456"
}
```

- `password` - string. 6 digits.

### **fcc\_telfm**

```
{
  "type": "fcc_telfm",
  "fuel_type": "gasoline",
  "engine_volume": 10.0,
  "multiplier": 0.0
}
```

- `fuel_type` - **enum**. Can be "gasoline" | "diesel" | "lpg".
- `engine_volume` - double. Can be 0.0 - 10.0.
- `multiplier` - double. Can be 0.0 - 10.0.

### **galileo\_tacho\_control**

```
{
  "type": "galileo_tacho_control",
  "function": "download"
}
```

### **galileo\_hds**

```
{
  "type": "galileo_hds",
  "mode": "enable",
  "max_acceleration_force": 1.26,
  "max_braking_force": 1.59,
  "max_cornering_force": 0.75
}
```

- `mode` - **enum**. Can be "enable" | "disable".
- `max_acceleration_force` - double. It is a max allowed acceleration force which can be reached while accelerating without triggering harsh acceleration event. Can be 0 - 2.55.
- `max_braking_force` - double. It is a max allowed braking force which can be reached while braking without triggering harsh braking event. Can be 0 - 2.55.
- `max_cornering_force` - double. It is a max allowed cornering angle which can be reached while cornering without triggering harsh cornering event. Can be 0 - 2.55.

### **harsh\_behavior\_hua\_sheng**

```
{
  "type": "harsh_behavior_hua_sheng",
  "mode": "enable",
  "max_acceleration_force": 1.0,
  "max_braking_force": 0.5,
  "max_cornering_force": 0.1
}
```

- `mode` - `enum`. Can be "enable" | "disable".
- `max_acceleration_force` - double. It is a max allowed acceleration force which can be reached while accelerating without triggering harsh acceleration event. Can be 0.1 - 1.0.
- `max_braking_force` - double. It is a max allowed braking force which can be reached while braking without triggering harsh braking event. Can be 0.1 - 1.0.
- `max_cornering_force` - double. It is a max allowed cornering angle which can be reached while cornering without triggering harsh cornering event. Can be 0.1 - 1.0.

### hbm\_telfm

```
{
  "type": "hbm_telfm",
  "mode": "enable",
  "max_acceleration_force": 0.3,
  "max_braking_force": 0.85,
  "max_angular_velocity": 0.1
}
```

- `mode` - `enum`. Can be "enable" | "disable".
- `max_acceleration_force` - double. It is a max allowed acceleration force which can be reached while accelerating without triggering harsh acceleration event. Can be 0.25 - 0.85 g.
- `max_braking_force` - double. It is a max allowed braking force which can be reached while braking without triggering harsh braking event. Can be 0.25 - 0.85 g.
- `max_cornering_force` - double. It is a max allowed cornering angle which can be reached while cornering without triggering harsh cornering event. Can be 0.1 - 1.0 rad/s.

### hbm\_telfm5x

```
{
  "type": "hbm_telfm5x",
  "mode": "enable",
  "max_acceleration_force": 0.5,
  "max_braking_force": 3.0,
}
```

```

    "max_angular_velocity": 10.0
}

```

- `max_acceleration_force` – double. It is a max allowed acceleration force which can be reached while accelerating without triggering harsh acceleration event. Can be 0.5 - 10.0 g.
- `max_braking_force` – double. It is a max allowed braking force which can be reached while braking without triggering harsh braking event. Can be 0.5 - 10.0 g.
- `max_angular_velocity` – double. It is a max allowed cornering angle which can be reached while cornering without triggering harsh cornering event. Can be 0.5 - 10.0 rad/s.

## hbm\_q1

```

{
  "type": "hbm_q1",
  "mode": "enable",
  "high_speed": 100,
  "high_speed_braking_delta": 50,
  "high_speed_acceleration_delta": 50,
  "medium_speed": 70,
  "medium_speed_braking_delta": 50,
  "medium_speed_acceleration_delta": 50,
  "low_speed_braking_delta": 50,
  "low_speed_acceleration_delta": 50
}

```

- `mode` - `enum`. Can be "enable" | "disable".
- `high_speed` - int. Can be 100 - 400.
- `high_speed_braking_delta` - int. Can be 0 - 100.
- `high_speed_acceleration_delta` - int. Can be 0 - 100.
- `medium_speed` - int. Can be 60 - 100.
- `medium_speed_braking_delta` - int. Can be 0 - 100.
- `medium_speed_acceleration_delta` - int. Can be 0 - 100.
- `low_speed_braking_delta` - int. Can be 0 - 100.
- `low_speed_acceleration_delta` - int. Can be 0 - 100.

## hbm\_ms\_q1

```

{
  "type": "hbm_ms_q1",
  "mode": "gps_only",
  "high_speed": 100,
  "high_speed_braking_delta": 50,
  "high_speed_acceleration_delta": 50,
}

```

```

    "medium_speed": 60,
    "medium_speed_braking_delta": 50,
    "medium_speed_acceleration_delta": 50,
    "low_speed_braking_delta": 50,
    "low_speed_acceleration_delta": 50,
    "turn_brake_threshold": 30,
    "turn_brake_duration": 320,
    "acceleration_threshold": 15,
    "acceleration_duration": 1200
}

```

- `mode` - `enum`. Can be "disable" | "gps\_only" | "motion\_sensor\_only" | "gps\_and\_motion\_sensor".
- `high_speed` - `int`. Can be 100 - 400.
- `high_speed_braking_delta` - `int`. Can be 0 - 100.
- `high_speed_acceleration_delta` - `int`. Can be 0 - 100.
- `medium_speed` - `int`. Can be 60 - 100.
- `medium_speed_braking_delta` - `int`. Can be 0 - 100.
- `medium_speed_acceleration_delta` - `int`. Can be 0 - 100.
- `low_speed_braking_delta` - `int`. Can be 0 - 100.
- `low_speed_acceleration_delta` - `int`. Can be 0 - 100.
- `turn_brake_threshold` - `int`. Can be 30 - 70.
- `turn_brake_duration` - `int`. Can be 320 - 800 milliseconds.
- `acceleration_threshold` - `int`. Can be 15 - 50.
- `acceleration_duration` - `int`. Can be 400 - 2000 milliseconds.

### harsh\_behavior\_bce

```

{
    "type": "harsh_behavior_bce",
    "is_switched_off": false,
    "acceleration_limit": 0.04,
    "braking_limit": 1.21,
    "cornering_limit": 2.38
}

```

- `is_switched_off` - `boolean`.
- `acceleration_limit` - `double`. Can be 0.04 - 3.
- `braking_limit` - `double`. Can be 0.04 - 3.
- `cornering_limit` - `double`. Can be 0.04 - 3.

### harsh\_behavior\_concox\_x1

```
{
  "type": "harsh_behavior_concox_x1",
  "acc_speed": 40,
  "acc_detection_time": 4,
  "braking_speed": 60,
  "braking_detection_time": 2
}
```

- `acc_speed` - int. Can be 0 - 100.
- `acc_detection_time` - int. Can be 0 - 10.
- `braking_speed` - int. Can be 0 - 100.
- `braking_detection_time` - int. Can be 0 - 10.

### harsh\_behavior\_tramigo

```
{
  "type": "harsh_behavior_tramigo",
  "mode": "enable",
  "max_acceleration_force": 0.5,
  "max_braking_force": 1.3
}
```

- `mode` - [enum](#). Can be "enable" | "disable".
- `max_acceleration_force` - double. Can be 0.1 - 8.
- `max_braking_force` - double. Can be 0.1 - 8.

### harsh\_behavior\_ruptela

```
{
  "type": "harsh_behavior_ruptela",
  "braking_limit": 30,
  "acceleration_limit": 60
}
```

- `braking_limit` - int. Can be 0 - 100.
- `acceleration_limit` - int. Can be 0 - 100.

### nimbelink\_accel

```
{
  "type": "nimbelink_accel",
  "mode": "enable",
  "x": 1.12,
  "y": 0.8,
}
```



```
    "z": 2.33
}
```

- `mode` - [enum](#). Can be "enable" | "disable".
- `x` - double. Can be 0 - 2.55.
- `y` - double. Can be 0 - 2.55.
- `z` - double. Can be 0 - 2.55.

### **hua\_sheng\_vibration\_sensitivity**

```
{
  "type": "hua_sheng_vibration_sensitivity",
  "sensitivity": "easy"
}
```

- `sensitivity` - [enum](#). Can be "easy" | "normal" | "hard" | "hardest".

### **ign\_ruptela**

For Ruptela devices. Represents configuration parameters related to ignition detection ("Engine detection" and "Custom ignition", as Ruptela documentation calls them).

[JSON-schema](#):

```
{
  "$schema" : "http://json-schema.org/draft-07/schema#",
  "type" : "object",
  "properties" : {
    "mode" : {
      "$ref" : "ruptela_ignition_mode.json"
    },
    "use_voltage" : {
      "type" : [ "boolean", "null" ]
    },
    "voltage" : {
      "type" : [ "number", "null" ]
    }
  },
  "required" : [ "mode" ],
  "$id" : "ruptela-ignition.json"
}

{
  "$schema" : "http://json-schema.org/draft-07/schema#",
  "type" : "string",
  "enum" : [ "always_on", "din", "movement_sensor", "custom" ],
  "$id" : "ruptela_ignition_mode.json"
}
```

### **ign\_src\_suntech**

```
{
  "type": "ign_src_suntech",
  "mode": "power_voltage",
  "power_voltage_low_level": 12000,
  "power_voltage_high_level": 19000
}
```

- `mode` - `enum`. Can be "power\_voltage" | "din1" | "movement".
- `power_voltage_low_level` - int. Can be 0 - 30000.
- `power_voltage_high_level` - int. Can be 0 - 30000.

### ign\_src\_telfm

```
{
  "type": "ign_src_telfm",
  "mode": "power_voltage",
  "power_voltage_low_level": 12000,
  "power_voltage_high_level": 24000
}
```

- `mode` - `enum`. Can be "power\_voltage" | "din1" | "movement".
- `power_voltage_low_level` - int. Can be 0 - 30000.
- `power_voltage_high_level` - int. Can be 0 - 30000.

### locus\_sec

```
{
  "type": "locus_sec",
  "signature": "signature",
  "sms_password": "23145",
  "reset": false
}
```

- `signature` - string. Length 1 - 32.
- `sms_password` - string. Length 1 - 32.
- `reset` - boolean.

### phonebook\_gt300

```
{
  "type": "phonebook_gt300",
  "capacity": 20,
  "items": [{ "name": "Karl", "phone": "555469874" }]
}
```

- `items` - array of contacts.
  - `name` - string. Contact name.

- `phone` - string. Phone number in the international format without "+" sign.

### phonebook\_pt100

```
{
  "type": "phonebook_pt100",
  "capacity": 3,
  "items": [{ "name": "Karl", "phone": "555469874" }]
}
```

- `items` - array of contacts.
  - `name` - string. Contact name.
  - `phone` - string. Phone number in the international format without "+" sign.

### pwr\_off\_key

```
{
  "type": "pwr_off_key",
  "mode": "enable"
}
```

- `mode` - [enum](#). Can be "enable" | "disable".

### scat\_mayak\_bt\_control

```
{
  "type": "scat_mayak_bt_control",
  "function": "bt_disable",
  "bt_state": true
}
```

- `function` - [enum](#). Can be "bt\_disable" | "bt\_enable" | "bt\_clear" | "bt\_write".
- `bt_state` - boolean.

### sos\_key

```
{
  "type": "sos_key",
  "mode": "report",
  "phone": "55548875236"
}
```

- `mode` - [enum](#). Can be "report" | "call\_report".
- `phone` - string. SOS phone to call. Phone number in the international format without "+" sign.

### starcom\_impact

```
{
  "type": "starcom_impact",
  "strong_duration": 12,
  "strong_force": 4,
  "strong_impact_enabled": true,
  "weak_duration": 9,
  "weak_force": 6,
  "weak_impact_enabled": true
}
```

- `strong_duration` - int. Required impact duration to trigger strong impact event. Each unit equals 2.5 milliseconds. Can be 0 - 14.
- `strong_force` - int. Required impact force triggering strong impact event. Each unit equals about 1.1g. Can be 1 - 7.
- `strong_impact_enabled` - boolean.
- `weak_duration` - int. Required impact duration to trigger weak impact event. Each unit equals 2.5 milliseconds. Can be 0 - 14.
- `weak_force` - int. Required impact force triggering weak impact event. Each unit equals about 1.1g. Can be 1 - 7.
- `weak_impact_enabled` - boolean.

#### **tacho\_company\_card**

```
{
  "type": "tacho_company_card",
  "company_card_number": "A2332BF23EC3245A"
}
```

- `company_card_number` - string. 16 HEX digits (0-9A-F).

#### **tacho\_remote\_download**

```
{
  "type": "tacho_remote_download",
  "company_card_number": "A2332BF23EC3245A",
  "vu_download_interval": 10,
  "card_download_interval": 2
}
```

- `company_card_number` - string. 16 HEX digits (0-9A-F).
- `vu_download_interval` - int. Min = 0.
- `card_download_interval` - int. Min = 0.

#### **teltonika\_tacho\_request**

```
{
  "type": "teltonika_tacho_request",
  "data_type": "activities",
  "activities_start_time": "2020-09-01",
  "activities_end_time": "2020-09-16"
}
```

- `data_type` - [enum](#). Can be "overview" | "activities" | "eventsAndFaults" | "detailedSpeed" | "technicalData" | "card1Download" | "card2Download".
- `activities_start_time` - string date. Format = "YYYY-MM-DD", not null only if `data_type` = "activities".
- `activities_end_time` - string date. Format = "YYYY-MM-DD", not null only if `data_type` = "activities".

### temporary\_digital\_password

```
{
  "type": "temporary_digital_password",
  "password": "231578",
  "duration_in_min": 17
}
```

- `password` - string. 6 digits.
- `duration_in_min` - int. Can be 10 - 255.

### time\_shift

```
{
  "type": "time_shift",
  "offset": 3.0
}
```

- `offset` - double. Can be -24.0 - 24.0 hours.

### tow\_detection\_ql

```
{
  "type": "tow_detection_ql",
  "mode": "enable",
  "engine_off_to_tow": 300,
  "fake_tow_delay": 300,
  "tow_interval": 12000,
  "rest_duration": 90,
  "motion_duration": 8300,
  "motion_threshold": 3
}
```

- `mode` - [enum](#). Can be "enable" | "disable".

- `engine_off_to_tow` - int. A time parameter to judge whether the device considered towed after the engine off. If the motion sensor doesn't detect stillness within the specified time after the engine off the device is being towed. Can be 0 - 900 seconds.
- `fake_tow_delay` - int. After the engine off and stillness detected, if motion sensor detects moving again, the device turns into a state called fake tow. If the device keeps in fake tow after a period defined by this parameter, it is considered towed. Can be 0 - 600 seconds.
- `tow_interval` - int. The period to send alarm messages. Can be 0 - 86400 seconds.
- `rest_duration` - int. A time parameter to make sure the device enters stillness status, i.e. the status of the device will be changed to stillness if the motion sensor detects stillness and maintains for a period defined by this parameter. Can be 0 - 3825 seconds, step 15.
- `motion_duration` - int. A time parameter to make sure the device enters motion status. Can be 0 - 9900 milliseconds, step 100.
- `motion_threshold` - int. The threshold for the motion sensor to measure whether the device is moving. Can be 2 - 9.

## **tow\_detection\_ql2**

```
{
  "type": "tow_detection_ql2",
  "mode": "enable",
  "engine_off_to_tow": 300,
  "fake_tow_delay": 300,
  "tow_interval": 12000,
  "rest_duration": 90,
  "motion_duration": 400,
  "motion_threshold": 3
}
```

- `mode` - [enum](#). Can be "enable" | "disable".
- `engine_off_to_tow` - int. A time parameter to judge whether the device considered towed after the engine off. If the motion sensor doesn't detect stillness within the specified time after the engine off the device is being towed. Can be 0 - 900 seconds.
- `fake_tow_delay` - int. After the engine off and stillness detected, if motion sensor detects moving again, the device turns into a state called fake tow. If the device keeps in fake tow after a period defined by this parameter, it is considered towed. Can be 0 - 600 seconds.
- `tow_interval` - int. The period to send alarm messages. Can be 0 - 86400 seconds.

- `rest_duration` - int. A time parameter to make sure the device enters stillness status, i.e. the status of the device will be changed to stillness if the motion sensor detects stillness and maintains for a period defined by this parameter. Can be 0 - 3825 seconds, step 15.
- `motion_duration` - int. A time parameter to make sure the device enters motion status. Can be 100 - 1000 milliseconds, step 100.
- `motion_threshold` - int. The threshold for the motion sensor to measure whether the device is moving. Can be 2 - 9.

### **tow\_detection\_telfm**

```
{
  "type": "tow_detection_telfm",
  "mode": "enable",
  "activation_timeout": 5,
  "threshold": 0.30
}
```

- `mode` - `enum`. Can be "enable" | "disable".
- `activation_timeout` - int. Can be 0 - 65535 minutes.
- `threshold` - double. Can be 0.10 - 5.00.

### **video\_stream\_howen**

```
{
  "type": "video_stream_howen"
}
```

### **virtual\_ign\_ql**

```
{
  "type": "virtual_ign_ql",
  "mode": "motion_sensor",
  "ign_on_voltage": 12000,
  "rest_duration_to_off": 120,
  "motion_duration_to_on": 75
}
```

- `mode` - `enum`. Can be "disabled" | "power\_voltage" | "motion\_sensor".
- `ign_on_voltage` - int. Can be 250 - 28000.
- `rest_duration_to_off` - int. A time parameter to make sure the device enters stillness status, i.e. the status of the device will be changed to stillness if the motion sensor detects stillness and maintains for a period of time defined by this parameter. Can be 1 - 255.

- `motion_duration_to_on` – A time parameter to make sure the device enters motion status. Can be 1 - 255.

### **no\_movement\_alarm**

```
{
  "type": "no_movement_alarm",
  "enabled": true,
  "timeout": 300,
  "pre_alarm_duration": 120
}
```

- `timeout` - int. Can be 30 - 65500. A time parameter when the device doesn't move.
- `pre_alarm_duration` - int. Can be 0 - 65500. A time parameter when the device continues not to move after timeout.

### **errors**

- 201 – Not found in the database (if there is no tracker with such ID belonging to authorized user).
- 208 – Device blocked (if tracker exists but was blocked due to tariff restrictions or some other reason).
- 214 – Requested operation or parameters are not supported by the device.

## **update**

Sets special settings for a specified tracker with the new one.

**required sub-user rights:** `tracker_configure`.

### **parameters**

name	description	type
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int
value	Settings object, see above.	JSON object



## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/settings/special/
update' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
123456, "value": {"type": "time_shift", "offset": 3.0}}'
```

## response

```
{ "success": true }
```

## errors

- 201 – Not found in the database - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 214 – Requested operation or parameters are not supported by the device.

Last update: December 26, 2022





# Specific portlets

## Specific portlets

Specific portlets that are used for models of three device manufacturers:

- Engine event behavior for ATrack.
- Guard mode for Yatut.
- Harsh behavior for Suntech.

### engine\_control\_atrack

Special settings to set the engine event behavior for ATrack.

```
{
  "power_voltage_high_level": 13800,
  "on_duration_seconds": 120,
  "power_voltage_low_level": 12800,
  "off_duration_seconds": 300
}
```

- `power_voltage_high_level` - int. Voltage in 0.001 volts for detecting engine ON state. Min=0, max=30000, default=13800 mV.
- `on_duration_seconds` - int. Duration in seconds that must elapse before the engine state change accepted. Min=0, max=600, default=1 second.
- `power_voltage_low_level` - int. Voltage in 0.001 volts for detecting engine OFF state. Min=0, max=30000, default=12800 mV.
- `off_duration_seconds` - duration in seconds that must elapse before the engine state change accepted. Min=0, max=600, default=5 seconds.

### guard\_mode\_yatut

Guard special settings for "Я ТУТ ПОИСК".

```
{
  "motion_sensor_mode": "double_period",
  "motion_sensor_first_period": "23:00-07:00",
  "motion_sensor_second_period": "10:00-17:00",
  "motion_sensor_amplitude": 10,
  "motion_sensor_duration": 30,
}
```

```

    "motion_sensor_ignore_time": 50,
    "motion_sensor_double_check": false,
    "perimeter_mode": "once_triggering",
    "perimeter_diameter": 1
}

```

- `motion_sensor_mode` - [enum](#). Can be "off" | "permanent" | "single\_period" | "double\_period". Default="off".
- `motion_sensor_first_period` - string time. Format= HH:mm-HH:mm, default="23:00-07:00" Required for `motion_sensor_mode` in single\_period/double\_period.
- `motion_sensor_second_period` - string time. Format= HH:mm-HH:mm, default="10:00-17:00" Required for `motion_sensor_mode` in double\_period.
- `motion_sensor_amplitude` - int. Min=1, max=255, default=5 Required for `motion_sensor_mode != off`.
- `motion_sensor_duration` - int. Min=1, max=255, default=5 seconds. Required for `motion_sensor_mode != off`.
- `motion_sensor_ignore_time` - int. Min=5, max=99, default=5 minutes. Required for `motion_sensor_mode != off`.
- `motion_sensor_double_check` - boolean. Default= `false`. Required for `motion_sensor_mode != off`.
- `perimeter_mode` - [enum](#). Can be "off" | "once\_triggering" | "permanent" | "point\_displacement". Default="off".
- `perimeter_diameter` - int. Min=1, max=999, default=1 kilometer. Required for `perimeter_mode != off`.

## harsh\_behavior\_suntech

Harsh driving settings for Suntech.

```

{
    "mode": "enable",
    "max_acceleration_force": 1.5,
    "max_braking_force": 0.05,
    "max_cornering_force": 3,
    "type": "harsh_behavior_suntech"
}

```

- `mode` - string. Can be "enable" | "disable".
- `max_acceleration_force` - double. Can be 0.05 – 3.0 g.
- `max_braking_force` - double. Can be 0.05 – 3.0 g.

- `max_cornering_force` - double. Can be 0.05 – 3.0 g.

Last update: October 31, 2021







# Engine hours

Contains API call to read engine hours (time when engine is on) counted for the specified period.

## API actions

API base path: `/tracker/stats/engine_hours`.

### read

Returns engine hours counted for the specified period.

#### parameters

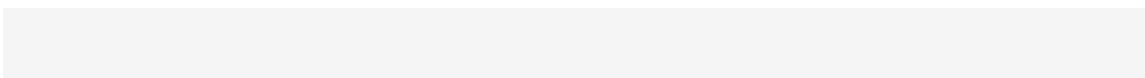
name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
from	From date/time.	<a href="#">date/</a> <a href="#">time</a>	"2020-09-24 03:24:00"
to	To date/time. Specified date must be after "from" date.	<a href="#">date/</a> <a href="#">time</a>	"2020-09-24 06:24:00"

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/stats/engine_hours/  
read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":  
123456, "from": "2020-09-24 03:24:00", "to": "2020-09-24  
06:24:00"}'
```

#### response



```
{  
  "success": true,  
  "value": 42.0  
}
```

## errors

- 204 – Entity not found - if there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked - if tracker exists but was blocked due to tariff restrictions or some other reason.
- 211 – Requested time span is too big - if interval between "from" and "to" is too big (maximum value specified in API config).
- 214 – Requested operation or parameters are not supported by the device - if device does not have ignition input.
- 219 – Not allowed for clones of the device - if specified tracker is a clone.

Last update: December 26, 2022





# Mileage

Contains API call to read mileage counted for the specified period.

## API actions

API base path: `/tracker/stats/mileage`.

### read

Returns mileage in kilometers in specified period grouped by trackers and day.

#### parameters

name	description	type
trackers	Array of tracker IDs (aka "object_id"). Trackers must belong to authorized user and not be blocked.	int array
from	From date/time.	date/ time
to	To date/time. Specified date must be after "from" date.	date/ time

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/tracker/stats/mileage/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "trackers": [123456], "from": "2020-09-24 03:24:00", "to": "2020-09-24 06:24:00"}'
```

#### response

```
{
  "success": true,
  "result": {
    "123456": {
```

```
    "2000-01-01": { "mileage": 0.0 },
    "2000-01-02": { "mileage": 0.0 },
    "2000-01-03": { "mileage": 199.09 }
  },
  "limit_exceeded": false
}
```

## errors

- 211 – Requested time span is too big - if interval between "from" and "to" is too big (maximum value specified in API config).
- 217 – List contains nonexistent entities.
- 221 – Device limit exceeded.

Last update: December 26, 2022







# Working with geofences

Geofences used in rules to limit rule area of activity. Also, geofence names shown in reports after the address, if an event happened inside the geofence.

This document describes CRUD actions for geofences. Note that geofence points handled separately because they are represented by big arrays of data.

Find instructions on working with geofences of different types [here](#).

## Entity description

**zone** is JSON object with one of types: `sausage`, `circle` or `polygon`.

**circle:**

```
{
  "id": 985472,
  "type": "circle",
  "label": "Zone name",
  "address": "Karlsplatz, 2",
  "color": "27A9E3",
  "radius": 150,
  "center": {
    "lat": 48.200940,
    "lng": 16.369856
  },
  "tags": [127, 15]
}
```

- `id` - int. Geofence ID.
- `label` - string. Geofence label.
- `address` - string. Geofence address.
- `color` - string. Geofence color in 3-byte RGB hex format.
- `radius` - int. Circle radius in meters.
- `center` - location object. Location of circle center.
- `tags` - int array. Array of tag IDs.

**polygon:**

```
{
  "id": 124597,
  "type": "polygon",
  "label": "Geofence name",
  "address": "Karlsplatz, 2",
  "color": "27A9E3",
  "tags": [1,236]
}
```

- `id` - int. Geofence ID.
- `label` - string. Geofence label.
- `address` - string. Geofence address.
- `color` - string. Geofence color in 3-byte RGB hex format.
- `tags` - int array. Array of tag IDs.

#### **sausage:**

Represents all points within certain distance to the specified polyline.

```
{
  "id": 12345,
  "type": "sausage",
  "label": "Geofence name",
  "address": "Karlsplatz, 2",
  "color": "27A9E3",
  "radius": 150,
  "tags": [289]
}
```

- `id` - int. Geofence ID.
- `label` - string. Geofence label.
- `address` - string. Geofence address.
- `color` - string. Geofence color in 3-byte RGB hex format.
- `radius` - int. Polyline radius in meters.
- `tags` - int array. Array of tag IDs.

## API actions

API base path: `/zone` .

## batch\_convert

Convert batch of tab-delimited circle geofences and return list of checked geofences with errors.

**required sub-user rights:** `zone_update`.

### parameters

name	description	type
batch	Batch of tab-delimited places.	string
file_id	ID of file preloaded with <a href="#">/data/spreadsheet/parse</a> method.	string
fields	Optional, array of field names, default is <code>["label", "address", "lat", "lng", "radius", "tags"]</code> .	enum array
geocoder	Optional. Geocoder type.	enum
default_radius	Optional. Radius for point, default is 100.	int

If 'file\_id' is set – 'batch' parameter will be ignored.

For `batch` parameter: \* address - required if no coordinates specified. \* lat - required if no address specified. \* long - required if no address specified.

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/zone/batch_convert' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "batch":  
  "Geofence for test Karlsplatz, 2"}'
```

### response

```
{  
  "success": true,  
  "list": [{  
    "id": null,  
    "type": "circle",  
    "label": "Geofence name",  
    "address": "Karlsplatz, 2",  
    "color": "27A9E3",  
  }]
```

```

    "radius": 100,
    "center": {
      "lat": 48.2009935,
      "lng": 16.3699642
    },
    "tags": []
  }],
  "limit_exceeded": false
}

```

- `id` - int. Geofence ID.
- `label` - string. Geofence label.
- `address` - string. Geofence address.
- `color` - string. Geofence color in 3-byte RGB hex format.
- `radius` - int. Circle radius in meters.
- `center` - location object. Location of circle center.
- `tags` - int array. Array of tag IDs.
- `limit_exceeded` - boolean. `true` if given batch constrained by limit.

#### response with errors object

```

{
  "success": true,
  "list": [{
    "id": null,
    "label": "Geofence name",
    "address": "incorrect address",
    "color": "27A9E3",
    "radius": 100,
    "center": {
      "lat": 0.0,
      "lng": 0.0
    },
    "errors": [{
      "parameter": "zone.center",
      "error": "Location should be correct with 'lat' and 'lng'
not null"
    }],
    "tags" : []
  }],
  "limit_exceeded": false
}

```

- `errors` - optional object. It appears if parameters incorrect.
  - `parameter` - string. Parameter name.
  - `error` - string. Error description.

## errors

- 234 - Invalid data format.

## create

Creates a new geofence.

**required sub-user rights:** `zone_update`.

## parameters

name	description	type
zone	zone JSON-object without <code>id</code> and <code>color</code> fields.	JSON object
points	Array of new <a href="#">points</a> for this geofence. Must contain at least 3 elements. MUST be omitted if zone does not support points (e.g. circle).	array of <code>zone point</code> objects
zone.color	Optional. Geofence color in 3-byte RGB hex format. Default is "27A9E3".	string

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/zone/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "zone": {"label": "Circle geofence", "type": "circle", "center": {"lat": 61.49504550221769, "lng": 23.775476217269897}, "radius": 50, "tags": [179227], "color": "03A9F4", "address": "Address"}}'
```

## response

```
{
  "success": true,
  "id": 1234567
}
```

- `id` - int. An ID of the created geofence.

## errors

- 202 - Too many points in a geofence – max allowed points count for a geofence is 100 for a polygon or 1024 for sausage.
- 230 - Not supported for this entity type – if "points" were specified, but geofence cannot have any points associated with it (e.g. if geofence is circle).
- 268 - Over quota – if the user's quota for geofences exceeded.

## delete

Deletes user's geofence by `zone_id` or array of `zone_ids`.

**required sub-user rights:** `zone_update`.

## parameters

name	description	type	format
zone_id	ID of a geofence.	int	1234567
zone_ids	Array of geofence IDs.	int array	[1234567, 2345678]

- Use only one parameter `zone_id` or `zone_ids`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/zone/delete' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "zone_id":  
1234567}'
```

### HTTP GET

```
https://api.navixy.com/v2/zone/delete?  
hash=a6aa75587e5c59c32d347da438505fc3&zone_id=1234567
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database.

- 203 - Delete entity associated with.

```
{
  "success": false,
  "status": {
    "code": 203,
    "description": "Delete entity associated with"
  },
  "entities": [
    {
      "type": "rules",
      "ids": [12345, 23456]
    }
  ]
}
```

- `ids` - int array. List IDs of the rules which uses the specified geofence.

## list

Gets all user geofences.

### parameters

name	description	type
filter	Optional. Filter for geofences label and description.	string
tag_ids	Optional. Tag IDs assigned to the geofences. The zones found must include all the tags from the list.	int array
offset	Optional. Offset from start of the found geofences for pagination.	int
limit	Optional. Limit of the found geofences for pagination.	int
with_points	Optional, default= <code>false</code> . If <code>true</code> , return geofence with its <code>points</code>	boolean

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/zone/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

### HTTP GET

```
https://api.navixy.com/v2/zone/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

## response

```
{
  "success": true,
  "list": [{
    "id": 12345,
    "type": "sausage",
    "label": "Zone name",
    "address": "Karlsplatz, 2",
    "color": "27A9E3",
    "radius": 150,
    "tags": [289]
  }]
}
```

- `list` - array of objects. Geofence objects without points field.

## read

Gets geofence by specified ID.

### parameters

name	description	type
zone_id	ID of a geofence.	int
with_points	Optional, default= false . If true , return geofence with its points	boolean



## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/zone/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "zone_id": 12345}'
```

### HTTP GET

```
https://api.navixy.com/v2/zone/read?
hash=a6aa75587e5c59c32d347da438505fc3&zone_id=12345
```

## response

```
{
  "success": true,
  "value": {
    "id": 12345,
    "type": "sausage",
    "label": "Zone name",
    "address": "Karlsplatz, 2",
    "color": "27A9E3",
    "radius": 150,
    "tags": [289]
  }
}
```

- `value` - Geofence object without points field.

## search\_location

Gets all geofence IDs and names within which a specified coordinates are located inside.

### parameters

name	description	type
location	Location coordinates (see: <a href="#">data types description section</a> section).	JSON object

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/zone/search_location' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "location": {"lat": 34.178868, "lng": -118.599672}}'
```

## response

```
{
  "list": [
    {
      "id": 18404,
      "label": "geofence 1"
    },
    {
      "id": 35284,
      "label": "geofence 2"
    }
  ],
  "success": true
}
```

- `id` - int. Geofence ID that containing a searched location.
- `label` - string. Geofence name.

## update

Update geofence parameters for the specified geofence. Note that geofence must exist, must belong to the current user, and its type cannot be changed. E.g. if you already have a geofence with ID=1 which type is "circle", not possible to submit a geofence which type is "polygon".

**required sub-user rights:** `zone_update`.

### parameters

name	description	type
zone	Geofence JSON-object with <code>id</code> and <code>color</code> fields.	JSON object
zone.color	Optional. Geofence color in 3-byte RGB hex format. Default is "27A9E3".	string

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/zone/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "zone":
{"id": 231512 "label": "Circle geofence", "type": "circle",
"center": {"lat": 61.49504550221769, "lng": 23.775476217269897},
"radius": 50, "tags": [179227], "color": "03A9F4",
"address": "Address"}}'
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if geofence with the specified ID cannot be found or belongs to another user.
- 231 - Entity type mismatch – if type of the submitted geofence differs from type of the geofence currently stored in the database.

## upload

Import geofences from KML file.

**required sub-user rights:** `zone_update`.

**MUST** be a POST multipart request (multipart/form-data), with one of the parts being a KML file upload (with the name "file").

## parameters

name	description	type
file	A KML file upload containing geofences data.	file upload
default_radius	Default radius for circle and route geofence in meters. Min 20, default 150.	int
dry_run	If <code>true</code> returns ready to create geofences or creates it and returns list of IDs otherwise. Default <code>true</code> .	boolean

name	description	type
redirect_target	Optional. URL to redirect. If <b>redirect_target</b> passed return redirect to <redirect_target>? response=<urlencoded_response_json>	string

## responses

if `dry_run=true`:

```
{
  "success": true,
  "list": [
    {
      "id": null,
      "label": "Simple line 1",
      "address": "",
      "color": "27A9E3",
      "points": [
        {
          "lat": 37.818844,
          "lng": -122.366278,
          "node": true
        },
        {
          "lat": 37.819267,
          "lng": -122.365248,
          "node": false
        },
        {
          "lat": 37.819861,
          "lng": -122.36564,
          "node": false
        },
        {
          "lat": 37.819429,
          "lng": -122.366669,
          "node": true
        }
      ],
      "radius": 150,
      "type": "sausage"
    }
  ]
}
```

if `dry_run=false`:

```
{
  "success": true,
```

```
"list": [ 1, 2 ]  
}
```

## errors

- 202 - Too many points in a geofence – max allowed points count for a geofence is 100 for a polygon or 1024 for sausage.
- 233 - No data file – if file part is missing.
- 234 - Invalid data format.
- 268 - Over quota – if the user's quota for geofences exceeded.

From `Placemark` with `Point` geometry will be created circle geofence with a `radius=default_radius`.

```
<?xml version="1.0" encoding="UTF-8"?>  
<kml xmlns="http://www.opengis.net/kml/2.2">  
  <Document>  
    <name>Points</name>  
    <Placemark>  
      <name>named point</name>  
      <Point>  
        <coordinates>  
          -122.366278,37.818844,30  
        </coordinates>  
      </Point>  
    </Placemark>  
  </Document>  
</kml>
```

From `Placemark` with `LineString` geometry will be created route geofence with a `radius=default_radius`.

```
<?xml version="1.0" encoding="UTF-8"?>  
<kml xmlns="http://www.opengis.net/kml/2.2">  
  <Document>  
    <name>Simple line</name>  
    <Placemark>  
      <LineString>  
        <coordinates>  
          -122.366278,37.818844,30  
          -122.365248,37.819267,30  
          -122.365640,37.819861,30  
          -122.366669,37.819429,30  
        </coordinates>  
      </LineString>  
    </Placemark>  
  </Document>  
</kml>
```

```
    </Document>
</kml>
```

From `Placemark` with `Polygon` geometry will be created polygon geofence. Polygons with holes not supported. In that case only the outer boundary will be imported and the inner boundary, holes, ignored.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <name>Simple polygon</name>
    <Placemark>
      <name>hollow box</name>
      <Polygon>
        <outerBoundaryIs>
          <LinearRing>
            <coordinates>
              -122.366278,37.818844,30
              -122.365248,37.819267,30
              -122.365640,37.819861,30
              -122.366669,37.819429,30
              -122.366278,37.818844,30
            </coordinates>
          </LinearRing>
        </outerBoundaryIs>
      </Polygon>
    </Placemark>
  </Document>
</kml>
```

From `Placemark` with `MultiGeometry` geometry will be created several geofences. If `Placemark.name` defined it will be used as geofence name with respect of hierarchy of `Folder` and `Document`.

Last update: January 15, 2024







# Geofence point

All actions to retrieve and manipulate points of the geofence. Note that `circle` geofence type can't have points.

## Point object structure

```
{
  "lat": 11.0,
  "lng": 22.0,
  "node": true
}
```

- `lat` - float. Point latitude.
- `lng` - float. Point latitude.
- `node` - boolean. Will be `true` if this point is a route node.

## API actions

API base path: `/zone/point`.

### list

Get points of user's geofence with `zone_id`.

#### parameters

name	description	type	format
zone_id	ID of a geofence.	int	1234567
count	Optional. If specified, the returned list will be simplified to contain this number of points.	int	300

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/zone/point/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "zone_id": 1234567}'
```

### HTTP GET

```
https://api.navixy.com/v2/zone/point/list?
hash=a6aa75587e5c59c32d347da438505fc3&zone_id=1234567
```

## response

```
{
  "success": true,
  "list": [{
    "lat": 11.0,
    "lng": 22.0,
    "node": true
  }]
}
```

- `list` - array of objects. List of point objects.

## errors

- 201 - Not found in the database – if geofence with the specified ID cannot be found or belongs to another user.
- 230 - Not supported for this entity type – if geofence cannot have any points associated with it (e.g. if geofence is circle).

## update

Update points for user's geofence with `zone_id`.

**required sub-user rights:** `zone_update`.

## parameters

name	description	type
zone_id	ID of a geofence.	int
points		

name	description	type
	Array of new points for this geofence. Must contain at least 3 elements. Maximum number of points depends on geofence type.	array of JSON objects

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/zone/point/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "zone_id": 1234567, "points": [{"lat": 11.0, "lng": 22.0, "node": true}, {"lat": 11.2, "lng": 22.2, "node": true}, {"lat": 11.4, "lng": 22.4, "node": true}]}'
```

## response

```
{ "success": true }
```

## errors

- 201 - Not found in the database – if geofence with the specified ID cannot be found or belongs to another user.
- 202 - Too many points in a geofence – if "points" array size exceeds limit for this geofence type. Max allowed points count for a geofence is 100 for a polygon or 1024 for sausage.
- 230 - Not supported for this entity type – if geofence cannot have any points associated with it (e.g. if geofence is circle).

Last update: December 26, 2022





# WebSocket API

Information about WebSocket API and how to open connection.

## Introduction

`WebSocket` is the alternate transport to getting data from the server. The process of notification about events occurs from the server to the client through a constantly open connection. This allows you to display changes in real time.

Currently, the [Atmosphere Framework](#) used as an application layer library and protocol.

## Standard workflow

Let's describe a standard workflow for WebSocket API:

1. Determine [API base URL](#).
2. Get the hash of an [API Key](#).
3. Open WebSocket connection by the path `/event/subscription/` with `Atmosphere` protocol parameters.
4. Subscribe on events using [subscribe action](#).
5. Listen and process the [incoming events](#).
6. Get the current tracker states after subscribe on a `state` event.
7. Subscribe and unsubscribe on the events if needed.
8. Unsubscribe when leaving monitoring page using [unsubscribe action](#).

### Note

- The [subscription requests](#) must contain the `hash` of an [API Key](#).
- Responses and errors for [subscribe](#) and [unsubscribe](#) actions are similar with common [API](#) format.
- All `WebSocket` frames use a `JSON` format. Exceptions are heartbeat frames containing "X".

## Open connection

In a simplified form, opening the WebSocket using [atmosphere-javascript](#) looks like this:

```
var subSocket;

function sendSubsrcibeRequest() {
    console.log('sending subsrcibe action to websocket');
    subSocket.push(JSON.stringify({
        action: 'subscribe',
        hash: 'e4c24xxx4a08e9xxxc337xxx5ca04e1',
        requests: [
            {
                type: 'state_batch',
                target: {
                    type: 'all'
                }
            }
        ]
    }));
}

var request = {
    url: 'https://api.navixy.com/v2/event/subscription',
    contentType : "application/json",
    logLevel : 'debug',
    transport : 'websocket',
    trackMessageLength : false,
    reconnectInterval: 2000,
    onOpen: function(r) {
        console.log('onOpen', r);
        sendSubsrcibeRequest();
    },
    onReopen: function(r) {
        console.log('onReopen', r);
        sendSubsrcibeRequest();
    },
    onMessage: function (msg) {
        console.log('onMessage', msg);
    },
    onClientTimeout: function(r) {
        console.log('onClientTimeout', r);
    },
    onTransportFailure: function(errorMsg, request) {
        console.log('onTransportFailure', errorMsg);
    },
    onClose: function(r) {
        console.log('onClose', r);
    },
    onError: function(r) {
        console.log('onError', r);
    },
    onReconnect: function(request, response) {
        console.log('onReconnect', response);
    }
}
```

```
    }  
};  
  
subSocket = atmosphere.subscribe(request);
```

Executing this code will lead to send a request

```
wss://domain.com/event/subscription?X-Atmosphere-tracking-id=0&X-  
Atmosphere-Framework=2.3.6-javascript&X-Atmosphere-  
Transport=websocket&Content-Type=application/json&X-atmo-  
protocol=true
```

and upgrade the connection to the WebSocket. After that, will be sent a first frame through the opened WebSocket channel:

```
b623a15d-9623-4fd8-a9d3-697036635c29|30000|X|
```

This is service message for the Atmosphere protocol negotiation. Now everything is ready to [subscribe on events](#).

## Common fields

All messages from client side contain field `action` with action name (e.g. "subscribe" or "unsubscribe").

All messages from server side contain field `type` with message type ("event", "response" or "error") and `data` with a payload.

Last update: March 10, 2023







# WebSocket Subscription

The `subscribe` and `unsubscribe` actions used by the client's side to subscribe on server events and unsubscribe from them. These actions are similar with any other [API REST actions](#) but must be sent inside open `WebSocket` channel and use only JSON format for messages between the client and server.

## Subscribe Action

The main difference between `state` and `state_batch` events is they can provide different amount of data every second. Use 'state' event for smaller fleets since it supports sending data up to 350 entries per second. For big or growing fleets better to use `state_batch` event since it supports sending data for up to 12000 entries per second.

## Request

Request parameters:

- `action` (text: "subscribe").
- `hash` - required, string, length=32. Session hash code obtained by [user/auth](#) action.
- `requests` - required, object array. See requests' structure below.

### Deprecated

Parameters below are deprecated by `requests` and should not be used.

- `trackers` - required, int array, without nulls. List of tracker IDs for the events that require a subscription.
- `events` - required, [enum](#) array, without nulls. List of events to subscribe. Event can be one of: `state`.

**Subscribe `state_batch` event sample:**

```
{
  "action": "subscribe",
  "hash": "f4bf1b754034213653dad99c78c4b237",
  "requests": [
    {
```

```

        "type": "state_batch",
        "target": {
            "type": "all"
        },
        "rate_limit": "5s",
        "format": "compact"
    }
]
}

```

If you use target 'trackers' for some devices and then subscribe again to other devices - in state\_batch event you will receive data from all subscribed devices at once.

### Subscribe state event sample

```

{
  "action": "subscribe",
  "hash": "4ce2b45d12a6c634154017511575369a",
  "requests": [
    {
      "type": "state",
      "trackers": [
        1701976,
        1701975
      ],
      "rate_limit": "5s",
      "format": "compact"
    }
  ]
}

```

### Sub requests:

- Batching (preferred):
  - `type` - required, text: "state\_batch".
  - `target` - required, object. One of targets below.
  - `rate_limit` - optional, string. A timespan for batching.
  - `format` - optional, [enum](#), one of: "full" (default), "compact".
- Simple:
  - `type` - required, text: "state".
  - `trackers` - required, int array. List of tracker ids.
  - `format` - optional, [enum](#), one of: "full" (default), "compact".

Sample:

```

{
  "type": "state_batch",
  "target": {
    "type": "selected",

```

```

    "tracker_ids": [15564, 15565, 15568]
  },
  "rate_limit": "5s",
  "format": "full"
}

```

#### REQUEST TARGETS:

- All trackers:
- `type` - required, text: "all".
- Selected trackers:
- `type` - required, text: "selected".
- `tracker_ids` - required, int array.

Sample:

```

{
  "type": "all"
}

```

## Response

Response parameters:

- `type` - required, text: "response".
- `action` - required, text: "subscription/subscribe".
- `events` - required, array of [enum](#), without nulls. List of the subscribed events. Event can be `state`.
- `data` - required, map . Map with events subscription result. One key per subscribed event.
- `state` - present if the "state" subscription requested, see sub response below.
- `state_batch` - present if the "state\_batch" subscription requested, see sub response below.

Sub response: \* `success` - required, boolean. \* `value` - required, map , present if success. The current status of requested trackers.

Keys is a tracker IDs, values - one of the item:

- `normal` - non-blocked, normal status. [State events](#) for this tracker will be delivered to client.
- `blocked` - tracker blocked. [State events](#) for this tracker will *not* be delivered to client.

[Lifecycle events](#) will be delivered. After unblocking, current tracker state will be sent automatically.

- `unknown` - tracker ID missed in the database or not belong to current user.
- `disallowed` - subscription for this tracker not allowed by the current session.

Response sample:

```
{
  "type": "response",
  "action": "subscription/subscribe",
  "events": ["state"],
  "data": {
    "state": {
      "value": {
        "15564": "normal",
        "15565": "blocked",
        "15568": "unknown"
      },
      "success": true
    }
  }
}
```

## The "state" event subscription

After subscribe on the "state", server will send the current states of all non-blocked trackers to which the subscription made in a separate packets. Receiver must be able to read information from these packets separately. When changing the state of any tracker to which a subscription made, the server will send a new state in the [event message](#).

## The "state\_batch" event subscription

After subscribe on the "state", server will send the current states of all non-blocked trackers to which the subscription made in one packet. Receiver must be able to parse data from different devices in this packet. After each period equal to `rate_limit`, the server will send a list of changed tracker states in the [event message](#).

## Automatic subscriptions

- Subscribing to a `state` or `state_batch` automatically creates a subscription to [lifecycle events](#).
- Subscribing to any event automatically creates a subscription to [logout events](#).

# Unsubscribe Action

For structure see [Subscribe Action](#).

## Error Response

If something goes wrong, the server may respond with an error. Error codes are similar to the [API errors codes](#).

Error response parameters:

- `type` - required, text: `"error"`.
- `action` - required, string - action from request (e.g. `"subscription/subscribe"`) or `"null"` for some unexpected errors.
- `status` - required - error code and description:
- `code` - required - error code (see [API errors codes](#)).
- `description` - required, string - error description.
- `data` - optional string - part of parameters from request or some info for unexpected errors.

Error response sample:

```
{
  "type": "error",
  "action": "subscription/subscribe",
  "status": {
    "code": 3,
    "description": "Wrong hash"
  },
  "data": {
    "events": ["state"],
    "trackers": [15564]
  }
}
```

Last update: April 21, 2023








# WebSocket Events

The server sends an `event` message through the WebSocket channel when an event occurs and client has subscription on this. All event messages contain the next fields:

- `type` - `enum`. "event".
- `event` - `enum`. Can be "state", "state\_batch", "lifecycle", or "logout".
- `data` - optional object. Specific event payload.

## State event

 **Can be used for 100 devices. If there are more devices, please use state batch event**

These messages are coming from server if client `subscribed` to the `state` events of the specific tracker that not blocked. It occurs in the next cases:

- Tracker state changed.
- Immediately after subscription.
- Immediately after unblocking.

Message fields:

- `type` - "event".
- `event` - "state".
- `data` - depends on `format` request parameter:
- "full" - `source state`.
- "compact" - `compact source state`.
- `user_time` - current time in user's timezone.


Message sample:

```
{
  "type": "event",
  "event": "state",
  "user_time": "2018-10-17 12:51:55",
  "data": {
    "source_id": 10284,
    "gps": {
```

```

    "updated": "2018-10-17 12:51:43",
    "signal_level": 100,
    "location": {
      "lat": 14.330065796228606,
      "lng": -90.99037259141691
    },
    "heading": 248,
    "speed": 0,
    "alt": 431
  },
  "connection_status": "active",
  "movement_status": "parked",
  "gsm": null,
  "last_update": "2018-10-17 12:51:46",
  "battery_level": null,
  "battery_update": null,
  "inputs": [false, false, false, false, false, false, false, false,
false],
  "inputs_update": "2018-10-17 12:51:43",
  "outputs": [false, false, false, false, false, false, false, false,
false],
  "outputs_update": "2018-10-17 12:51:43",
  "actual_track_update": "2018-10-04 22:47:07"
}
}

```

 `source_id` **is not a** `tracker_id`.

## State batch event

These messages are coming from server if client [subscribed](#) to the `state_batch` events of the specific tracker that not blocked. It occurs in the next cases:

- Immediately after subscription.
- `rate_limit` period passed.

Message fields:


- `type` - "event".
- `event` - "state".
- `data` - depends on `format` request parameter:
- "full" - [source state](#) array.
- "compact" - [compact source state](#) array.
- `user_time` - current time in user's timezone.

Message sample:

```

{
  "type": "event",
  "event": "state_batch",
  "user_time": "2018-10-17 12:51:55",
  "data": [
    {
      "source_id": 10284,
      "gps": {
        "updated": "2018-10-17 12:51:43",
        "signal_level": 100,
        "location": {
          "lat": 14.330065796228606,
          "lng": -90.99037259141691
        },
        "heading": 248,
        "speed": 0,
        "alt": 431
      },
      "connection_status": "active",
      "movement_status": "parked",
      "gsm": null,
      "last_update": "2018-10-17 12:51:46",
      "battery_level": null,
      "battery_update": null,
      "inputs": [false, false, false, false, false, false, false, false,
false],
      "inputs_update": "2018-10-17 12:51:43",
      "outputs": [false, false, false, false, false, false, false, false,
false],
      "outputs_update": "2018-10-17 12:51:43",
      "actual_track_update": "2018-10-04 22:47:07"
    }
  ]
}

```

 source\_id **is not a** tracker\_id.

## Compact source state

Sample:

```

{
  "source_id": 10284,
  "gps": {
    "updated": "2018-10-17 12:51:43",
    "signal_level": 100,
    "location": {
      "lat": 14.330065796228606,
      "lng": -90.99037259141691
    },
    "heading": 248,

```

```
    "speed": 0,  
    "alt": 431  
  },  
  "connection_status": "active",  
  "movement_status": "parked",  
  "last_update": "2018-10-17 12:51:46"  
}
```

## Lifecycle event

These messages are coming from the server if client [subscribed](#) to the `state` events of the specific tracker. It occurs in the next cases:

- Tracker blocked.
- Tracker unblocked.
- Tracker corrupted (removed).

Message fields:

- `type` - "event".
- `event` - "lifecycle".
- `data` - required object.
  - `source_id` - source ID.
  - `lifecycle_event` - lifecycle event type. Can be "block", "unblock", or "corrupt".

Message sample:

```
{  
  "type": "event",  
  "event": "lifecycle",  
  "data": {  
    "source_id": 123456,  
    "lifecycle_event": "block"  
  }  
}
```

## Logout event

These messages are coming from server if client [subscribed](#) to any event. It occurs in the next cases:

- User logged out.

- User session expired (did not renew during one month).
- Sub-user is blocked by master-user.
- User has restored his password.
- User has changed his password.
- User blocked from admin panel.
- User was corrupted (removed).

Message fields:

- `type` - "event".
- `event` - "logout".
- `data` - "session closed".

Message sample:

```
{  
  "type": "event",  
  "event": "logout",  
  "data": "session closed"  
}
```

Last update: March 20, 2023







# Getting Started

## Navixy Panel API

The structure of Panel API (aka Administration API) – request paths, response and error formats – is the same as for user API, so we highly recommend reading [Backend API: getting started](#).

Two main differences are *authorization system* and *request paths*.

## Panel API base URL

Panel API resides in `panel/` subsection of API URL. So you can determine URL to API calls like this:

- `https://api.eu.navixy.com/v2/panel/` for European Navixy ServerMate platform.
- `https://api.us.navixy.com/v2/panel/` for American Navixy ServerMate platform.
- `https://api.your_domain/panel/` for the self-hosted (On-Premise) installations.

For example, to make `account/auth` API call in Navixy ServerMate, you should use the URL:

```
https://api.navixy.com/v2/panel/account/auth
```

## Authorization

In order to authorize, you should make a GET or POST request to `/account/auth/` with `login` (your administration panel login) and `password` (its password), which returns JSON object, containing `hash` (hexadecimal unique string) of the newly created Panel API session, which you should use in other Panel API calls.

Please note that you cannot use Panel API session hash in user API or vice versa.

You must keep in mind that string type containing any symbols except ASCII codes from 32 to 127 must be [URL encoded](#) before transfer.

For example, in on-premise installations, there is a default user with login `admin` and password `admin`. You can authorize with it like this (all HTTP request examples are made using [curl](#) \*nix utility):

## POST

```
$ curl -d 'login=admin&password=admin' \
-X POST http://api.domain.com/v2/panel/account/auth/
```

## GET

This method is not recommended. Just for example:

```
$ curl http://api.domain.com/panel/v2/account/auth/?
login=admin&password=admin
```

And you'll get answer like this:

```
{
  "hash": "1dc2b813769d846c2c15030884948117",
  "success": true,
  "permissions": { ... }
}
```

The value returned in `hash` field (in this example it is `1dc2b813769d846c2c15030884948117`) should be saved for further use.

If API call completes successfully, the HTTP response code is `200 OK`, and `success` field in returned JSON is equal to `true`.

If there is any error, for example, wrong credentials were used, HTTP response code differs from 200, `success` field is `false`, and `status` field contains the description of the error.

For example:

```
{
  "success": false,
  "status": {
    "code": 12,
    "description": "Dealer not found"
  }
}
```

The "description" field is just a human-readable hint, you should not check its contents programmatically as it may change in the future.

For more info, please see `account/auth`.

## Using session hash

After successful authorization, you can make other Panel API calls. You should **always** pass the session hash you obtained earlier as the `hash` parameter. This parameter is not listed in parameters list of every API call, but still required by default.

For example, to list first ten users belonging to your system account, you can use the following Panel API call (the hash is from previous example):

```
$ curl -X POST 'http://api.domain.com/v2/panel/user/list/' \
  -d 'hash=1dc2b813769d846c2c15030884948117&limit=10'
```

## Session expiration

Each session, and its associated hash key, has a limited lifespan that defaults to 24 hours. This is a change from the previous setting of 30 days, implemented for improved security. You need to periodically obtain a new hash key.

If you attempt to make a Panel API call with an expired session hash, you will receive an error:

```
{
  "success": false,
  "status": {
    "code": 4,
    "description": "User not found or session ended"
  }
}
```

In this case, just obtain new hash using `account/auth`.

## How to securely share panel's credentials

To share access to the admin panel, and at the same time not to worry about data security, we recommend contacting the technical support team to create a technical panel account. Provide the email address for which the technical account will be created. You will receive a login and password for the account.

The possibilities of tech account:

- Add new users
- Modify data of current users
- Add new trackers

- Clone current trackers
- Change owner of a tracker
- Change tracker data plan
- Analyze incoming data with the air console

Technicians are not allowed:

- Delete users
- Remove trackers
- Add, change, delete plans
- Change platform settings

## Panel API Permissions

Every call to panel api requires a set (possibly empty) of permissions. To determine if user is allowed to execute api call, user's permissions is compared with call's required permissions. If user does not have at least one required permission, the call is not executed and error "Operation not permitted" is returned.

Each permission is defined as a pair of category (e.g. `trackers`) and operation (e.g. `read`). A set of permissions within one category is often grouped as in the following example:

- `trackers` : create, read

This defines two permissions: (`trackers`, `create`) and (`trackers`, `read`).

List of all possible categories and operations:

- `accounting` : generate
- `activation_code` : create, read, update
- `base` : get\_dealer\_info
- `notification_settings` : read, update
- `service\_settings` : read, update
- `tariffs` : create, read, update
- `trackers` : create, read, update, delete
- `transactions` : create, read, update
- `users` : create, read, update, delete
- `user\_sessions` : create

- sms : create
- tracker\\_bundles : read, update

Last update: November 22, 2023





# Account

API calls on getting the panel's hash, getting permissions and logout.

## API actions

API path: `panel/account`.

### auth

Does not require session hash and does not need any permissions. Auths dealer in a panel (planned also for dealer's "sub-users") and gets hash.

#### parameters

name	description	type
login	A panel's login (number).	string
password	A panel's password.	string

#### example

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/account/auth' \  
-H 'Content-Type: application/json' \  
-d '{"login": "20410", "password": "12f@14Y$"}'
```

#### response

```
{  
  "hash": "fa7bf873fab9333144e171372a321b06",  
  "success": true,  
  "permissions": {  
    "base": [  
      "get_dealer_info"  
    ],  
    "service_settings": [  
      "read",  
      "update"  
    ],  
    "notification_settings": [  
      "read",  
      "update"  
    ]  
  }  
}
```



```

        "read",
        "update"
    ],
    "trackers": [
        "corrupt",
        "create",
        "delete",
        "global",
        "read",
        "report",
        "update"
    ],
    "users": [
        "corrupt",
        "create",
        "read",
        "update"
    ],
    "user_sessions": [
        "create"
    ],
    "tariffs": [
        "create",
        "read",
        "update"
    ],
    "transactions": [
        "create",
        "read"
    ],
    "activation_code": [
        "read",
        "update"
    ],
    "password": [
        "update"
    ],
    "email_gateways": [
        "create",
        "delete",
        "read",
        "send_email",
        "update"
    ],
    "subpaas": [
        "create",
        "delete",
        "read",
        "update"
    ]
}

```

- `hash` - string. A session key.

- `permissions` - object representing permissions for the panel. See panel account [permissions](#).

## errors

- 11 - Access denied - if dealer blocked.
- 12 - Dealer not found.

## get\_permissions

Returns permissions for current panel session.

## parameters

Only session `hash`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/account/
get_permissions' \
-H 'Content-Type: application/json' \
-d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/account/get_permissions?
hash=fa7bf873fab9333144e171372a321b06
```

## response

```
{
  "success": true,
  "permissions": {
    "base": [
      "get_dealer_info"
    ],
    "service_settings": [
      "read",
      "update"
    ],
    "notification_settings": [
      "read",
      "update"
    ],
    "trackers": [
      "corrupt",
      "create",
      "delete",
      "global",
      "read",
```

```

        "report",
        "update"
    ],
    "users": [
        "corrupt",
        "create",
        "read",
        "update"
    ],
    "user_sessions": [
        "create"
    ],
    "tariffs": [
        "create",
        "read",
        "update"
    ],
    "transactions": [
        "create",
        "read"
    ],
    "activation_code": [
        "read",
        "update"
    ],
    "password": [
        "update"
    ],
    "email_gateways": [
        "create",
        "delete",
        "read",
        "send_email",
        "update"
    ],
    "subpaas": [
        "create",
        "delete",
        "read",
        "update"
    ]
}

```

## errors

[General](#) types only.

## logout

Ends the current session.

## parameters

Only session hash.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/account/logout' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/account/logout?  
hash=fa7bf873fab9333144e171372a321b06
```

## response

```
{  
  "success": true  
}
```

## errors

[General](#) types only.

Last update: December 26, 2022





# Equipment

API call to get the list of all available equipment.

## Equipment object

```
{
  "equip_id": 33,
  "model_name": "SPT10 SB",
  "model_code": "pt10",
  "vendor": "3. NAVIXY S Series (personal)",
  "name": "NAVIXY S10"
}
```

- `equip_id` - int. Equipment ID.
- `model_name` - string. A model's original name.
- `model_code` - string. A model code which should be inserted to tracker bundles.
- `vendor` - string. A vendor's name.
- `name` - string. A model's name used by a vendor.

## API actions

API path: `panel/equipment`.

### list

Returns list of all equipment which can be assigned to tracker bundles.

*required permissions:* `tracker_bundles: "read"`.

### parameters

Only session `hash`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/equipment/list' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/equipment/list?  
hash=fa7bf873fab9333144e171372a321b06
```

## response

```
{  
  "success": true,  
  "list" : [{  
    "equip_id": 33,  
    "model_name": "SPT10 SB",  
    "model_code": "pt10",  
    "vendor": "3. NAVIXY S Series (personal)",  
    "name": "NAVIXY S10"  
  }]  
}
```

## errors

[General](#) types only.

Last update: July 19, 2022







# Gateways

Information about email gateway objects. Email gateway can be owned by a dealer or leased from platform owner.

## Email gateway object

Own email gateway: Now supported only SMTP provider.

```
{
  "id": 2,
  "leasable": false,
  "label": "Paas gate",
  "provider": "smtp",
  "params": {
    "default_from_address": "no-reply@domain.tld",
    "mail.smtp.user": null,
    "mail.smtp.password": null,
    "mail.smtp.host": "localhost",
    "mail.smtp.port": 25,
    "mail.smtp.ssl.port": 465,
    "mail.smtp.ssl.trust_all_hosts": false,
    "mail.smtp.auth": true,
    "mail.debug": false,
    "mail.smtp.starttls.enable": false,
    "mail.smtp.starttls.required": false,
    "mail.smtp.use_ssl": false,
    "mail.smtp.timeout": 60000,
    "mail.smtp.connectiontimeout": 60000,
    "mail.transport.protocol": "smtp"
  }
}
```

Leasable email gateway:

```
{
  "id": 1,
  "label": "Platform gate",
  "default_from_address": "no-reply@domain.tld"
}
```

## API actions

API path: `panel/gateways/email`.

## list

Gets list of available email gateways for the panel.

*required permissions:* email\_gateways: "read".

### parameters

Only session hash.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/gateways/email/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/panel/gateways/email/list?
hash=fa7bf873fab9333144e171372a321b06
```

### response

```
{
  "success": true,
  "bound_gateway": 2,
  "own": [{
    "id": 1,
    "leasable": false,
    "label": "Paas gate",
    "provider": "smtp",
    "params": {
      "default_from_address": "no-reply@domain.tld",
      "mail.smtp.user": null,
      "mail.smtp.password": null,
      "mail.smtp.host": "localhost",
      "mail.smtp.port": 25,
      "mail.smtp.ssl.port": 465,
      "mail.smtp.ssl.trust_all_hosts": false,
      "mail.smtp.auth": true,
      "mail.debug": false,
      "mail.smtp.starttls.enable": false,
      "mail.smtp.starttls.required": false,
      "mail.smtp.use_ssl": false,
      "mail.smtp.timeout": 60000,
      "mail.smtp.connectiontimeout": 60000,
      "mail.transport.protocol": "smtp"
    }
  }],
  "leasable": [
    {
      "id": 2,
```

```
        "label": "Default",  
        "provider": "mandrill_smtp",  
        "default_from_address": "no-reply@x-gpsmail.com"  
    }  
]  
}
```

## errors

[General](#) types only.

Last update: December 26, 2022





# Order

API call to read the order by its ID.

## API actions

API path: `panel/order`.

### read

Reads order by specified ID.

*required permissions:* `tracker_bundles: "read"`.

### parameters

name	description	type
order_id	Order ID.	int

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/order/read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "order_id":  
12341}'
```

#### HTTP GET

```
https://api.navixy.com/v2/panel/order/read?  
hash=fa7bf873fab9333144e171372a321b06&order_id=12341
```

### response

```
{  
  "success": true,  
  "value": {  
    "id": 3,  
    "user_id": 11346,  
    "seller_id": 1,  
    "amount": 1,  
    "sum": 34300.00,  
    "type": "equip",
```



```

    "payer": "Leonard Bernstein",
    "recipient": "Leonard Bernstein",
    "contacts": "",
    "place": "111111 Leipzig, Leipzig Tieckstrasse, 2",
    "comment": "",
    "creation_time": "2009-12-10 01:00:36",
    "status": "created",
    "bundles": [
      {
        "id": 2,
        "equip_id": 117,
        "equip_vendor": "Trackers of different manufacturers",
        "equip_name": "GPS/GSM terminal Teltonika FM1100",
        "equip_model": "FM1200",
        "model_code": "gv500",
        "imei": "355085050027285",
        "iccid": "89701010064407635201",
        "assign_time": "2014-12-15 13:42:54",
        "order_id": 3
      }
    ]
  }
}

```

#### errors

- 201 – Not found in the database - if specified order does not exist.

Last update: December 26, 2022





# Tariff

API calls for interaction with tariff plans.

## Tariff object

```
{
  "id": 12163,
  "name": "Premium",
  "group_id": 3,
  "active": true,
  "type": "monthly",
  "price": 12.55,
  "early_change_price": 23.0,
  "device_limit": 2000,
  "has_reports": true,
  "store_period": "1y",
  "device_type": "tracker",
  "proportional_charge": false,
  "service_prices": {
    "incoming_sms": 0.3,
    "outgoing_sms": 0.3,
    "service_sms": 0.2,
    "phone_call": 0.6,
    "traffic": 0.09
  }
}
```

- `id` - int. Tariff ID.
- `name` - string. Tariff name.
- `group_id` - int. Tariff group number.
- `active` - boolean. `true` if user allowed change his current tariff to this one.
- `type` - [enum](#). Type of tariff. Can be "monthly" or "activeday" (for "tracker" device\_type only).
- `price` - double. Tariff subscription price (usually per month).
- `early_change_price` - double. Price of change tariff from current to another. With the last change in less than 30 days ( `tariff.freeze.period` config option). When not passed or `null` user cannot change tariff frequently.
- `device_limit` - int. A maximum limit of devices per user. Not used for cameras and sockets.
- `has_reports` - boolean. If `true` the tariff has reports.

- `store_period` - string. Data storage period, e.g. "2h" (2 hours), "3d" (3 days), "5m" (5 months), "1y" (one year).
- `device_type` - [enum](#). Device type. Can be "tracker", "camera" or "socket".
- `proportional_charge` - boolean. `true` if monthly fee will be smaller when device was blocked during month (for "monthly" tariffs only).
- `service_prices` - JSON object with service prices.
  - `incoming_sms` - double. Incoming sms price.
  - `outgoing_sms` - double. Outgoing sms price.
  - `service_sms` - double. Service sms price.
  - `phone_call` - double. Phone voice notification sms price.
  - `traffic` - double. Traffic price per 1 MB.

## API actions

API path: `panel/tariff`.

### create

Creates a new tariff.

*required permissions:* `"tariffs": "create"`.

### parameters

name	description	type
tariff	Tariff object without ID field.	JSON object

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tariff/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "tariff":
{"name": "Premium", "group_id": 3, "active": true, "type":
"monthly", "price": 12.55, "early_change_price": 23.0,
"device_limit": 2000, "has_reports": true, "store_period": "1y",
"device_type": "tracker", "proportional_charge": false,
"service_prices": {"incoming_sms": 0.3, "outgoing_sms": 0.3,
"service_sms": 0.2, "phone_call": 0.6, "traffic": 0.09}}}'
```

## response

```
{
  "success": true,
  "id" : 123568
}
```

- `id` - int. An ID of the created tariff.

## errors

- 201 – Not found in the database - if specified tariff does not exist or belongs to different dealer.
- 214 – Requested operation or parameters are not supported by the device - when `device_type` does not support specified tariff `type`.
- 244 – Duplicate entity label - if there's another dealer's tariff with the same `name`.

## list

Returns list of all tariffs belonging to dealer.

If "filter" is used, entities will be returned only if filter string contains one of the following fields: `id`, `name`, `price`, `device_type`.

*required permissions:* `"tariffs": "read"`.

## parameters

name	description	type
<code>device_type</code>	Optional. Filter by device type. One of "tracker", "camera" or "socket".	enum
<code>filter</code>	Optional. Text filter.	string
<code>order_by</code>	Optional. List ordering. One of: <code>id</code> , <code>name</code> , <code>device_type</code> , <code>group_id</code> , <code>price</code> .	string
<code>ascending</code>	Optional. Default is <code>true</code> . If <code>true</code> , ordering will be ascending, descending otherwise.	boolean
<code>offset</code>	Optional. Default is <code>0</code> . Starting offset, used for pagination.	int

name	description	type
limit	Optional. Max number of records to return, used for pagination.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tariff/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tariff/list?
hash=fa7bf873fab9333144e171372a321b06
```

## response

```
{
  "success": true,
  "list" : [{
    "id": 12163,
    "name": "Premium",
    "group_id": 3,
    "active": true,
    "type": "monthly",
    "price": 12.55,
    "early_change_price": 23.0,
    "device_limit": 2000,
    "has_reports": true,
    "store_period": "1y",
    "device_type": "tracker",
    "proportional_charge": false,
    "service_prices": {
      "incoming_sms": 0.3,
      "outgoing_sms": 0.3,
      "service_sms": 0.2,
      "phone_call": 0.6,
      "traffic": 0.09
    }
  }],
  "wholesale_service_prices" : {
    "incoming_sms": 0.27,
    "outgoing_sms": 0.27,
    "service_sms": 0.17,
    "phone_call": 0.55,
    "traffic": 0.05
  },
}
```

```
    "count" : 42
}
```

- `list` - objects array. List of tariff plans. See tariff object [here](#).
- `wholesale_service_prices` - JSON object. Wholesale prices for all services (what dealer will pay per sms, per call, per mb).
- `count` - int. Total number of records (ignoring offset and limit).

## read

Returns tariff with specified ID.

*required permissions:* `"tariffs": "read"`.

### parameters

name	description	type
tariff_id	Tariff ID to read.	int

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tariff/read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "tariff_id":  
12163}'
```

#### HTTP GET

```
https://api.navixy.com/v2/panel/tariff/read?  
hash=fa7bf873fab9333144e171372a321b06&tariff_id=12163
```

### response

```
{  
  "success": true,  
  "value": {  
    "id": 12163,  
    "name": "Premium",  
    "group_id": 3,  
    "active": true,  
    "type": "monthly",  
    "price": 12.55,  
    "early_change_price": 23.0,  
    "device_limit": 2000,  
    "has_reports": true,  
  }  
}
```



```

    "store_period": "1y",
    "device_type": "tracker",
    "proportional_charge": false,
    "service_prices": {
      "incoming_sms": 0.3,
      "outgoing_sms": 0.3,
      "service_sms": 0.2,
      "phone_call": 0.6,
      "traffic": 0.09
    }
  }
}

```

- `value` - JSON object. See tariff object [here](#).

## errors

- 201 – Not found in the database - if specified tariff does not exist or belongs to different dealer.

## update

Updates existing tariff.

*required permissions:* `tariffs: "update"`.

## parameters

name	description	type
tariff	Tariff object without <code>device_type</code> field.	JSON object

## example

### cURL

```

curl -X POST 'https://api.navixy.com/v2/panel/tariff/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "tariff": {
    "id": 12345, "name": "Premium", "group_id": 3, "active": true,
    "type": "monthly", "price": 12.55, "early_change_price": 23.0,
    "device_limit": 2000, "has_reports": true, "store_period": "1y",
    "proportional_charge": false, "service_prices": {"incoming_sms":
    0.3, "outgoing_sms": 0.3, "service_sms": 0.2, "phone_call": 0.6,
    "traffic": 0.09}}}'

```

## response

```
{  
  "success": true  
}
```

### errors

- 201 – Not found in the database - if specified tariff does not exist or belongs to different dealer.
- 214 – Requested operation or parameters are not supported by the device when `device_type` does not support specified tariff `type`.
- 244 – Duplicate entity label - if there's another dealer's tariff with the same `name`.

## defaults object

```
{  
  "tariff_id": 1234,  
  "activation_bonus": 1.1,  
  "free_days": 14,  
  "free_days_device_limit": 3  
}
```

- `tariff_id` - int. An ID of the default tariff for this device type.
- `activation_bonus` - double. Activation bonus - money added to bonus balance upon device registration.
- `free_days` - int. Amount of free (without tariff fee) days after device registration.
- `free_days_device_limit` - int. A maximum number of activated user's devices with free period (null means no limit).

## defaults/read

Returns current tariff defaults for trackers and cameras.

*required permissions:* `tariffs: "read"`.

### parameters

Only session `hash`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tariff/defaults/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tariff/defaults/read?
hash=fa7bf873fab9333144e171372a321b06
```

## response

```
{
  "success": true,
  "tracker": {
    "tariff_id": 1234,
    "activation_bonus": 1.1,
    "free_days": 14,
    "free_days_device_limit": 3
  },
  "camera": {
    "tariff_id": 1289,
    "activation_bonus": 0.5,
    "free_days": 7,
    "free_days_device_limit": 3
  }
}
```

## errors

[General](#) types only.

## defaults/update

Updates current tariff defaults for trackers and cameras.

*required permissions:* tariffs: "update".

## parameters

name	description	type
tracker	Defaults object with ID field.	JSON object

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tariff/defaults/
update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "tracker":
{"tariff_id": 1234, "activation_bonus": 1.1, "free_days": 14,
"free_days_device_limit": 3}}'
```

## response

```
{
  "success": true
}
```

## errors

- 239 – New tariff doesn't exist - if tariff with specified ID does not exist.
- 237 – Invalid tariff - if new tariff has incompatible device type.

Last update: December 26, 2022





# Timezone

API call to get information about all supported timezones for the specified locale.

## API actions

API path: `panel/timezone`.

### list

Gets information about all supported timezones for the specified locale. Does not require authorization.

#### parameters

name	description	type
locale	Locale code to set language of descriptions.	<a href="#">enum</a>

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/timezone/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"locale": "en"}'
```

##### HTTP GET

```
https://api.navixy.com/v2/panel/timezone/list?locale=en
```

#### response

```
{  
  "success": true,  
  "list": [{  
    "zone_id": "America/Tijuana",  
    "description": "Tijuana",  
    "base_offset": -8.0,  
    "dst_offset": 1,  
    "country_code": "MX",  
    "alt_ids": [  
      "America/Ensenada",  
      "America/Santa_Isabel"    ]  
  }]
```

```
    ]  
  }]  
}
```

- `zone_id` - string. Timezone ID, which is used throughout the API.
- `description` - string. Localized description of the timezone.
- `description` - int. Base timezone offset in hours, e.g. 1 for London. May be negative.
- `description` - int. DST offset in hours. `0` if no DST rules for this timezone.
- `description` - string. ISO country code for the timezone.
- `alt_ids` - string array. List of string, optional, alternative timezone IDs.

## errors

[General](#) types only.

Last update: December 26, 2022







# Tracker

API calls to interact with trackers in the admin panel.

## Tracker object

```
{
  "id": 111231,
  "avatar_file_name" : "avatar",
  "clone": true,
  "comment": "Need to change SIM till next month",
  "creation_date": "2020-02-02",
  "group_id": 0,
  "dealer_id": 20410,
  "deleted": false,
  "label": "Truck",
  "user_id": 183654,
  "model_name": "Teltonika FMB120",
  "last_connection": "2020-02-02 12:44",
  "source": {
    "id": 456751,
    "device_id": "8624369656654",
    "model": "telfmb120",
    "blocked": false,
    "tariff_id": 13457,
    "creation_date": "2020-02-02",
    "tariff_end_date": "2021-02-02",
    "connection_status": "idle",
    "phone": "79995693344",
    "corrupted": true
  }
}
```

- `id` - int. Tracker ID aka `object_id`.
- `avatar_file_name` - optional string. Passed only if present.
- `clone` - boolean. `true` if this tracker is clone.
- `comment` - string. Comment (description) related to the tracker.
- `creation_date` - [date/time](#). Tracker or clone creation date.
- `group_id` - int. Tracker group ID. `0` if no group.
- `dealer_id` - int. An ID of a dealer to which this tracker (or clone) belongs to.
- `deleted` - boolean. True if tracker or clone has been marked as deleted.
- `label` - string. Tracker label.

- `user_id` - int. An ID of the user to which this tracker (or clone) belongs to.
- `model_name` - string. Human-readable tracker model name.
- `last_connection` - [date/time](#). Time when this tracker last connected to the server (in UTC+0 timezone).
- `source` - source JSON object.
  - `id` - int. Source ID.
  - `device_id` - string. Source\_imei.
  - `model` - string. Tracker model name from "models" table.
  - `blocked` - boolean. `true` if tracker has been blocked due to tariff end, etc.
  - `tariff_id` - int. An ID of tracker's tariff from "main\_tariffs" table.
  - `creation_date` - [date/time](#). Date when this tracker first registered in the system.
  - `tariff_end_date` - [date/time](#). Date of next tariff prolongation or null.
  - `connection_status` - [enum](#). Current connection status.
  - `phone` - string. Phone of the device. Can be null or empty if device has no GSM module or uses bundled SIM which number hidden from the user.
  - `corrupted` - boolean. `true` when tracker has been corrupted using /tracker/corrupt, and not passed when it is not corrupted.

## API actions

API path: `panel/tracker`.

### active/history/list

Provides information about trackers which were considered "active" by our PaaS billing system, on a month-by-month basis.

*required permissions:* `trackers: "read"`.

#### parameters

name	description	type
from	Start year and month for searching, e. g. "2021-02".	year-month string

name	description	type
to	End year and month for searching. e. g. "2021-03".	year-month string

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/active/history/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "from": "2021-02", "to": "2021-03"}'
```

## response

```
{
  "success": true,
  "list": [
    {
      "month": "2021-02",
      "amount": 1,
      "trackers": [
        {
          "tracker_id": 14,
          "user_id": 3,
          "label": "test",
          "device_id": "123321"
        }
      ]
    }
  ]
}
```

- `month` - string. Year and month of stats entry.
- `amount` - int. overall number of active trackers during a month.
- `trackers` - A basic info about active trackers

## errors

- 211 – Requested time span is too big.

## bundle/assign

Assign bundle to specified ICCID.

*required permissions:* `tracker_bundles: "update"`.

## parameters

name	description	type
bundle_id	ID of the bundle.	int
iccid	Must consist of printable characters and have length between 3 and 20.	string

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/bundle/assign' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "bundle_id": 1241, "iccid": "78974217758"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/bundle/assign?hash=fa7bf873fab9333144e171372a321b06&bundle_id=1241&iccid=78974217758
```

## response

```
{
  "success": true
}
```

## errors

- 201 – Not found in the database - if bundle not found.
- 208 – Device blocked - if SIM card blocked.
- 223 – Phone number already in use - if SIM card already in use.
- 226 – Wrong ICCID - if SIM card not found.
- 247 – Entity already exists - if ICCID is already exist.
- 250 – Not allowed for deleted devices - if SIM card deleted.

## bundle/order/assign

Assigns bundle to specified order ID.

*required permissions:* tracker\_bundles: "update".

## parameters

name	description	type
bundle_id	ID of a bundle.	int
order_id	ID of a bundle. Nullable.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/bundle/order/assign' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "bundle_id": 1241, "order_id": 78974217758}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/bundle/order/assign?hash=fa7bf873fab9333144e171372a321b06&bundle_id=1241&order_id=78974217
```

## response

```
{
  "success": true
}
```

## errors

- 201 – Not found in the database if bundle not found.

## bundle/import

Adds multiple bundles at once.

*required permissions:* tracker\_bundles: "create".

## parameters

name	description	type
imeis	Array of IMEI numbers.	string array

name	description	type
equip_id	ID of equipment to associate with all specified IMEIs.	int
factory_preset	Whether this device was preconfigured on factory or not.	boolean

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/bundle/import' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "imeis": ["896654523569742", "754854"], "equip_id": 13785, "factory_preset": false}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/bundle/import?hash=fa7bf873fab9333144e171372a321b06&bundle_id=1241&order_id=78974217
```

## response

```
{
  "success": true
}
```

## errors

- 201 – Not found in the database - if bundle not found.
- 247 – Entity already exists - if one of IMEIs is already exist.
- 204 – Entity not found - if there is no equipment with specified equip\_id.

## bundle/list

Gets list of all bundles. If `filter` is used, entities will be returned only if filter string contained within one of the following fields: `id`, `imei`, `model_code`, `iccid`, `assign_time`.

*required permissions:* `tracker_bundles: "read"`.



## parameters

name	description
clones_filter	Optional. Possible values: <code>exclude_clones</code> (filter out "cloned" trackers from results), <code>only_include_clones</code> (results shall contain only "cloned" trackers <code>not_set</code> ).
filter	string
Optional. Text filter string.	
order_by	Optional. Specify list ordering. Can be one of <code>id</code> , <code>label</code> , <code>status</code> , <code>model</code> , <code>device_id</code> , <code>phone</code> , <code>creation_date</code> , <code>user_id</code> , <code>comment</code> . Default order by <code>id</code> .
ascending	If <code>true</code> , ordering will be ascending, descending otherwise. Default is <code>true</code> .
offset	Optional. Starting offset, used for pagination. Default is <code>0</code> .
limit	Optional. Max number of records to return, used for pagination.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/bundle/list' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/bundle/list?
hash=fa7bf873fab9333144e171372a321b06
```

## response

```
{
  "success": true,
  "list" : [<bundle>],
  "count" : 42
}
```

- `list` - array of bundle objects.
- `count` - int. Total number of records (ignoring offset and limit).

## errors

- 201 – Not found in the database - if `user_id` or `tariff_id` specified but was not found.

## bundle/read

Returns the bundle object with the specified imei.

*required permissions:* `tracker_bundles: "read"`.

## parameters

name	description	type
imei	Device's IMEI.	string

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/bundle/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "imei": "835664527777452"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/bundle/read?
hash=fa7bf873fab9333144e171372a321b06&imei=835664527777452
```

## response

```
{
  "success": true,
  "value" : <bundle>
}
```

## errors

- 201 – Not found in the database - if bundle not found.

## bundle/update

Assign specified equipment to bundle.

*required permissions:* `tracker_bundles: "update"`.

#### parameters

name	description	type
bundle_id	ID of the bundle.	int
equip_id	Valid equipment ID.	int

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/bundle/
update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "bundle_id":
13457, "equip_id": 35468}'
```

##### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/bundle/update?
hash=fa7bf873fab9333144e171372a321b06&bundle_id=13457&equip_id=35468
```

#### response

```
{
  "success": true
}
```

#### errors

- 201 – Not found in the database - if bundle not found.
- 204 – Entity not found - if there is no equipment with specified `equip_id`.

#### clone

Creates a clone of the existing non-clone tracker. The method allows cloning from and to a subpaas's user account that is in the admin account hierarchy. Cloning from a user of one subpaas to another user of another subpaas in the same hierarchy is also possible.

*required permissions:* `trackers: "create"`.

## parameters

name	description	type
tracker_id	ID of the tracker. Tracker must belong to authorized dealer.	int
label	User-defined label for clone, e.g. "Courier". Must consist of printable characters and have length between 1 and 60.	string
user_id	ID of the user who will become the owner of the clone.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/clone' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "tracker_id":  
134537, "user_id": 354468, "label": "Courier"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/clone?  
hash=fa7bf873fab9333144e171372a321b06&tracker_id=134537&user_id=354468
```

## response

```
{  
  "success": true,  
  "id": 3947  
}
```

- `id` - int. An ID of the created clone.

## errors

- 219 - Not allowed for clones of the device – when source tracker is clone itself.
- 201 - Not found in the database – when specified `tracker_id` not found.
- 246 - Invalid user ID – when user ID is same as source tracker's owner id, or it does not exist/belong to authorized dealer.
- 247 - Entity already exists – if destination user already has a clone of this tracker.
- 252 - Device already corrupted – when tracker's source corrupted.

## console/connect

Returns auth token for connection to tracker command console.

*required permissions:* trackers: "update".

### parameters

name	description	type
tracker_id	ID of a tracker. Tracker must belong to authorized dealer.	int

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/console/
connect' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "tracker_id":
134537}'
```

#### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/console/connect?
hash=fa7bf873fab9333144e171372a321b06&tracker_id=134537
```

### response

```
{
  "success": true,
  "key": "6ad7490d4ec7f600ef10d4db41584980cd3ac230",
  "timestamp": 1399957326
}
```

- `key` - string. A key which is used to connect to console.
- `timestamp` - int. A timestamp which is used to connect to console.

Establish WS connection with a URL:

```
wss://ws.navixy.com/console?
device=<device_id>&key=<key>&timestamp=<timestamp>&dealer_id=<dealer_id>
```

JSON objects come in the next text frames:

```
{
  "data":
  [
    ["Time", "2020-06-09 10:02:37.0"],
    ["Location valid", "yes"],
```

```

    ["Latitude", "-33.4595716"],
    ["Longitude", "-70.7805233"],
    ["Speed", "0"],
    ["Heading", "229"],
    ["Moving", "false"],
    ["Satellites", "7"],
    ["Hardware mileage", "3707.85"],
    ["Mileage", "3853.16"],
    ["Digital input status", "8"],
    ["Analog input 1", "0.004"],
    ["Analog input 2", "0.02"],
    ["Digital output status", "3"],
    ["board_voltage", "11.619"],
    ["temp_sensor", "23.0"],
    ["GSM Level", "13"],
    ["GSM Operator code", "73002"],
    ["Battery level", "3.827"]
  ],
  "type": "status"
}

```

## errors

- 230 - Not supported for this entity type – when tracker deleted or blocked.
- 201 - Not found in the database – when tracker with such `device_id` not found.
- 252 - Device already corrupted – when tracker's source corrupted.

## corrupt

Mark tracker as deleted and corrupt its source `device_id` and `phone`. Rename tracking table.

*required permissions:* `trackers: "corrupt"`.

## parameters

name	description	type
tracker_id	ID of a tracker. Tracker must belong to authorized dealer.	int
corrupt_clones	Optional. Default is <code>true</code> . Remove clones of the tracker for other users	boolean

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/corrupt' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "tracker_id": 134537}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/corrupt?
hash=fa7bf873fab9333144e171372a321b06&tracker_id=134537
```

## response

```
{
  "success": true
}
```

## errors

- 201 – Not found in the database - if tracker not found.
- 219 – Not allowed for clones of the device - if source tracker is clone itself.
- 252 – Device already corrupted.
- 253 – Device has clones and `corrupt_clones` is false.

```
{
  "success": false,
  "status": {
    "code": 253,
    "description": "Device has clones"
  },
  "list": [234651]
}
```

- `list` - int array. Clones tracker\_ids list.

## batch\_clone

Creates clones from the specified set of existing non-clone trackers. The following actions are allowed within the same admin account hierarchy:

- Cloning from and to a subpaas's user account
- Cloning from a user of one subpaas to another user of another subpaas
- Cloning in a single operation from users that belong to different subpaas accounts

The maximum number of trackers to clone per operation is 1000. Labels from the original trackers are preserved.

*required permissions:* trackers: "create".

name	description	type
tracker_ids	Tracker ID list. Each of these trackers must not be a clone and must be accessible to the target user.	int array
user_id	Target user ID that is accessible from the admin panel hierarchy.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/batch_clone' \
-H 'Content-Type: application/json' \
-d '{"hash": "fa7bf873fab9333144e171372a321b06", "user_id": 998836, "tracker_ids": [134537, 458412]}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/batch_clone?
hash=fa7bf873fab9333144e171372a321b06&user_id=998836&tracker_ids=[1345
```

## response

name	description	type
list	Resulting list of created clone tracker IDs.	array of integers
success	Action's execution status.	boolean

Example:

```
{
  "list": [
    587469,
    587470
  ],
  "success": true
}
```



## errors

The operation is applied transactionally: it completes only if `"success": true` is received for the whole batch, otherwise, the cloning process for all trackers is rolled back.

- [Standard errors](#).
- 7 - Invalid parameters. Size must be between 1 and 1000 - triggered when the clone request exceeds 1000 trackers.
- 217 – List contains nonexistent entities - if at least one tracker from the request is not found.
- 247 – Entity already exists - if at least one of the trackers already has its clone in the target user. The error provides the list of trackers in the target user that caused the error.

Example:

```
{
  "status": {
    "code": 247,
    "description": "Entity already exists"
  },
  "list": [
    10191656,
    10191657
  ],
  "success": false
}
```

## batch\_delete\_clones

Deletes the specified set of trackers that are clones of other trackers. The action will be considered as completed successfully, even if some trackers could not be deleted. Then for the rest response will contain a description of the reasons why the deletion failed.

*required permissions:* `trackers: "delete"`.

name	description	type
trackers	Tracker ID list. Each of these trackers must be a clone and be accessible for current user.	int array

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/  
batch_delete_clones' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "trackers":  
[134537, 458412]}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/batch_delete_clones?  
hash=fa7bf873fab9333144e171372a321b06&trackers=[134537, 458412]
```

## response

name	description	type
success	Action's execution status.	boolean
deleted_count	Number of successfully deleted clones from trackers.	int
not_deleted_count	Number of not deleted clones.	int
not_deleted_trackers	Optional. Description of failed deletion operations. {"id": integer, "error": string}.	array of objects

## Example:

```
{  
  "success": true,  
  "deleted_count": 2,  
  "not_deleted_count": 3,  
  "not_deleted_trackers": [  
    {  
      "id": 2,  
      "error": "Not a clone"  
    },  
    {  
      "id": 3,  
      "error": "Entity not found"  
    },  
    {  
      "id": 4,  
      "error": "Already deleted"  
    }  
  ]  
}
```

```
]
}
```

## errors

- [Standard errors](#).

## delete\_clone

Deletes a clone of the existing tracker.

*required permissions:* trackers: "delete".

## parameters

name	description	type
tracker_id	ID of a tracker. Tracker must belong to authorized dealer and must be a clone.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/delete_clone' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "tracker_id": 134537}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/delete_clone?
hash=fa7bf873fab9333144e171372a321b06&tracker_id=134537
```

## response

```
{
  "success": true
}
```

## errors

- 201 - Not found in the database – if tracker not found.
- 249 - Operation available for clones only – if source tracker is not a clone.
- 203 - Delete entity associated with – if there are some rules or vehicles associated with tracker.

```
{
  "success": false,
  "status": {
    "code": 203,
    "description": "Delete entity associated with"
  },
  "rules": [10]
}
```

- `rules` - int array. A list of associated rule IDs.

or

```
{
  "success": false,
  "status": {
    "code": 203,
    "description": "Delete entity associated with"
  },
  "vehicles": [11]
}
```

- `vehicles` - int array. A list of associated vehicle IDs.
- 252 - Device already corrupted – when tracker's source corrupted.

## list

Returns list of all trackers belonging to dealer (with optional filtering by `filter` string, `user_id` and/or `tariff_id`).

If `filter` is used, entities will be returned only if filter string contain one of the following fields: `id`, `label`, `source.id`, `source.device_id`, `source.model`, `source.phone`, `user_id`.

*required permissions:* `trackers: "read"`.

### parameters

name	description	type
<code>user_id</code>	Optional. ID of the user. User must belong to authorized dealer.	int
<code>tariff_id</code>	Optional. ID of the tariff. Tariff must belong to authorized dealer.	int

name	description	type
filter	Optional. Text filter string.	string
order_by	Optional. List ordering. Can be one of "id", "label", "status", "model", "device_id", "phone", "creation_date", "last_connection".	string
ascending	Optional. If <code>true</code> , ordering will be ascending, descending otherwise. Default is <code>true</code> .	boolean
offset	Optional. Starting offset, used for pagination. Default is <code>0</code> .	int
limit	Optional. Max number of records to return, used for pagination.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/list' \
-H 'Content-Type: application/json' \
-d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/list?
hash=fa7bf873fab9333144e171372a321b06
```

## response

```
{
  "success": true,
  "list" : [{
    "id": 111231,
    "avatar_file_name" : "avatar",
    "clone": true,
    "comment": "Need to change SIM till next month",
    "creation_date": "2020-02-02",
    "group_id": 0,
    "dealer_id": 20410,
    "deleted": false,
    "label": "Truck",
    "user_id": 183654,
    "model_name": "Teltonika FMB120",
    "last_connection": "2020-02-02 12:44",
    "source": {
      "id": 456751,
```

```

        "device_id": "8624369656654",
        "model": "telfmb120",
        "blocked": false,
        "tariff_id": 13457,
        "creation_date": "2020-02-02",
        "tariff_end_date": "2021-02-02",
        "connection_status": "idle",
        "phone": "79995693344",
        "corrupted": true
    }
}],
    "count" : 42
}

```

- `list` - array of objects. Tracker object described [above](#).
- `count` - int. Total number of records ignoring `offset` and `limit`.

### errors

- 201 – Not found in the database - if specified `user_id` or `tariff_id` not found.

### move

Moves the existing non-clone tracker to another user belonging to the same dealer.

**Tracker will be unbound from any rules associated with it.**

*required permissions:* `trackers: "create", "delete"`.

### parameters

name	description	type
tracker_id	ID of the tracker. Tracker must belong to authorized dealer.	int
user_id	ID of the user who will become the owner of the tracker.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/move' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "tracker_id":  
1245678, "user_id": 214034}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/move?  
hash=fa7bf873fab9333144e171372a321b06&tracker_id=1245678&user_id=21403
```

## response

```
{  
  "success": true  
}
```

## errors

- 219 - Not allowed for clones of the device – when source tracker is clone.
- 201 - Not found in the database – when tracker not found.
- 246 - Invalid user ID – when `user_id` is the same as source tracker's owner id, or it does not exist/belong to authorized dealer.
- 247 - Entity already exists – when destination user already has a clone of this tracker.
- 252 - Device already corrupted – when tracker's source corrupted.

## read

Returns the tracker object with the specified ID.

*required permissions:* `trackers: "read"`.

## parameters

name	description	type
tracker_id	ID of the tracker. Tracker must belong to authorized dealer.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/read' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "tracker_id":  
1245678}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/read?  
hash=fa7bf873fab9333144e171372a321b06&tracker_id=1245678
```

## response

```
{  
  "success": true,  
  "value" : {  
    "id": 111231,  
    "avatar_file_name" : "avatar",  
    "clone": true,  
    "comment": "Need to change SIM till next month",  
    "creation_date": "2020-02-02",  
    "group_id": 0,  
    "dealer_id": 20410,  
    "deleted": false,  
    "label": "Truck",  
    "user_id": 183654,  
    "model_name": "Teltonika FMB120",  
    "last_connection": "2020-02-02 12:44",  
    "source": {  
      "id": 456751,  
      "device_id": "8624369656654",  
      "model": "telfmb120",  
      "blocked": false,  
      "tariff_id": 13457,  
      "creation_date": "2020-02-02",  
      "tariff_end_date": "2021-02-02",  
      "connection_status": "idle",  
      "phone": "79995693344",  
      "corrupted": true  
    }  
  }  
}
```

- `value` - JSON object. Tracker object described [above](#).

## errors

- 201 - Not found in the database – when tracker not found.
- 252 - Device already corrupted – when tracker's source corrupted.



## register\_retry

Sends tracker registration commands and resets all tracking settings. Can be executed once in 120 seconds for every tracker.

Device models `navixymobile*`, `mobile_unknown*`, `iosnavixytracker*` are not supported.

*required permissions:* `trackers: "update"`.

### parameters

name	description	type
tracker_id	ID of the tracker. Tracker must belong to authorized dealer.	int
send_register_commands	Indicates send or not to send activation commands to device (via SMS or GPRS channel). If parameter is not specified or equals <code>null</code> will be used the platform settings. Default: <code>null</code> .	boolean

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/register_retry' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "tracker_id": 1245678, "send_register_commands": true}'
```

#### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/register_retry?
hash=fa7bf873fab9333144e171372a321b06&tracker_id=1245678&send_register
```

### response

```
{
  "success": true
}
```

### errors

- 201 - Not found in the database – when tracker not found.
- 252 - Device already corrupted – if tracker corrupted.

- 264 - Timeout not reached – if another register retry request for this tracker done in last 120 seconds.
- 208 - Device blocked – when tracker exists but was blocked due to tariff restrictions or some other reason.
- 219 - Not allowed for clones of the device – when specified tracker is a clone.
- 214 - Requested operation or parameters are not supported by the device – when device does not have GSM module.
- 252 - Device already corrupted – when tracker's source corrupted.

## settings/update

Updates tracker settings.

*required permissions:* `trackers: "update"`.

### parameters

name	description	type
tracker_id	ID of the tracker. Tracker must belong to authorized dealer.	int
label	User-defined label for this tracker, e.g. "Courier". Must consist of printable characters and have length between 1 and 60. Cannot contain <code>&lt;</code> and <code>&gt;</code> symbols.	string
deleted	If <code>true</code> , tracker will be marked as deleted and will not be shown in user's interface.	boolean
comment	Optional. A comment (description) related to the tracker. Up to 3000 symbols.	string

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/settings/update' \
-H 'Content-Type: application/json' \
-d '{"hash": "fa7bf873fab9333144e171372a321b06", "tracker_id": 1245678, "label": "Courier", "deleted": false}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/settings/update?hash=fa7bf873fab9333144e171372a321b06&tracker_id=1245678&label=Courier
```

## response

```
{
  "success": true
}
```

## errors

- 201 - Not found in the database – when tracker not found.
- 252 - Device already corrupted – when tracker's source corrupted.

## source/update

Updates source settings. Can block and unblock a device.

*required permissions:* trackers: "update".

## parameters

name	description	type
tracker_id	ID of the tracker. Tracker must belong to authorized dealer.	int
blocked	If <code>true</code> , tracker will be marked as blocked.	boolean

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/source/
update' \
-H 'Content-Type: application/json' \
-d '{"hash": "fa7bf873fab9333144e171372a321b06", "tracker_id":
1245678, "blocked": false}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/source/update?
hash=fa7bf873fab9333144e171372a321b06&tracker_id=1245678&blocked=false
```

## response

```
{
  "success": true
}
```

## errors

- 201 - Not found in the database – when tracker not found.
- 252 - Device already corrupted – when tracker's source corrupted.

## tariff/change

*required permissions:* [trackers: "update", "transactions": "create",  
"tariffs": "read"].

## parameters

name	description	type
tracker_id	ID of tracker. Tracker must belong to authorized dealer.	int
tariff_id	New tariff ID.	int
repay	Repay remainder of current tariff payment.	boolean
charge	Charge payment for new tariff. For monthly and everyday tariffs.	boolean

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/tracker/tariff/change' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "tracker_id": 1245678, "tariff_id": 15843, "repay": false, "charge": true}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/tracker/tariff/change?
hash=fa7bf873fab9333144e171372a321b06&tracker_id=1245678&tariff_id=158
```

## response

```
{
  "success": true
}
```

## errors

- 201 – Not found in the database.
- 219 – Not allowed for clones of the device.
- 221 - Device limit exceeded – when new tariff device limit is less than count of user's trackers.
- 237 – Invalid tariff - if there are no tariff with tracker.tariff\_id and belongs to dealer.
- 238 – Changing tariff not allowed.
- 239 – New tariff doesn't exist.
- 250 – Not allowed for deleted devices.
- 252 - Device already corrupted – when tracker's source corrupted.

## Conditions of the change

The current dealer can change tracker tariff from `t1` to `t2`, if:

1. Tracker:
  - is not removed.
  - belongs to the dealer's user.
  - is not a clone.
2. `t1.tariff_id != t2.tariff_id`, i.e. it is impossible to change to the same tariff.
3. `t1.dealer_id = t2.dealer_id = dealer.effectiveDealerId`, i.e. both tariffs belong to the current dealer.

4. `t2.device = tracker` , i.e. only tracker tariffs are available.
5. depending on `t2.doc_type` :
  - `doc_type=0` (for all) – without conditions.
  - `doc_type=1` (for physical persons) – `user.face=1` (physical person).
  - `doc_type=2` (for legal entities) – `user.face=2` (legal entity) or `user.face=3` (SP).
  - `doc_type=3` (paas) – without conditions.
6. `t2.device_limit >= count` of trackers in user's cabinet.

## Repayment

Repayment will be carried out if the following conditions met:

1. The "repay" flag set (repay).
2. A current tariff – monthly: `t1.type = monthly` .
3. Tariff – paid: `tariff.price > 0`.
4. That the current tariff didn't end ( `tariff_end != 1` )
5. The tariff expiration date defined: `tariff_end_date != 0` (for monthly tariffs it has to be carried out always).
6. The free period expired: `created_date + free_period <= current date` where `free_period` obtained from the hardcodes table or from `default_model_settings`.
7. There is still at least one paid day on a tariff: `remainder > 0` The rest of days on a tariff: `remainder = the number of whole days before the end of the current tariff`.

amount to be repaid = `ceil(tariff.price * remainder / amt)` , where `amt` – the number of days in the current month, a `ceil` – the operation of taking the integer part.

## Change

```

tariff_id = next_tariff = new tariff ID
tariff_change = current date
if tariff is active (tariff_end = false) then
    tariff_end = false
    last_charged_date = current date
    if new tariff is monthly and the flag "to charge" is not set
    (charge) then
        tariff_end_date = the first day of the next month from the
        current date
    else
        tariff_end_date = tomorrow date
else (tariff is not active: tariff_end = true)
    last_charged_date = yesterday date
    if new tariff is monthly then
        if the flag "to charge" is set (charge) then
            tariff_end_date = current date
            tariff_end = true

```

```

else
    tariff_end_date = the first day of the next month from
the current date
    tariff_end = false
if the new tariff is everyday then
    if the flag "to charge" is set(charge) then
        tariff_end_date = current date
        tariff_end = true
    else
        tariff_end_date = tomorrow date
        tariff_end = false
if the new tariff is activeday then
    tariff_end_date = 0
    tariff_end = false

```

All dates according to UTC time.

## raw\_command/send

Sends the GPRS command to the device, processing it in a protocol-dependent manner beforehand.

**required sub-user rights:** `tracker_update`.

### parameters

name	description	type
device_id	Fixed device ID, e.g. IMEI.	string
command	Text or hexadecimal representation of the command.	string
type	Optional. Default is <code>text</code> . Can be "text" or "hex".	string
reliable	Optional. default is <code>true</code> . If <code>false</code> the command doesn't need to be resent when the device is disconnected or if no acknowledgment is received.	boolean

### example

#### cURL

```

curl -X POST 'https://api.navixy.com/v2/panel/tracker/raw_command/
send' \
-H 'Content-Type: application/json' \
-d '{"hash": "fa7bf873fab9333144e171372a321b06", "device_id":
"889654248978", "command": "setparam 101:4"}'

```

## response

```
{
  "success": true
}
```

## errors

- 7 - Invalid parameters.
- 201 - Not found in the database – if there is no tracker with such device ID belonging to authorized user.
- 252 - Device already corrupted – if tracker's source corrupted.

## example response with an error:

```
{
  "success": false,
  "status": {
    "code": 7,
    "description": "Invalid parameters"
  },
  "errors": [
    {
      "parameter": "command",
      "error": "Non-hex string"
    }
  ]
}
```

Last update: March 11, 2024







# Dealer get info

API call to get information about a dealer.

## API actions

API path: `panel/dealer/`.

### get\_info

Gets information about dealer's tariff, balance, available features, etc.

*required permissions:* base: `"get_dealer_info"`.

### parameters

Only session `hash`.

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/dealer/get_info' \
-H 'Content-Type: application/json' \
-d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/panel/dealer/get_info?
hash=fa7bf873fab9333144e171372a321b06
```

### response

```
{
  "success": true,
  "id": 9000,
  "seller_id": 3,
  "parent_dealer_id": 5001,
  "contract_type": "PAAS",
  "tariff_id": 5,
  "tariff": {
    "id": 5,
    "name": "PaaS Tariff",
    "type": "monthly",
    "currency": "RUB",
    "license_price": null,
    "min_license_pay": null,
  }
}
```

```

        "vat": false,
        "trial": false,
        "premium_gis": true,
        "service_prices": {
            "incoming_sms": 2.0,
            "outgoing_sms": 0.95,
            "service_sms": 0.95,
            "phone_call": 15,
            "traffic": 1.5
        },
        "store_period": "P3Y"
    },
    "demo_tariff": false,
    "tracker_tariff_end_date": "2015-12-31",
    "store_period": "P6M",
    "demo_ends": null,
    "title": "Navixy Demo",
    "block_status": "NOT_BLOCKED",
    "legal_name": "Company",
    "active_amount": 99,
    "active_amount_own": 80,
    "active_amount_subpaas": 19,
    "active_limit": 100,
    "locale": "en_US",
    "domain": "demo.navixy.com",
    "favicon": "paas/5001/custom.ico",
    "logo": "paas/5001/logo.png",
    "enable_trackers": true,
    "enable_cameras": false,
    "paas_activation_date": "2015-03-01",
    "license_balance": 0.0,
    "seller_currency": "USD",
    "features": [
        "branding_web",
        "branding_mobile",
        "navixy_label",
        "tracking",
        "reports",
        "fleet",
        "field_service",
        "premium_gis"
    ],
    "default_user_time_zone": "Europe/London"
}

```

- `id` - int. Dealer id.
- `parent_dealer_id` - int. An ID of parent dealer.
- `contract_type` - [enum](#). Contract type: "PARTNER", "AGENT" or "PAAS".
- `tariff_id` - int. PaaS tariff id.
- `tariff` - PaaS tariff info.
  - `license_price` - nullable double. Price per license.
  - `min_license_pay` - nullable double. Minimum license payment.

- `trial` - boolean. If `true` the plan is Trial.
- `premium_gis` - boolean. If `true` premium GIS enabled for the partner.
- `store_period` - string. Max data store period for users.
- `demo_tariff` - boolean. `true` for "TRIAL" PaaS tariffs.
- `store_period` - string. Max data store period for users on `demo_tariff`.
- `demo_ends` - string. TRIAL period end date or null.
- `block_status` - [enum](#). Panel and PaaS users block status. One of: "NOT\_BLOCKED", "INITIAL\_BLOCK", "BLOCK\_LOGIN" or "CLIENTS\_BLOCKED".
- `legal_name` - string. Dealer legal name.
- `active_amount` - int. Amount of all active trackers (with Sub-PaaS).
- `active_amount_own` - int. Amount of active trackers (without Sub-PaaS).
- `active_amount_subpaas` - int. Amount of Sub-PaaS' active trackers.
- `active_limit` - int. Active trackers limit.
- `locale` - [enum](#). Dealer's default locale.
- `domain` - string. Dealer's domain.
- `favicon` - string. Path or URL to dealer's interface favicon or null.
- `logo` - string. Path or URL to dealer's logotype or null.
- `paas_activation_date` - string. Date of activation pay.
- `features` - string array. Set of the allowed [dealer features](#).
- `default_user_time_zone` - string. [Time zone id](#) for new users to be created via [user/upload](#). Also, this zone will be selected by default when creating a new user in the Navixy Admin Panel.

## errors

- 201 - Not found in the database.

Last update: November 15, 2023





# Activation code

API calls for interacting with activation codes used for device registration.

## Activation code object

```
{
  "tariff_id": 12163,
  "bonus_amount": 0,
  "free_days": 14,
  "money_amount": 0,
  "device_type": "tracker",
  "code": "5248654776",
  "activated": false,
  "activation_date": null,
  "device_id": 0,
  "tariff_name": "Tracker demo tariff"
}
```

- `tariff_id` - int. A tariff ID.
- `bonus_amount` - int. Bonus that will be added to a user's balance when the device with this code activates.
- `free_days` - int. Count of free days.
- `money_amount` - int. Money that will be added to a user's balance.
- `code` - string. A code value.
- `activated` - boolean. If `true` it is activated.
- `device_id` - int. A device ID which activated with this code. It will be `0` if code not activated yet.
- `tariff_name` - string. Tariff name.

## API actions

API base path: `panel/dealer/activation_code`.

### create

Creates specified `count` of activation codes with passed `tariff_id`, `bonus_amount` and `free_days`. Returns count of created codes.



*required permissions:* `activation_code: ["read", "create"]`.

#### parameters

name	description	type
count	A count of codes to creation.	int
tariff_id	An ID of new tariff (must belong to current dealer).	int
bonus_amount	A new bonus amount.	int
free_days	A new free period.	int

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/dealer/activation_code/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "count": 10, "tariff_id": 12457, "bonus_amount": 3, "free_days": 5}'
```

##### HTTP GET

```
https://api.navixy.com/v2/panel/dealer/activation_code/create?hash=fa7bf873fab9333144e171372a321b06&count=10&tariff_id=12457&bonus_a
```

#### response

```
{
  "success": true,
  "count": 10
}
```

- `count` - int. Count of actually created codes.

#### errors

- 201 - Not found in the database - when tariff with `tariff_id` not found for a current dealer.

## list

Lists all dealer activation codes. If `filter` is used, entities will be returned only if filter string will contain one of the following fields: `code`, `tariff_id`, `device_id`, `device_type`.

*required permissions:* `activation_code: "read"`.

### parameters

name	description	type
filter	Optional. Text filter string.	string
order_by	Optional. Specify list ordering. Can be one of "code", "activated", "tariff_id", "tariff_name", "device_type", "money_amount", "bonus_amount", "free_days".	string
ascending	Optional. If <code>true</code> , ordering will be ascending, descending otherwise. Default is <code>true</code> .	boolean
offset	Optional. Starting offset, used for pagination. Default is <code>0</code> .	int
limit	Optional. Max number of records to return, used for pagination.	int

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/dealer/activation_code/list' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/panel/dealer/activation_code/list?  
hash=fa7bf873fab9333144e171372a321b06
```

### response

```
{  
  "success": true,  
  "list": [{  
    "tariff_id": 12163,  
    "bonus_amount": 0,  
  }]
```

```

        "free_days": 14,
        "money_amount": 0,
        "device_type": "tracker",
        "code": "1201245293",
        "activated": true,
        "activation_date": "2020-11-12 20:00:08",
        "device_id": 464606,
        "tariff_name": "Tracker demo tariff"
    }],
    "count" : 1
}

```

- `list` - array of objects. List of [activation code objects](#).
- `count` - int. Total number of records (ignoring offset and limit).

## update

Changes `tariff_id`, `bonus_amount` and `free_days` for all activation codes which: \* has `code` listed in `codes` parameter. \* belongs to a current dealer. \* not activated yet. \* belongs to the same `device_type` as a new tariff and return count of updated codes.

*required permissions:* `activation_code: "update"`.

## parameters

name	description	type
<code>codes</code>	Codes to update.	string array
<code>tariff_id</code>	An ID of a new tariff. Have to belong to a current dealer.	int
<code>bonus_amount</code>	A new bonus.	int
<code>free_days</code>	A new free period.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/dealer/activation_code/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "codes": ["12315124", "12451576"], "tariff_id": 12457, "bonus_amount": 3, "free_days": 5}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/dealer/activation_code/update?hash=fa7bf873fab9333144e171372a321b06&codes=["12315124", "12451576"]&tariff_id=12457&bonus_amount=3&free_days=5
```

## response

```
{
  "success": true,
  "count": 5
}
```

- `count` - int. Count of actually updated codes.

## errors

- 201 - Not found in the database - when a tariff with `tariff_id` not found for a current dealer.

Last update: December 26, 2022





# Password

API call to update dealer's password.

## API actions

API base path: `panel/dealer/password`.

### update

Changes password for the authorized dealer.

*required permissions:* `password: "update"`.

#### parameters

name	description	type
old_password	Current dealer's password.	string
new_password	New password for the dealer, 6 to 20 printable characters.	string

#### example

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/dealer/password/
update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06",
"old_password": "qwerty", "new_password": "B1r7d@Y"}'
```

#### response

```
{
  "success": true
}
```

#### errors

- 245 - New password must be different - if `old_password` = `new_password`.

- 248 - Wrong password - if `old_password` is wrong.

Last update: December 26, 2022







# Image

API calls for interaction with images that used for branding of the panel.

## API actions

API path: `panel/dealer/settings/image`.

### delete

Deletes an image of specified `type`.

*required permissions:* `service_settings: "update"`.

### parameters

name	description	typ
type	Image type to delete. Can be one of <code>logo</code> , <code>favicon</code> , <code>login_wallpaper</code> , <code>desktop_wallpaper</code> , <code>document_logo</code> .	stri

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/dealer/settings/image/delete' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "type": "logo"}'
```

#### HTTP GET

```
https://api.navixy.com/v2/panel/dealer/settings/image/delete?hash=fa7bf873fab9333144e171372a321b06&type=logo
```

### response

```
{
  "success": true
}
```

## errors

- 201 - Not found in the database - when there are no settings for a dealer in the db.

## upload

Uploads image of specified `type`.

**MUST** be a POST multipart request (multipart/form-data), with one of the parts being an image file upload (with the name "file").

File part **mime** type must be one of:

- `image/jpeg`
- `image/pjpeg`
- `image/png`
- `image/gif`
- `image/webp`
- `image/x-icon` (for favicon type)

*required permissions:* `service_settings: "update"`.

## parameters

name	description
type	Image type to delete. Can be one of <code>logo</code> , <code>favicon</code> , <code>login_wallpaper</code> , <code>desktop_wallpaper</code> , <code>document_logo</code> .
file	Image file.
redirect_target	Optional. A URL to redirect.

If `redirect_target` passed a return redirect to `response=<urlencoded response json>`.

## response

```
{
  "success": true
}
```

## errors

- 201 - Not found in the database - when there are no settings for dealer in the db.
- 233 - No data file - if `file` part not passed.
- 234 - Invalid data format - if passed `file` with unexpected `mime` type.
- 236 - Feature unavailable due to tariff restrictions - if branding feature disabled for this dealer.
- 254 - Cannot save file - on some file system errors.

Last update: January 22, 2023





# Notification

API calls to read and update notification settings.

## Notification settings object

```
{
  "email_from": "NAVIXY <no-reply@navixy.com>",
  "email_footer": "\n\n--nSincerely, Navixy",
  "email_special": "no-reply@navixy.com",
  "sms_originator": "demo.navixy.com",
  "caller_id": "491761234543"
}
```

- `email_from` - string. Email from which notification messages will be sent. Can be email address ("[no-reply@navixy.com](mailto:no-reply@navixy.com)") or email with a name ("NAVIXY [no-reply@navixy.com](mailto:no-reply@navixy.com)").
- `email_footer` - string. Footer which is added to all notification emails. Arbitrary text up to 600 characters.
- `email_special` - string. Special email address for PaaS reports.
- `sms_originator` - string. Max length is 20, must match `(p{L}|d|[-' " ., :/])*`. E.g. "demo.navixy.com" or "491761234567".
- `caller_id` - string. Voice messages originator. Max length is 20, must match `(p{L}|d|[-' " ., :/])*`. E.g. "491761234543".

## API actions

API path: `panel/dealer/settings/notification`.

### read

Gets current monitoring notification settings.

*required permissions:* `notification_settings: "read"`.

### parameters

Only session `hash`.



## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/dealer/settings/
notification/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/dealer/settings/notification/read?
hash=fa7bf873fab9333144e171372a321b06
```

## response

```
{
  "success": true,
  "value": {
    "email_from": "NAVIXY <no-reply@navixy.com>",
    "email_footer": "\n\n---nSincerely, Navixy",
    "email_special": "no-reply@navixy.com",
    "sms_originator": "demo.navixy.com",
    "caller_id": "491761234543"
  }
}
```

- `value` - [Notification settings object](#) described above.

## errors

[General](#) types only.

## update

Updates notification settings for the current dealer.

*required permissions:* `notification_settings: "update"`.

## parameters

name	description	type
email_from	Email from which notification messages will be sent. Can be email address or email with a name.	string
email_footer	Footer which is added to all notification emails. Arbitrary text up to 600 characters.	string

name	description	type
email_special	Optional. Special email address for PaaS reports.	string
sms_originator	SMS originator. Max length is 20.	string
caller_id	Voice messages originator. Max length is 20.	string

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/dealer/settings/
notification/update' \
-H 'Content-Type: application/json' \
-d '{"hash": "fa7bf873fab9333144e171372a321b06", "email_from":
"NAVIXY <no-reply@navixy.com>", "email_footer": "\n\n---
nSincerely, Navixy", "sms_originator": "demo.navixy.com",
"caller_id": "491761234543"}'
```

## response

```
{
  "success": true
}
```

## errors

[General](#) types only.

Last update: December 26, 2022





# Service

API calls to read and update panel's service settings.

## Service settings object

```
{
  "service_title": "monitoring service",
  "locale": "en_US",
  "demo_login": "demo",
  "demo_password": "demo",
  "maps": ["osm", "wikimapia", "yandexpublic", "osmmapnik"],
  "default_map": {
    "type": "osm",
    "location": {
      "lat": 33.0,
      "lng": 22.0
    },
  },
  "zoom": 2,
},
"currency": "EUR",
"payment_link": "http://payme.ru",
"promo_url": "http://monitoring.com/about",
"google_client_id": "io54p54ijy54",
"domain": "track.agent.com",
"favicon": "http://test.com/favicon.ico",
"app_logo": "paas/5001/app_logo.png",
"logo": "paas/5001/logo.png",
"document_logo": "paas/5001/document_logo.png",
"login_wallpaper": "paas/5001/login.png",
"desktop_wallpaper": "http://test.com/test.jpg",
"login_footer": "All rights reserved.",
"allow_registration": true,
"show_mobile_apps": true,
"default_user_settings": {
  "geocoder": "google",
  "route_provider": "progorod",
  "measurement_system": "metric",
  "translit": false
},
"display_model_features_link": false,
"limited_domain": false,
"allowed_maps": ["osm", "wikimapia", "yandexpublic",
"osmmapnik"],
"color_theme": "aqua",
"app_color_theme": "blue_1",
"privacy_policy_link": "http://privacy-policy-url",
"tos": "Terms Of Service text",
"no_register_commands": false,
```

```
"default_user_time_zone": "Europe/London"  
}
```

- `service_title` - string. Service name.
- `locale` - [enum](#). Default locale of the dealer.
- `demo_login` - string. If not empty, demo button will use this login to authorize.
- `demo_password` - string. If not empty, demo button will use this password to authorize.
- `maps` - [enum](#) array. Maps available in monitoring system. When a domain is platform owner's subdomain then only free maps are available.
- `default_map` - default map settings object.
  - `type` - [enum](#). Default map code.
  - `location` - location object. Default location to show on the map when monitoring opens. Location object described in [data types description section](#).
  - `zoom` - int. Default zoom level to use.
- `currency` - [enum](#). Code of the currency which can be shown in UI.
- `payment_link` - string. A link to dealer's payment system. Can be null or empty.
- `promo_url` - string. Customizable "About company" URL. Can be null or empty.
- `google_client_id` - string. Google Maps client ID (not supported by the interface yet).
- `domain` - string. Domain which will be used for monitoring system.
- `favicon` - string. Nullable, path or URL to dealer's interface favicon.
- `app_logo` - string. Nullable, path or URL to dealer's mobile app logotype.
- `logo` - string. Nullable, path or URL to dealer's logotype.
- `document_logo` - string. Nullable, path or URL to dealer's logotype for documents.
- `login_wallpaper` - string. Nullable, path or URL to dealer's interface login wallpaper.
- `desktop_wallpaper` - string. Nullable, path to dealer's interface wallpaper.
- `login_footer` - string. Nullable, footer which will be included in login page.
- `allow_registration` - boolean. If `true` allows self-registration of users.
- `show_mobile_apps` - boolean. If `true` when entering the service from a mobile device or tablet, users will be prompted to download the mobile application or continue using the mobile web UI.
- `default_user_settings` - default user settings object.
  - `geocoder` - string. Default geocoder.

- `route_provider` - string. Default route provider.
- `measurement_system` - [enum](#). Measurement system.
- `date_format` - Optional [enum](#). Date representation.
- `hour_mode` - Optional [enum](#). Time representation.
- `translit` - boolean. SMS transliteration. If `true` allows you to reduce the number of characters in an SMS message by replacing the characters of the national alphabet with close Latin ones.
- `display_model_features_link` - boolean. When `true` shows in model info link to [navixy.com](https://navixy.com) (UI option).
- `limited_domain` - boolean. If `true`, paas domain has limitations.
- `allowed_maps` - [enum](#). List of maps available for selection in "maps" list.
- `color_theme` - string. 128 chars max. Color theme code or empty string (for default theme).
- `app_color_theme` - string. 128 chars max. Mobile app color theme code or empty string (for default theme).
- `privacy_policy_link` - string. A link to privacy policy.
- `tos` - string. Terms Of Service text.
- `no_register_commands` - boolean. If `true` then do not send commands to devices on activation.
- `default_user_time_zone` - string. [Time zone id](#) for new users to be created via [user/upload](#). Also, this zone will be selected by default when creating a new user in the Navixy Admin Panel.

## API actions

API path: `panel/dealer/settings/service`.

### read

Gets current monitoring service settings.

*required permissions:* `service_settings: "read"`.

### parameters

Only session `hash`.

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/dealer/settings/service/read' \
-H 'Content-Type: application/json' \
-d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/dealer/settings/service/read?
hash=fa7bf873fab9333144e171372a321b06
```

## response

```
{
  "success": true,
  "value": {
    "service_title": "monitoring service",
    "locale": "en_US",
    "demo_login": "demo",
    "demo_password": "demo",
    "maps": ["osm", "wikimapia", "yandexpublic", "osmmapnik"],
    "default_map": {
      "type": "osm",
      "location": {
        "lat": 33.0,
        "lng": 22.0
      },
      "zoom": 2
    },
    "currency": "EUR",
    "payment_link": "http://payme.ru",
    "promo_url": "http://monitoring.com/about",
    "google_client_id": "io54p54ijy54",
    "domain": "track.agent.com",
    "favicon": "http://test.com/favicon.ico",
    "app_logo": "paas/5001/app_logo.png",
    "logo": "paas/5001/logo.png",
    "document_logo": "paas/5001/document_logo.png",
    "login_wallpaper": "paas/5001/login.png",
    "desktop_wallpaper": "http://test.com/test.jpg",
    "login_footer": "All rights reserved.",
    "allow_registration": true,
    "show_mobile_apps": true,
    "default_user_settings": {
      "geocoder": "google",
      "route_provider": "progorod",
      "measurement_system": "metric",
      "translit": false
    },
    "display_model_features_link": false,
    "limited_domain": false,
    "allowed_maps": ["osm", "wikimapia", "yandexpublic", "osmmapnik"],
    "color_theme": "aqua",
```



```

    "app_color_theme": "blue_1",
    "privacy_policy_link": "http://privacy-policy-url",
    "tos": "Terms Of Service text",
    "no_register_commands": false,
    "default_user_time_zone": "America/New_York"
  }
}

```

- `value` - [Service settings object](#) described above.

## errors

[General](#) types only.

## update

Updates monitoring service settings for the current dealer.

Note: wallpapers, logos and favicons cannot be edited here.

*required permissions:* `service_settings: "update"`.

## parameters

name	description	type
service_title	Service name.	string
locale	Default locale of the dealer.	<a href="#">enum</a>
demo_login	If not empty, demo button will use this login to authorize.	string
demo_password	If not empty, demo button will use this password to authorize.	string
maps	Maps available in monitoring system.	<a href="#">enum</a> array
default_map	Default map settings object.	JSON object
currency	Code of the currency which will be shown in UI.	<a href="#">enum</a>

name	description	type
payment_link	A link to dealer's payment system. Can be null or empty.	string
promo_url	Customizable "About company" URL. Can be null or empty.	string
google_client_id	Google maps client ID.	string
domain	Domain which will be used for monitoring system.	string
login_footer	Nullable, footer which will be included in login page.	string
allow_registration	If <code>true</code> allows self-registration of users.	boolean
show_mobile_apps	If <code>true</code> shows mobile apps to users who opens mobile web UI.	boolean
default_user_settings	Default user settings object.	JSON object
display_model_features_link	When <code>true</code> shows in model info link to navixy.com (UI option).	boolean
limited_domain	If <code>true</code> , paas domain has limitations.	boolean
allowed_maps	List of maps available for selection in "maps" list.	enum
color_theme	128 chars max. Color theme code or empty string (for default theme).	string
app_color_theme	128 chars max. Mobile app color theme code or empty string (for default theme).	string
privacy_policy_link	A link to privacy policy.	string

name	description	type
tos	Terms Of Service text.	string
no_register_commands	If <code>true</code> then do not send commands to devices on activation.	boolean
default_user_time_zone	Time zone by default for new users.	string

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/dealer/settings/
notification/update' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06",
"service_title": "monitoring service", "locale": "en_US",
"demo_login": "demo", "demo_password": "demo", "maps": ["osm",
"wikimapia", "yandexpublic", "osmmapnik"], "default_map": {"type":
"osm", "location": {"lat": 33.0, "lng": 22.0}, "zoom": 2},
"currency": "EUR", "payment_link": "http://payme.ru", "promo_url":
"http://monitoring.com/about", "google_client_id": "io54p54ijy54",
"domain": "track.agent.com", "login_footer": "All rights
reserved.", "allow_registration": true, "show_mobile_apps": true,
"default_user_settings": {"geocoder": "google", "route_provider":
"progorod", "measurement_system": "metric", "translit": false},
"display_model_features_link": false, "limited_domain": false,
"allowed_maps": ["osm", "wikimapia", "yandexpublic", "osmmapnik"],
"color_theme": "aqua", "app_color_theme": "blue_1",
"privacy_policy_link": "http://privacy-policy-url", "tos": "Terms
Of Service text", "no_register_commands": false,
"default_user_time_zone": "Europe/London"}'
```

## response

```
{
  "success": true
}
```

## errors

- 247 - Entity already exists(409) - when `domain` already used by other dealer.

Last update: April 11, 2024





# Subpaas actions

API calls to interact with Subpaases.

## Subpaas object

```
{
  "subpaas_id": 18,
  "title": "SubppaasTitle",
  "jur_name": "SubppaasJurName",
  "login": "subpaaslogin",
  "creation_date": "2018-11-15",
  "block_type": "NOT_BLOCKED",
  "users_count": 2,
  "active_users_count": 1,
  "trackers_count": 0,
  "active_trackers_count": 0,
  "contact_fio": "fio",
  "contact_post": "post"
}
```

- `subpaas_id` - int. Subpaas id.
- `title` - string. Subpaas' name.
- `jur_name` - string. Legal (juristic) company name.
- `creation_date` - string. Creation date.
- `block_type` - [enum](#). Panel and Subpaas' users block status. One of: "NOT\_BLOCKED", "INITIAL\_BLOCK", "BLOCK\_LOGIN" or "CLIENTS\_BLOCKED".
- `users_count` - int. Count of users.
- `active_users_count` - int. Count of active users.
- `trackers_count` - int. All devices of Subpaas.
- `active_trackers_count` - int. Active devices of Subpaas.
- `contact_fio` - string. Contact person.
- `contact_post` - string. Contact post (position).
- `contact_phone` - string. Contact's phone.

## API actions

API base path: `panel/subpaas`.

### create

Creates subpaas. After creation its `dealer_block_type` will be in `INITIAL_BLOCK` status.

#### parameters

name	description	type
password	Subpaas' password.	string
title	Subpaas' name.	string
email	Company email.	string
jur_name	Legal (juristic) company name.	string
jur_country	Subpaas' country	string
link_monitoring	Subpaas' domain name.	string

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/subpaas/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "password": "B1r7d@Y", "title": "Company", "email": "email@company.com", "jur_name": "Company", "jur_country": "Finland", "link_monitoring": "company.com"}'
```

### response

```
{
  "success": true
}
```

## errors

- 13 – If the dealer
  - is not paas.
  - has different status than `NOT_BLOCKED`.
  - his tariff doesn't allow subpaases.

## list

Gets a list of all subpaases for a dealer. Dealer ID will be taken from the session key.

## parameters

name	description	type
order_by	Optional. Sort option. Can be "subpaas_id", "title", "jur_name", "login", "block_type", "creation_date". Default is <code>subpaas_id</code> .	enum
ascending	Optional. If <code>true</code> ordering will be ascending, descending otherwise. Default is <code>true</code> .	boolean
limit	Optional. Pagination. Maximum subpaases to return	int
offset	Optional. Pagination. Get subpaases starting from.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/subpaas/list' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/subpaas/list?  
hash=fa7bf873fab9333144e171372a321b06
```

## response

```
{  
  "success": true,  
  "list": [  
    {  
      "id": 1,  
      "title": "Subpaas 1",  
      "jur_name": "Jurisdiction 1",  
      "login": "login1",  
      "block_type": "block_type1",  
      "creation_date": "2023-01-01",  
      "status": "NOT_BLOCKED",  
      "tariff": "tariff1",  
      "allow_subpaases": true  
    },  
    {  
      "id": 2,  
      "title": "Subpaas 2",  
      "jur_name": "Jurisdiction 2",  
      "login": "login2",  
      "block_type": "block_type2",  
      "creation_date": "2023-01-02",  
      "status": "NOT_BLOCKED",  
      "tariff": "tariff2",  
      "allow_subpaases": true  
    }  
  ]  
}
```



```

{
  "subpaas_id": 18,
  "title": "SubppaasTitle",
  "jur_name": "SubppaasJurName",
  "login": "subpaaslogin",
  "creation_date": "2018-11-15",
  "block_type": "NOT_BLOCKED",
  "users_count": 2,
  "active_users_count": 1,
  "trackers_count": 0,
  "active_trackers_count": 0
}
]
}

```

- `list` - array of objects. List of [subpaas objects](#) described above.

## errors

- 13 – If the dealer
  - is not paas.
  - has different status than `NOT_BLOCKED`.
  - his tariff doesn't allow subpaases.

## read

Gets subpaas info by its id.

## parameters

name	description	type
subpaas_id	Subpaas ID.	int

## examples

### cURL

```

curl -X POST 'https://api.navixy.com/v2/panel/subpaas/read' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "subpaas_id": 97834}'

```

### HTTP GET

```

https://api.navixy.com/v2/panel/subpaas/read?
hash=fa7bf873fab9333144e171372a321b06&subpaas_id=97834

```

## response

```
{
  "success": true,
  "value": {
    "subpaas_id": 18,
    "block_type": "NOT_BLOCKED",
    "title": "Rus Sub-PaaS",
    "jur_name": "000 Sub-PaaS",
    "email": "sub-dealer@email.com",
    "jur_country": "country",
    "link_monitoring": "link",
    "contact_fio": "fio",
    "contact_post": "post",
    "contact_phone": "phone"
  }
}
```

- `value` - [subpaas object](#) described above.

## errors

- 13 – If the dealer
  - is not paas.
  - has different status than `NOT_BLOCKED`.
  - his tariff doesn't allow subpaases.

## update

Updates a subpaas with specified ID.

## parameters

name	description	type
subpaas_id	Subpaas' ID.	int
password	Subpaas' password.	string
title	Subpaas' name.	string
email	Company email.	string
jur_name	Legal (juristic) company name.	string

name	description	type
jur_country	Subpaas' country	string
link_monitoring	Subpaas' domain name.	string
contact_fio	Contact person.	string
contact_post	Contact post (position).	string
contact_phone	Contact's phone.	string
block_type	Panel and PaaS users block status. One of: "NOT_BLOCKED", "INITIAL_BLOCK", "BLOCK_LOGIN" or "CLIENTS_BLOCKED".	enum

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/subpaas/create' \
-H 'Content-Type: application/json' \
-d '{"hash": "fa7bf873fab9333144e171372a321b06", "password": "B1r7d@Y", "title": "Company", "email": "email@company.com", "jur_name": "Company", "jur_country": "Finland", "link_monitoring": "company.com", "contact_fio": "fio", "contact_post": "CEO", "contact_phone": "79999902190", "block_type": "NOT_BLOCKED"}'
```

## response

```
{
  "success": true
}
```

## errors

- 13 –
  - The dealer is not paas.
  - The dealer has different status than `NOT_BLOCKED`.
  - Subpaases are not permitted for dealer.
  - `block_type` is `DELETED`.
  - Found subpaas is in `DELETED` status.

- Found subpaas is not in `INITIAL_BLOCK` status and `block_type` is `INITIAL_BLOCK`.
- Found subpaas is in `INITIAL_BLOCK` status and `block_type` is not `INITIAL_BLOCK`.

Last update: December 26, 2022





# Change password

API base path: `panel/subpaas/password`.

API call to change subpaas password.

## API actions

API base path: `panel/subpaas/password`.

### change

Changes subpaas password.

#### parameters

name	description	type
subpaas_id	Subpaas' ID.	int
new_password	New subpaas' password, 6 to 20 printable characters.	string

#### example

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/subpaas/password/
change' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "subpaas_id":
99874, "new_password": "Fr1d@Y$"}'
```

#### response

```
{
  "success": true
}
```

#### errors

- 13 –
  - The dealer is not paas.

- The dealer has different status than `NOT_BLOCKED`.
- The dealer's tariff does not allow subpaases.
- Found subpaas is in `DELETED` status.

Last update: December 26, 2022







# Subpaas session key

API call to create a subpaas session key.

## API actions

API base path: `panel/subpaas/session`.

### create

Creates a subpaas session.

#### parameters

name	description	type
subpaas_id	Subpaas' ID.	int

#### examples

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/subpaas/session/create' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "subpaas_id": 97834}'
```

##### HTTP GET

```
https://api.navixy.com/v2/panel/subpaas/session/create?  
hash=fa7bf873fab9333144e171372a321b06&subpaas_id=97834
```

#### response

```
{  
  "success": true,  
  "hash": "600d4a5400000000600d4a5400000000"  
}
```

## errors

• 13 –

- The dealer is not paas.
- The dealer has different status than `NOT_BLOCKED`.
- The dealer's tariff does not allow subpaases.
- Found subpaas is not in `NOT_BLOCKED` status.

Last update: December 26, 2022





# User

API calls on work with users in the admin panel.

## User object structure

```
{
  "dealer_id": 5001,
  "activated": true,
  "verified": true,
  "login": "user@test.com",
  "first_name": "John",
  "middle_name": "William",
  "last_name": "Smith",
  "legal_name": "E. Biasi GmbH",
  "legal_type": "legal_entity",
  "phone": "491761234567",
  "post_country": "Germany",
  "post_index": "61169",
  "post_region": "Hessen",
  "post_city": "Wiesbaden",
  "post_street_address": "Marienplatz 2",
  "registered_country": "Germany",
  "registered_index": "61169",
  "registered_region": "Hessen",
  "registered_city": "Wiesbaden",
  "registered_street_address": "Marienplatz 2",
  "state_reg_num": "12-3456789",
  "tin": "1131145180",
  "okpo_code": "93281776",
  "iec": "773101001",
  "id": 38935,
  "balance": 10.01,
  "bonus": 0,
  "creation_date": "2021-03-01 13:00:00",
  "trackers_count": 10,
  "comment": "about user"
}
```

- `dealer_id` - int. Dealer ID.
- `activated` - boolean. `true` if user activated (allowed to login).
- `verified` - boolean. `true` if user's email verified.
- `login` - string. User email as login. Must be valid unique email address.
- `first_name` - string. Contact person first name.
- `middle_name` - string. Contact person middle name.

- `last_name` - string. Contact person last name.
- `legal_name` - string. User legal name (for "legal\_entity" only).
- `legal_type` - [enum](#). Can be "legal\_entity", "individual" or "sole\_trader".
- `phone` - string. Contact phone 10-15 digits without "+" sign.
- `post_country` - string. Country part of user's post address.
- `post_index` - string. Index part of user's post address.
- `post_region` - string. Region part of user's post address.
- `post_city` - string. City from postal address.
- `post_street_address` - string. Street address.
- `registered_country` - string. Country part of user's registered address.
- `registered_index` - string. Index part of user's registered address.
- `registered_region` - string. Region part of user's registered address.
- `registered_city` - string. City from registered address.
- `registered_street_address` - string. User's registered address.
- `state_reg_num` - string. State registration number. E.g. EIN in the USA, OGRN in the Russia. 15 characters max.
- `tin` - string. Taxpayer identification number aka "VATIN".
- `okpo_code` - string, optional. All-Russian Classifier of Enterprises and Organizations, used in Russia for "legal\_entity" or "sole\_trader".
- `iec` - string, optional. Industrial Enterprises Classifier aka "KPP" (used in Russia. for "legal\_entity" only).
- `id` - int. User id. Next fields are read-only, they should not be used in `user/update` and `user/create`.
- `balance` - double. User balance.
- `bonus` - double. User bonus balance.
- `creation_date` - [date/time](#). Date and time when user created, in UTC.
- `trackers_count` - user trackers count.
- `comment` - comment about user (when creating and editing, the field must be separate from this object).

## Discount object structure

```
{
  "value": 5.5,
```



```
"min_trackers": 10,  
"end_date": "2021-03-01",  
"strategy": "sum_with_progressive"  
}
```

- `value` - double. Personal discount percent, min 0 max 100.
- `min_trackers` - int. Minimum active trackers to apply discount, min 0.
- `end_date` - [date/time](#). Discount end date, null means open date, nullable.
- `strategy` - [enum](#). One of "no\_summing", "sum\_with\_progressive".\

## API actions

API path: `panel/user`.

### change\_password

Changes password of a user.

*required permissions:* `users: "update"`.

#### parameters

name	description	type
<code>user_id</code>	ID of a user.	int
<code>password</code>	User's new password, 6 to 20 printable characters.	string

#### example

##### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/user/change_password'  
\br/>-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "user_id":  
231432, "password": "12@14Y$"}'
```

#### response

```
{  
  "success": true  
}
```

## errors

- 201 – Not found in the database - if specified user does not exist or belongs to different dealer.

## corrupt

Marks user and its sub users and trackers as deleted and corrupt all user trackers.

*required permissions:* `users: "corrupt"`.

## parameters

name	description	type
user_id	User id.	int
login	Login of a user. Login parameter must match user login.	string
corrupt_clones	Optional. Default is <code>true</code> . Remove clones of the user's trackers for other users	boolean

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/user/corrupt' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "user_id":  
231432, "login": "user@login.com"}'
```

## response

```
{  
  "success": true  
}
```

## errors

- 201 – Not found in the database - if a user not found.
- 252 – Device already corrupted - if some of user's tracker already corrupted.
- 253 – Device has clones - if some of user's tracker has a clone and `corrupt_clones` is false.

```
{
  "success": false,
  "status": {
    "code": 253,
    "description": "Device has clones"
  }
}
```

## create

Creates a new user.

*required permissions:* [users: "corrupt", "global"] .

- users: "global" - Optional. Allows creating users of users, not only owned by a current dealer (use `user.dealer_id` parameter for other owners).

## parameters

name	description	type
user	<a href="#">User object</a> without the <code>id</code> , <code>dealer_id</code> , <code>comment</code> and read-only fields.	JSON object
time_zone	User timezone.	string
locale	User locale.	string
password	User password, 6 to 20 printable characters.	string
discount	<a href="#">Discount object</a> .	JSON object
default_tariff_id	Optional. ID of a default tariff plan for user's trackers	int
<code>comment</code>	Comment	String, max length 255, only printable characters

If `user.verified` not passed then it set equal to `user.activated` .

## example

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/user/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "user":
{"activated": true, "verified": true, "login": "user@test.com",
"first_name": "John", "middle_name": "William", "last_name":
"Smith", "legal_name": "E. Biasi GmbH", "legal_type":
"legal_entity", "phone": "491761234567", "post_country":
"Germany", "post_index": "61169", "post_region": "Hessen",
"post_city": "Wiesbaden", "post_street_address": "Marienplatz 2",
"registered_country": "Germany", "registered_index": "61169",
"registered_region": "Hessen", "registered_city": "Wiesbaden",
"registered_street_address": "Marienplatz 2", "state_reg_num":
"12-3456789", "tin": "1131145180", "okpo_code": "93281776", "iec":
"773101001"}, "time_zone": "Europe/Moscow", "locale": "en-US",
"password": "12@14Y$", "discount": {"value": 5.5, "min_trackers":
10, "end_date": null, "strategy": "sum_with_progressive"},
"comment": "about user"}'
```

## response

```
{
  "success": true,
  "id" : 15534
}
```

- `id` - int. An ID of the created user.

## errors

- 206 - Login already in use – if this email already registered.

## export

Returns list of all users belonging to dealer as file.

If `filter` is used (parameter `filter` is passed, it isn't empty and does not consist only of space characters), entities will be returned only if filter string is contained within one of the following fields: `id`, `login`, `last_name`, `first_name`, `middle_name`, `phone`, `post_city`, `post_region`, `post_country`, `post_index`, `post_street_address`, `registered_country`, `registered_index`, `registered_region`, `registered_city`, `registered_street_address`, `tin`, `iec`, `legal_name`.

*required permissions:* `users: "read"`.

## parameters

name	description	type
filter	Optional. Text filter string.	string
order_by	Optional. Specify list ordering. May be one of: <code>id</code> , <code>login</code> , <code>last_name</code> , <code>balance</code> , <code>bonus</code> , <code>phone</code> , <code>post_city</code> . Default is <code>id</code> .	string
ascending	Optional. If <code>true</code> , ordering will be ascending, descending otherwise. Default is <code>true</code> .	boolean
limit	Optional. Max number of records to return, used for pagination.	int
offset	Optional. Starting offset, used for pagination. Default is <code>0</code> .	int
hide_inactive	Optional. If <code>true</code> only activated users will be returned. Default is <code>false</code> .	boolean
format	Optional. Format of exported list. Can be <code>xlsx</code> or <code>csv</code> . Default is <code>xlsx</code> .	string
columns	Optional. A list of columns to export. Default is <code>["id", "login", "first_name", "middle_name", "last_name", "phone"]</code> .	string array

About user object structure see [above](#).

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/user/export' \  
  -H 'Content-Type: application/json' \  
  -d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/user/export?  
hash=fa7bf873fab9333144e171372a321b06
```

## response

`XLSX` or `CSV` file download starts.

## errors

- [General](#) types only.

## list

Returns a list of all users belonging to dealer.

If `filter` is used (parameter `filter` is passed, it is not empty and does not consist only of space characters), entities will be returned only if filter string is contained within one of the following fields: `id`, `login`, `last_name`, `first_name`, `middle_name`, `phone`, `post_city`, `post_region`, `post_country`, `post_index`, `post_street_address`, `registered_country`, `registered_index`, `registered_region`, `registered_city`, `registered_street_address`, `tin`, `iec`, `legal_name`.

*required permissions:* `users: "read"`.

## parameters

name	description	type
<code>filter</code>	Optional. Text filter string.	string
<code>order_by</code>	Optional. Specify list ordering. May be one of: <code>id</code> , <code>login</code> , <code>last_name</code> , <code>balance</code> , <code>bonus</code> , <code>phone</code> , <code>post_city</code> . Default is <code>id</code> .	string
<code>ascending</code>	Optional. If <code>true</code> , ordering will be ascending, descending otherwise. Default is <code>true</code> .	boolean
<code>limit</code>	Optional. Max number of records to return, used for pagination.	int
<code>offset</code>	Optional. Starting offset, used for pagination. Default is <code>0</code> .	int
<code>hide_inactive</code>	Optional. If <code>true</code> only activated users will be returned. Default is <code>false</code> .	boolean

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/user/list' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "fa7bf873fab9333144e171372a321b06"}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/user/list?  
hash=fa7bf873fab9333144e171372a321b06
```

## response

```
{  
  "success": true,  
  "list" : [{  
    "dealer_id": 5001,  
    "activated": true,  
    "verified": true,  
    "login": "user@test.com",  
    "first_name": "John",  
    "middle_name": "William",  
    "last_name": "Smith",  
    "legal_name": "E. Biasi GmbH",  
    "legal_type": "legal_entity",  
    "phone": "491761234567",  
    "post_country": "Germany",  
    "post_index": "61169",  
    "post_region": "Hessen",  
    "post_city": "Wiesbaden",  
    "post_street_address": "Marienplatz 2",  
    "registered_country": "Germany",  
    "registered_index": "61169",  
    "registered_region": "Hessen",  
    "registered_city": "Wiesbaden",  
    "registered_street_address": "Marienplatz 2",  
    "state_reg_num": "12-3456789",  
    "tin": "1131145180",  
    "okpo_code": "93281776",  
    "iec": "773101001",  
    "id": 38935,  
    "balance" : 10.01,  
    "bonus": 0,  
    "creation_date" : "2021-03-01 13:00:00",  
    "trackers_count": 10,  
    "comment": "about user"  
  }],  
  "count" : 1  
}
```

- `list` - array of JSON objects. A list of [user objects](#).
- `count` - int. Total number of records (ignoring offset and limit).

## errors

- [General](#) types only.

## read

Returns user info by its id.

*required permissions:* users: "read".

## parameters

name	description	type
user_id	An ID of a user to read.	int

## examples

### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/user/read' \  
-H 'Content-Type: application/json' \  
-d '{"hash": "fa7bf873fab9333144e171372a321b06", "user_id":  
231485}'
```

### HTTP GET

```
https://api.navixy.com/v2/panel/user/read?  
hash=fa7bf873fab9333144e171372a321b06&user_id=231485
```

## response

```
{  
  "success": true,  
  "value" : {  
    "dealer_id": 5001,  
    "activated": true,  
    "verified": true,  
    "login": "user@test.com",  
    "first_name": "John",  
    "middle_name": "William",  
    "last_name": "Smith",  
    "legal_name": "E. Biasi GmbH",  
    "legal_type": "legal_entity",  
    "phone": "491761234567",  
    "post_country": "Germany",  
    "post_index": "61169",  
    "post_region": "Hessen",  
    "post_city": "Wiesbaden",  
    "post_street_address": "Marienplatz 2",
```



```

    "registered_country": "Germany",
    "registered_index": "61169",
    "registered_region": "Hessen",
    "registered_city": "Wiesbaden",
    "registered_street_address": "Marienplatz 2",
    "state_reg_num": "12-3456789",
    "tin": "1131145180",
    "okpo_code": "93281776",
    "iec": "773101001",
    "id": 38935,
    "balance" : 10.01,
    "bonus": 0,
    "creation_date" : "2021-03-01 13:00:00",
    "trackers_count": 10,
    "comment": "about user"
  },
  "discount": {
    "value": 5.5,
    "min_trackers": 10,
    "end_date": "2021-03-01",
    "strategy": "sum_with_progressive"
  },
  "default_tariff_id": 123
}

```

- `value` - JSON object. [User object](#) described above.
- `discount` - JSON object. [Discount object](#) described above.
- `default_tariff_id` - integer number, nullable. ID of a tariff plan which will be applied to user's trackers by default.

## errors

- 201 - Not found in the database – when user with specified ID not found or belongs to other dealer.

## update

Updates existing user with new field values (see [user object](#)). User must exist and must belong to authorized dealer. Changing of `legal_type` is not permitted, i.e. this field will not be changed.

*required permissions:* `users: "update"`.

## parameters

name	description	type
user		JSON object

name	description	type
	User object without comment and read-only fields.	
discount	Discount object.	JSON object
default_tariff_id	Optional. ID of a default tariff plan for user's trackers	int
comment	Comment	String, max length 255, only printable characters

If `user.verified` not passed then it set equal to `user.activated`.

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/user/update' \
-H 'Content-Type: application/json' \
-d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "user":
{"dealer_id": 5001, "activated": true, "verified": true, "login":
"user@test.com", "first_name": "John", "middle_name": "William",
"last_name": "Smith", "legal_name": "E. Biasi GmbH", "legal_type":
"legal_entity", "phone": "491761234567", "post_country":
"Germany", "post_index": "61169", "post_region": "Hessen",
"post_city": "Wiesbaden", "post_street_address": "Marienplatz 2",
"registered_country": "Germany", "registered_index": "61169",
"registered_region": "Hessen", "registered_city": "Wiesbaden",
"registered_street_address": "Marienplatz 2", "state_reg_num":
"12-3456789", "tin": "1131145180", "okpo_code": "93281776", "iec":
"773101001", "id": 38935}, "discount": {"value": 5.5,
"min_trackers": 10, "end_date": null, "strategy":
"sum_with_progressive"}, "comment": "about user"}'
```

### response

```
{
  "success": true
}
```

### errors

- 201 - Not found in the database – if specified user does not exist or belongs to different dealer.
- 206 - Login already in use – if specified "login" is used by another user.

## session/create

Creates an interface session for specified user and returns the hash for the created session.

*required permissions:* [users: "update", user\_sessions: ["create", "global"]].

user\_sessions: "global" - Optional. Allows sessions of users creation, not only owned by a current dealer.

### parameters

name	description	type
user_id	An ID of a user to create session.	int

### examples

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/user/session/create' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "user_id": 231485}'
```

#### HTTP GET

```
https://api.navixy.com/v2/panel/user/session/create?
hash=fa7bf873fab9333144e171372a321b06&user_id=231485
```

### response

```
{
  "success": true,
  "hash" : "a2caa32267f028bd41b982980467132c"
}
```

- hash - string. Hash of the created session.

### errors

- 201 - Not found in the database – if specified user does not exist or belongs to different dealer.

## transaction/change\_balance

Changes user balance (increase or decrease) or bonus and write this change in transactions (type = `payment`, subtype = `partner`).

New balance (bonus) must be not negative.

*required permissions:* [`users: "update"`, `transactions: "create"`].

### parameters

name	description	type
user_id	An ID of user whom balance changed.	int
amount	Amount to change. Can be negative.	double (2 digits after decimal mark)
type	Type of balance to change. Can be "balance" or "bonus".	enum
text	Description of transaction.	string (min length is 5 chars)

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/user/transaction/change_balance' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "fa7bf873fab9333144e171372a321b06", "user_id": 231485, "amount": 2.05, "type": "balance", "text": "additional payment"}'
```

### response

```
{
  "success": true
}
```

### errors

- 201 – Not found in the database – if user not found or not owned by a current dealer.

- 251 – Insufficient funds (403) – if user have not enough funds to withdraw passed (negative) amount.

## transaction/list

Gets list of user's billing transactions for the specified period. Same as [/transaction/list](#) from main api.

*required permissions:* [users: "read", transactions: "read"].

### parameters

name	description	type
user_id	An ID of user whom transactions listed. must be owned by a current dealer.	int
from	Start date/time for searching.	date/ time
to	End date/time for searching. Must be after "from" date.	date/ time
limit	Optional. A maximum number of the returned transactions.	int

### example

#### cURL

```
curl -X POST 'https://api.navixy.com/v2/panel/user/transaction/
list' \
-H 'Content-Type: application/json' \
-d '{"hash": "fa7bf873fab9333144e171372a321b06", "user_id":
231485, "from": "2020-02-03 03:04:00", "to": "2021-02-03
03:04:00"}'
```

### response

```
{
  "success": true,
  "list": [{
    "description": "Recharge bonus balance during tracker
registration",
    "type": "bonus_charge",
    "subtype": "register",
    "timestamp": "2021-01-28 08:16:40",
```

```

    "user_id": 12203,
    "dealer_id": 5001,
    "tracker_id": 303126,
    "amount": -10.0000,
    "new_balance": 800.0000,
    "old_balance": 810.0000,
    "bonus_amount": 10.0000,
    "new_bonus": 10.0000,
    "old_bonus": 0.0000
  }
}

```

- `list` - array of objects. List of transaction objects.
  - `description` - string. Transaction description.
  - `type` - `enum`. Type of transaction.
  - `subtype` - `enum`. Subtype of transaction.
  - `timestamp` - `date/time`. When transaction created.
  - `user_id` - int. ID of a user which made a transaction.
  - `dealer_id` - int. ID of a dealer.
  - `tracker_id` - int. Tracker id. 0 if transaction not associated with tracker.
  - `amount` - double. Amount of money in transaction, can be negative. e.g. -10.0000 means 10 money units removed from user's balance.
  - `new_balance` - double. User's money balance after transaction.
  - `old_balance` - double. User's money balance before transaction.
  - `bonus_amount` - double. Amount of bonus used in transaction, can be negative. e.g. 10.0000 means 10 bonuses units added to user's bonus balance.
  - `new_bonus` - double. User's bonus balance after transaction.
  - `old_bonus` - double. User's bonus balance before transaction.

## errors

- 201 – Not found in the database - if user not found or not owned by a current dealer.

## upload

Upload users from CSV or XLS file.

**MUST** be a POST multipart request (multipart/form-data), with one of the parts being a CSV or XLS file upload (with the name "file").

CSV column separator is `;`. Columns header for CSV and XLS (headers with `*` is required):

```
Email address*;Password*;Status*;Legal status*;Surname*;Name*;Middle
name;Phone number;Comment;Country;Region;City;Street, address;Zip code;Legal
name;Tax number;IEC;Registration country;Registration region;Registration
city;Registration address;Registration zip code;Discount;End date of
discount;Device limit
```

For RU locale:

```
Адрес электронной почты*;Пароль*;Статус*;Юридический
статус*;Фамилия*;Имя*;Отчество;Номер
телефона;Комментарий;Страна;Регион;Город;Улица, дом, квартира;Почтовый
индекс;Юридическое название;ИНН;КПП;ОГРН;ОКПО;Страна регистрации;Регион
регистрации;Город регистрации;Улица, дом регистрации;Почтовый индекс
регистрации;Скидка;Дата окончания скидки;Минимальное число устройств для
скидки
```

Legal status must be one of the following numbers:

- 1 - individual
- 2 - legal entity
- 3 - sole trader

For legal entity (2) and sole trader (3) in addition to the required `*` the following columns must be present and filled with data:

```
Country;Region;City;Street, address;Zip code;Legal name;Registration
region;Registration city;Registration address;Registration zip code
```

Except `Legal name` for sole trader (3) it is not required.

The remaining columns are optional and can be omitted. All columns can be in any order.

New users will be created with the time zone specified in `default_user_time_zone` service [setting](#).

*required permissions:* `[users: "create"]`.

## parameters

name	description	type
file	A XLS or CSV file containing users data.	File upload
redirect_target	Optional URL to redirect. If <b>redirect_target</b> passed return redirect to <code>&lt;redirect_target&gt;?response=&lt;urlencoded_response_json&gt;</code> .	string

## response

```
{
  "success": true,
  "total": 1,
  "errors": 0
}
```

## errors

Most error responses include `row_number` - the line number in the file where the error was found.

- 206 – Login already in use – if this email already registered.

```
{
  "row_number" : 2,
  "status" : {
    "code" : 206,
    "description" : "Login already in use"
  },
  "success" : false
}
```

- 273 – Duplicate login in source file.

```
{
  "row_number" : 4,
  "status" : {
    "code" : 273,
    "description" : "Duplicate login"
  },
  "success" : false
}
```

- 274 – Empty data file. No rows to load were found in the source file.



```
{
  "status" : {
    "code" : 274,
    "description" : "Empty data file"
  },
  "success" : false
}
```

- 7 – Invalid parameters. Required columns not found or there has data validation errors.

```
{
  "errors" : [ {
    "error" : "required column not found",
    "parameter" : "users_import.password"
  }, {
    "error" : "required column not found",
    "parameter" : "users_import.email"
  } ],
  "row_number" : 1,
  "status" : {
    "code" : 7,
    "description" : "Invalid parameters"
  },
  "success" : false
}
```

```
{
  "errors" : [ {
    "error" : "E-mail must be valid",
    "parameter" : "user.login"
  } ],
  "row_number" : 2,
  "status" : {
    "code" : 7,
    "description" : "Invalid parameters"
  },
  "success" : false
}
```

Last update: October 6, 2023





# Navixy Eco Fleet API

The structure of Eco Fleet API is close to the user API, so we highly recommend reading [Backend API: getting started](#).

The main differences are *request paths*, *authorization system* and *request format*.

## Base URL

Eco Fleet API resides in `eco_fleet` subsection of API URL. So you can determine URL to API calls like this:

- `https://api.eu.navixy.com/eco_fleet` for European Navixy ServerMate platform.
- `https://api.us.navixy.com/eco_fleet` for American Navixy ServerMate platform.

For example, to make a sensor quality API call in European Navixy ServerMate, you should use the URL:

```
https://api.navixy.com/eco_fleet/v1/trackers/123/sensors/321/  
quality
```

## Auth

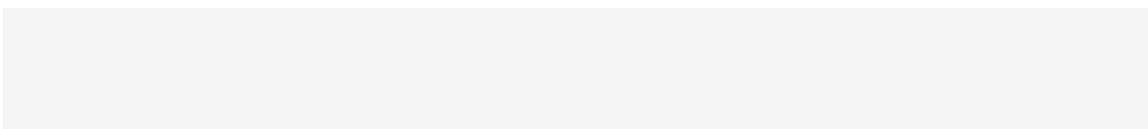
### Authentication

Authentication is handled by [Backend API](#).

### Authorization

You should pass the session hash you obtained earlier as the `Authorization` HTTP header with `NVX` auth scheme.

Example:



```
$ curl -X GET 'https://api.navixy.com/eco_fleet/v1/trackers/123/sensors/321/quality' \
-H 'Authorization: NVX 5dd33ef0ab37b6aaf2064ecdf50c4cdc'
```

## Response format

The responses are usually in `application/json` content type. Consult the API call documentation in question for details.

## Errors

Errors are distinguished by HTTP status code ( $\geq 400$ ) and follow [RFC 7807](#).

Example:

```
{
  "type": "errors/default/bad-request",
  "title": "Bad Request",
  "status": 400,
  "detail": "id: must be greater than or equal to 1"
}
```

### Common error types

- `errors/default/bad-request` - Causes: missing or invalid parameter value.
- `errors/default/unauthorized` - Causes: missing `Authorization` header or credentials are insufficient or expired.

## Date/time formats

According to [ISO 8601](#).

Example: `1999-12-31T23:59:59Z`.

Last update: December 1, 2023





# Fuel data resampling

**Navixy Eco Fleet Lab has developed a solution for comprehensive fuel data analysis. This method regenerates data sets which will be beneficial for variety of exercises, enabling researchers, developers, and diagnostics experts to leverage the processed data.**

**Diagnostic process** is an important procedure undertaken by numerous partners and investigators. Its primary purpose is to identify the underlying causes, which is essential for efficient fuel management by identifying any abnormalities. For example, they analyze key events - drains and refueling for fraud or look for a reason for the appearance of noise in the fuel data.

When partners or integrators incorporate fuel-related data into third-party systems for **further analysis and processing, including machine learning**, it can be highly effective in identifying behavioral patterns and detecting exceptions. This allows for more efficient investigations and enhances overall data processing capabilities.

Effective data management relies on **accurate and synchronized raw data**. However, inconsistencies in different data sets (i.e. position, speed, and fuel level) across various timeframes can pose challenges.

To tackle the challenges posed by incomplete or inconsistent data, we utilize advanced statistical models. With our API requests, you can easily **access and download processed datasets** from our platform for specific time periods. This API request specifically provides fuel-related data in a convenient CSV format.

**Data resampling refers to the joint process of creating a uniform data structure by organizing existing values and generating new ones (for missing values) in a chronological order, while considering equal time intervals. This approach ensures data integrity and facilitates analysis.**

## data\_resampling

### Description

The response is presented in a convenient CSV table format, incorporating columns below:

- Time - timestamp (depending on the timezone the tracker is located)



- [FUEL\_SENSOR\_NAME] - Fuel level (The column name is derived based on the sensor name. There could be more than 1 column)
- SPEED - object speed (km/h)
- MOVEMENT - movement status (0 - parking, 1 - moving, 2 - Idle)
- LNG - Longitude
- LAT - Latitude

**API path:** /trackers/\$tracker\_id/resampling

## Parameters

### Standard list

name	description	type	format
hash	Hash of an API key is required for API calls to identify user.	string	"NVX c459c3589336ebf26ff18257a
tracker_id	ID of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked.	int	123456
interval	Sensor readings' datetime interval which will be analyzed. Last week by default.	interval	2023-08-24T08:04:36.306Z/ 2023-08-26T08:04:36.306Z

name	description	type	format
step_size	Resampling step in minutes	int	1
resampling_props	Array of data resampling parameters for various data types	resampling_props	array

#### Additional list of resampling parameters

name	description	type	format
data_type	Data type. Options: FUEL, SPEED, MOVEMENT, LNG, LAT	string	"SPEED"
resampling_method	Resampling method. Options: FOLLOWING, PREVIOUS, MEDIAN_IN_WINDOW, AVERAGE_IN_WINDOW, AVERAGE.	string	"FOLLOWING"
delta	Delta time in seconds. Has different meaning for different algorithms.	int	null/1
fixed_value	A fixed value that will be used if no data is found in the interval. You can specify null or any number.	float	null/5.5

#### DESCRIPTION OF RESAMPLING\_METHOD

- FOLLOWING - In this series, the subsequent value is utilized to substitute any missing values. Consequently, neighboring missing values are all substituted with the subsequent valid value. If there are any missing values at the end of the series, they are replaced with the preceding valid value. If delta is not null than algorithm

changes: if the time interval  $[T-\Delta, T+0]$  contains at least one value, otherwise `fixed_value`.

- **PREVIOUS** - The previous value in the series is utilized to substitute missing values. Consequently, the neighboring missing values are replaced with the earliest preceding valid value. If delta is not null than algorithm changes: if the time interval  $[T-0, T+\Delta]$  contains at least one value, otherwise `fixed_value`.
- **MEDIAN\_IN\_WINDOW** - In this algorithm, the presence of delta is imperative. The average of all the neighboring values in the series within the interval  $[T-\Delta, T+\Delta]$ , `fixed_value` if no values.
- **AVERAGE\_IN\_WINDOW**- In this algorithm, the presence of delta is imperative. The median of all the neighboring values in the series within the interval  $[T-\Delta, T+\Delta]$ , `fixed_value` if no values.
- **AVERAGE** - To replace missing values in a series, we use the average of the two neighboring values. For any missing values between valid ones, we replace them with the average of the surrounding valid values. If the series begins or ends with missing values, we substitute them with the next or previous valid value accordingly. If delta is not equal to null than algorithm changes: if the interval  $[T-\Delta, T+\Delta]$  contains at least one value, otherwise `fixed_value`.

**We recommend utilizing distinct methods for varying data types, as outlined in the table below. However, the choice of which methods to employ ultimately depends on your individual needs and expectations.**

Method	Data type	Use case
Following	Ordered	Data that is missing at the end of a time series or sequence, i.e Fuel, Movement
Previous	Ordered	Data that is missing at the beginning of a time series or sequence, i.e Fuel, Movement
Median	Evenly distributed	Data that is not normally distributed, i.e. Fuel, Speed
Average	Normally distributed	Data that is not evenly distributed, i.e. Fuel, Speed

## Example

```

curl -X 'POST' \
  'https://api.navixy.com/eco_fleet/v1/trackers/12345/resampling?
interval=P7D/2020-12-31T00:00Z' \
  -H 'accept: text/csv' \
  -H 'Authorization: NVX 22eac1c27af4be7b9d04da2ce1af111b' \
  -H 'Content-Type: application/json' \
  -d '{
    "step_size": 1,
    "resampling_props": [
      {
        "data_type": "FUEL",
        "resampling_method": "PREVIOUS",
        "delta": 600,
        "fixed_value": null
      },
      {
        "data_type": "SPEED",
        "resampling_method": "PREVIOUS",
        "delta": null,
        "fixed_value": null
      },
      {
        "data_type": "MOVEMENT",
        "resampling_method": "PREVIOUS",
        "delta": null,
        "fixed_value": null
      },
      {
        "data_type": "LNG",
        "resampling_method": "PREVIOUS",
        "delta": null,
        "fixed_value": null
      },
      {
        "data_type": "LAT",
        "resampling_method": "PREVIOUS",
        "delta": null,
        "fixed_value": null
      }
    ]
  }'

```

## response

```

Time,Fuel level,SPEED,MOVEMENT,LNG,LAT
2023-08-24T08:04,132.85823,38.0,1,37.68325,55.580612
2023-08-24T08:05,132.85823,38.0,1,37.68325,55.580612
2023-08-24T08:06,131.57695,38.0,1,37.68325,55.580612
2023-08-24T08:07,128.12737,38.0,1,37.68325,55.580612
2023-08-24T08:08,130.59135,38.0,1,37.68325,55.580612

```

## Errors

- 204 - Entity not found – if there is no tracker with such ID belonging to authorized user.
- 208 - Device blocked – if tracker exists but was blocked due to tariff restrictions or some other reason.

Last update: December 28, 2023





# Fuel Sensor Quality Index

Contains API calls to interact with fuel sensor quality index.

## Resource

Resource path: `/trackers/{tracker_id}/sensors/{sensor_id}/quality`.

## GET

Returns the fuel sensor quality index calculated from sensor readings within a specified datetime period.

### parameters

name	description	type
tracker_id	ID of the tracker which has the sensor.	integer
sensor_id	ID of the sensor to analyze.	integer
interval	Sensor readings' datetime interval which will be analyzed. Last week by default.	interval

### examples

```
curl -X GET 'https://api.navixy.com/eco_fleet/v1/trackers/123/sensors/321/quality?interval=P7D/2020-12-31T00:00Z' \
-H 'Authorization: NVX 22eac1c27af4be7b9d04da2ce1af111b'
```

### response

```
{
  "smoothness": 8.29
}
```

- `smoothness` - a smoothness score of the sensor readings. Higher values indicate reduced noise in sensor readings, while lower values suggest increased noise.



## TYPES

Name	Description	JSON type	Restrictions
Score	An abstract measurement score.	number	$\geq 1.0 \ \&\& \ \leq 10.0$

## errors

- `errors/entity/not-found` - Entity not found. Thrown if sensor or calibration table is missing.
- `errors/external-api/navixy` - Error accessing Navixy API. See `detail` field and consult [Backend API documentation](#).
- `errors/sensors/quality/not-enough-readings` - Not enough sensor readings in given interval. Try using interval with enough vehicle usage or changing readings' sending frequency and waiting for data accumulation.

Last update: December 4, 2023





# Navixy Data Warehouse API

The structure of Data Warehouse API is mostly similar to [Backend API](#), so if you're familiar with the basics of user API, this will be a great advantage.

## Base URL

Data Warehouse API resides in `dwh` subsection of API URL and does not belong to backend APIv2. You need to determine URL to API calls like this: \* `https://api.eu.navixy.com/dwh/v1` for European Navixy ServerMate platform. \* `https://api.us.navixy.com/dwh/v1` for American Navixy ServerMate platform.

For example, to make raw data readings API request in European Navixy ServerMate, you need to use this URL:

```
https://api.eu.navixy.com/dwh/v1/tracker/raw_data/read
```

## Auth

### Authentication

Authentication is handled by [Backend API](#).

### Authorization

Requests to Data Warehouse API are made using user session hash or API key. It can be passed as the Authorization HTTP header with NVX auth scheme, or within a `-d` (data) command.

Example:

### with Authorization header

```
curl -X 'POST' \
  'https://api.eu.navixy.com/dwh/v1/tracker/raw_data/get_inputs' \
  -H 'accept: text/csv' \
  -H 'Content-Type: application/json' \
  -H 'Authorization: NVX 8a41497ed8e77fa68b9c4a9420971fdb' \
  -d '{"tracker_id": 123456}'
```

### with hash

```
curl -X 'POST' \
  'https://api.eu.navixy.com/dwh/v1/tracker/raw_data/get_inputs' \
  -H 'accept: text/csv' \
  -H 'Content-Type: application/json' \
  -d '{"hash": "6dc7d304dec4434f4c4202ec42817f83","tracker_id": 123456}'
```

## Response format

Depending on the API request, the responses can be in application/json or CSV markdown content types.

## Errors

Errors are distinguished by HTTP status code ( $\geq 400$ ) and follow [RFC 7807](#).

Example:

```
{
  "type": "errors/default/bad-request",
  "title": "Bad Request",
  "status": 400,
  "detail": "id: must be greater than or equal to 1"
}
```

### Common error types

- `errors/default/bad-request` - Causes: missing or invalid parameter value.
- `errors/default/unauthorized` - Causes: missing `Authorization` header or credentials are insufficient or expired.

## Date/time formats

According to [ISO 8601](#). Described details on date/time formats and examples in [Raw data request - date and time](#).

Last update: December 20, 2023







## Requesting tracker raw data

Sometimes, TSP, integrators, and developers need to look at the original, unprocessed data from tracking devices, often called "raw data". This helps them to get more accurate information and a better understanding of the collected data. It can also help fix issues with the tracking process. Plus, they can utilize this raw data in other systems for more analysis or use in different parts of the business.

### Typical use case

To efficiently manage fuel consumption, especially for fleet management or cargo transportation businesses, integrating your ERP system with the Navixy platform is crucial. This integration enables you to seamlessly request raw fuel data from specific devices, enhancing your overall fuel monitoring and expense tracking capabilities.

In this situation, you may need to retrieve the following data:

- Fuel Level
- Device location
- Mileage (odometer)
- Speed
- Date and time
- Any other data valuable to your business

It is important to realize that in this case we will not get human-readable information about fuel drains and refills, excessive consumption, idling and other analytics. Here we are talking about raw data received directly from the device and decoded by Navixy platform, in other words, instantaneous data readings for a certain period. Data obtained in this way is suitable for further processing and analyzing on your side.

### Requesting inputs list

Tracking devices from different manufacturers have different specifics of work and send data in different forms. In addition, sensors can be of different types: digital and analog, wired and wireless, built-in and external. Also, there can be several sensors monitoring the same type of readings: for example, two fuel sensors in two tanks, internal and external temperature sensors, etc.

It is for this reason that before requesting raw data, we have to understand what data the device is capable to report to the platform, and what the data inputs are named. To do this, we need to use `raw_data/get_inputs` request.

Example for a device with ID 123456:

#### cURL

```
curl -X 'POST' \
'https://api.eu.navixy.com/dwh/v1/tracker/raw_data/get_inputs' \
-H 'Content-Type: application/json' \
-d '{"hash": "6dc7d304dec4434f4c4202ec42817f83", "tracker_id": 123456}'
```

The platform will notify us about success, and we will see the following JSON object in response:

```
{
  "discrete_inputs": 2,
  "discrete_outputs": 1,
  "inputs": [
    "analog_1",
    "battery_voltage",
    "board_voltage",
    "ext_temp_sensor_4",
    "freq_1",
    "hw_mileage",
    "impulse_counter_1",
    "lls_level_4",
    "lls_temperature_4"
  ],
  "states": [
    "hardware_key"
  ],
  "success": true
}
```

Among the obtained inputs we see the one of interest to us - `lls_level_4`. This is a fuel level input, and the index 4 indicates the value is a range of LLS levels from `1` to `4`. This means the device can send fuel data on maximum of four inputs: from `lls_level_1` to `lls_level_4`.

We also see the `hw_mileage` which will allow us to get value of the hardware odometer.

**The presence of some inputs in the received response does not mean that data is certainly available on these inputs. It means that data may come on them, but whether it is actually available or not depends on the configuration of a particular device.**

## Requesting raw data readings

Now we know the name of the inputs where the fuel data comes from, and we can use them in the API query. However, we also need other parameters for more accurate analysis. This data is not related to sensors, so the above query does not return it. They come in simple columns and are listed on the appropriate documentation page.

We will use the following columns:

- `lat`
- `lng`
- `speed`

In addition, we will use names for inputs according to the information obtained earlier:

- `inputs.lls_level_1`
- `inputs.hw_mileage`

**We specify `inputs.lls_level_1` because we know that our device only sends data on this input. If we didn't know the input number, we could have specified all four possible inputs, and then the inputs without data would just get zero values.**

The API request for raw data in our case must look like below:

### cURL


```
curl -X 'POST' \
'https://api.eu.navixy.com/dwh/v1/tracker/raw_data/read' \
-H 'accept: text/csv' \
-H 'Content-Type: application/json' \
-d '{
  "hash": "6dc7d304dec4434f4c4202ec42817f83",
  "tracker_id": "123456",
  "from": "2023-11-29T08:31:00Z",
  "to": "2023-11-29T08:32:00Z",
  "columns":
  ["lat", "lng", "speed", "inputs.lls_level_1", "inputs.hw_mileage"]}'
```

The response is returned in a CSV table format:

```
"msg_time","lat","lng","speed","inputs.lls_level_1","inputs.hw_mileage"
"2023-11-29T08:31:10Z",54.2312716,69.5261833,0,3307,24250.798
"2023-11-29T08:31:12Z",54.1811183,69.5331349,24,3274,24257.16
"2023-11-29T08:31:27Z",54.1802066,69.5333599,21,\N,\N
"2023-11-29T08:31:27Z",54.1802066,69.5333599,21,3274,24257.262
"2023-11-29T08:31:30Z",54.180055,69.533395,20,3274,24257.279
"2023-11-29T08:31:31Z",54.180005,69.5334083,20,3274,24257.285
"2023-11-29T08:31:33Z",54.179915,69.5334266,17,3274,24257.295
```

```
"2023-11-29T08:31:34Z", 54.1798766, 69.533435, 15, 3274, 24257.299
"2023-11-29T08:31:36Z", 54.1798116, 69.5334433, 12, 3274, 24257.306
"2023-11-29T08:31:37Z", 54.179785, 69.5334416, 10, 3274, 24257.309
"2023-11-29T08:31:39Z", 54.1797366, 69.533425, 9, 3274, 24257.315
"2023-11-29T08:31:40Z", 54.1797183, 69.5334083, 8, 3274, 24257.315
"2023-11-29T08:31:42Z", 54.1796883, 69.5333449, 10, 3274, 24257.322
"2023-11-29T08:31:43Z", 54.17968, 69.5333016, 12, 3274, 24257.325
"2023-11-29T08:31:45Z", 54.1796816, 69.53318, 15, 3274, 24257.333
"2023-11-29T08:31:47Z", 54.1796816, 69.533025, 20, 3274, 24257.343
```

In the above example we see the output only for one minute of tracking. When querying raw data over a long period of time, the response can reach significant sizes - this must be considered.

 In one of the lines we see `\N` instead of fuel level and mileage values. This means that no such information was received in this data packet. The `\N` symbol represents `NULL`.

The above example is one of the simplest, but clearly demonstrates the process of using an API request to read raw data. You can query a lot of data at once and over large periods of time, depending on your objectives.

Last update: December 27, 2023






## Raw data request - date and time


When requesting raw data, you must specify the exact period of time for which you need the data, only in this case the platform is able to correctly process your request and return you the necessary information.

As in the Backend API, here you can specify date and time either in the usual `YYYYY-MM-DD HH:mm:ss` form with or without time zone, or in accordance with ISO 8601.

The difference is that ISO 8601 is the default format for Data Warehouse API requests.

 **The platform allows you to request raw data for any period not exceeding 30 days back from the current date. Raw data for earlier periods will not be requested.**

### ISO 8601

 **ISO 8601 is an international standard of representing date and time-related data in unambiguous form, both human- and machine readable. However, some programs, including Microsoft Excel, may not be able to read such timestamps, therefore it is optional.**

Using this date/time standard when requesting raw data, you can specify the exact time for which you need the data, as well as the time zone - not only as in the user account or UTC+0, but any other time zone of your choice, if required.

According to ISO 8601, the date and time are represented starting with year, followed by month, day, hour, minutes, seconds, milliseconds and time zone offset.

Date and time format:

```
[yyyy]-[mm]-[dd]T[hh]:[mm]:[ss]±[offset]
```

Time zone offset can be specified using any of the following formats:

- `+--HH:mm` - for example, -06:00 or +05:30.
- `+--HHmm` - for example, -0500 or +0100.
- `+--HH` - for example, -03 or +07.
- `Z` - no offset (UTC+0).

## Timestamp examples:

Let's assume that the client account is set to UTC+4 (Dubai) time, and the time we need is 10:20 AM, the date is December 2, 2023. Then we can specify the timestamp in one of the following ways:

- `2023-12-02T10:20:00+04:00`
- `2023-12-02T10:20:00+04`
- `2023-12-02T06:20:00Z` (converted to UTC+0)


Another example. The client account is set to UTC-6 (Mexico) time, and the time we need is 10:55 PM, the date is December 11, 2023. Then we can specify the timestamp in one of the following ways:

- `2023-12-11T23:55:00-06:00`
- `2023-12-11T23:55:00-06`
- `2023-12-12T04:55:00Z` (converted to UTC+0, mind the date)

## API request example:

### cURL

```
curl -X 'POST' \
'https://api.eu.navixy.com/dwh/v1/tracker/raw_data/read' \
-H 'accept: text/csv' \
-H 'Content-Type: application/json' \
-d '{
  "hash": "6dc7d304dec4434f4c4202ec42817f83",
  "tracker_id": "10033823",
  "from": "2023-12-12T09:00:00Z",
  "to": "2023-12-12T09:25:00Z",
  "columns": ["lat", "lng", "discrete_inputs.1", "inputs.board_voltage"]}'
```

 **The output for a raw data request will always contain a `msg_time` column that contains time stamps according to user account time zone. If you need to obtain `msg_time` in any other time zone, please refer to Time zone section below.**

## Regular date and time

Another valid option to specify date and time is the usual `YYYY-MM-DD HH:mm:ss` format.

Since this is not the default format, you need to specify the parameter `iso_datetime=false` in the API request.



## API request example:

### cURL

```
curl -X 'POST' \
'https://api.eu.navixy.com/dwh/v1/tracker/raw_data/read' \
-H 'accept: text/csv' \
-H 'Content-Type: application/json' \
-d '{
  "hash": "6dc7d304dec4434f4c4202ec42817f83",
  "iso_datetime": false,
  "tracker_id": "10033823",
  "from": "2023-12-12 14:00:00",
  "to": "2023-12-12 14:25:00",
  "columns": ["lat", "lng", "discrete_inputs.
1", "inputs.board_voltage"]}'
```

In this case, the retrieved data will be **in the time zone of the user account**.

## Time zone

There may be situations when you need to obtain data in some specific time zone different from user account. This can be useful when the customer's time zone differs from yours due to geographical reasons.

In this case you need to supplement your request with the `time_zone` parameter and specify the required zone ID. You can request all the possible zone IDs using [timezone/list](#) request from Backend API.

## API request example:

### cURL

```
curl -X 'POST' \
'https://api.eu.navixy.com/dwh/v1/tracker/raw_data/read' \
-H 'accept: text/csv' \
-H 'Content-Type: application/json' \
-d '{
  "hash": "6dc7d304dec4434f4c4202ec42817f83",
  "iso_datetime": false,
  "time_zone": "Europe/London",
  "tracker_id": "10033823",
  "from": "2023-12-12 14:00:00",
  "to": "2023-12-12 14:25:00",
  "columns": ["lat", "lng", "discrete_inputs.
1", "inputs.board_voltage"]}'
```

## Time period

When requesting raw data, you have an option of specifying the request period in two ways: by specifying the date and time "from" and "to" or by specifying an interval. Both methods are equally valid, but you should use only the one of your choice.

### "from" and "to"

A common way to indicate the period of data request is to specify two timestamps of start and end. This is done using the `from` and `to` parameters. The values are specified either according to ISO 8601 or in a regular `YYYY-MM-DD HH:mm:ss` form - as described above.

Examples:

```
"from": "2023-11-30T17:00:00-06:00",  
"to": "2023-11-30T18:00:00-06:00",
```

or

```
"from": "2023-11-30 17:00:00",  
"to": "2023-11-30 18:00:00",
```

**Note that the `to` date and time must be after the `from`, otherwise the query will result in an `Invalid parameters` error.**

## Interval

An alternative method of indicating the request period is an interval. Here you specify the start or end date and time of the period appended by duration of the period.

**When specifying the `interval` parameter, then `from` and `to` parameters must not be specified. These are mutually exclusive ways of specifying the data request period.**

The interval can be specified in different forms:

- Starting from a specific date and time.
- Ending with a specific date and time
- Indicated by start and end timestamps, without a period.

Possible `interval` parameter formats:

```
[start date and time]/P[dd]T[hh]H[mm]M[ss]S
```

or

```
P[dd]T[hh]H[mm]M[ss]S/[end date and time]
```

or

```
[start date and time]/[end date and time]
```

The date and time can be specified either according to ISO 8601 or in a regular form.

**PT** in interval value stands for "Period and Time" and indicates the period after the specified time stamp. For example, **PT1H30M15S** means 1 hour 30 minutes 15 seconds.

If you need to request data for several days, you can specify the amount of days between **P** and **T**. For example, **P2DT3H45M10S** means 2 days 3 hours 45 minutes 10 seconds.

Examples:

- "interval": "2023-11-30T17:00:00-0600/PT1H30M10S" - data will be requested from 17:00:00 to 18:30:10, November 30 (UTC-6).
- "interval": "2023-11-30 17:00:00/P2DT2H45M10S" - data will be requested from November 30, 17:00:00 to December 2, 19:45:10 (according to user account time zone).
- "interval": "P2DT2H45M10S/2023-11-30 17:00:00" - data will be requested from November 28, 14:14:50 to November 30, 17:00:00 (according to user account time zone).



**All of the above methods of specifying the date and time are equally correct. Therefore, you can choose any of them that you find more convenient or that better matches the format used in your integrations.**

Last update: February 2, 2024






# Raw data

Tracking devices vary significantly in terms of form factor, connectivity, and use cases, but their most substantial divergence lies in the data they monitor and report. Even when dealing with various device types, the datasets can differ significantly; however, the platform processes all data and stores it in the database in some manner.

This API requests allows you to get and download parsed raw data from the platform for the various period of time, but not more than last 30 days. This API request returns data for all inputs and state fields which the tracker transmitted.

 **Parsed raw data - Data obtained immediately after decoding (parsing) incoming data packets, taking into account the protocol and specifics of the device model from which the packets were received.**

## API actions

API base path: `/tracker/raw_data/`

### get\_inputs

Returns available metering inputs and state fields of a device.

Before requesting raw data, it is crucial to comprehend the device's data capabilities and the specific names of the data input and state fields it possesses.

It's important to note that this API request does not provide actual device data. Rather, it serves as a supplementary request aimed at gaining insights into the fields, which data that can be obtained subsequently.

### parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). The tracker must be associated with the user whose hash is being used for the request, and not tariff-blocked.	int	123456

## example

### cURL

```
curl -X 'POST' \
'https://api.eu.navixy.com/dwh/v1/tracker/raw_data/get_inputs' \
-H 'Content-Type: application/json' \
-d '{"hash": "6dc7d304dec4434f4c4202ec42817f83", "tracker_id": 123456}'
```

## response

```
{
  "discrete_inputs": 2,
  "discrete_outputs": 1,
  "inputs": [
    "analog_1",
    "battery_voltage",
    "board_voltage",
    "ext_temp_sensor_4",
    "freq_1",
    "hw_mileage",
    "impulse_counter_1",
    "lls_level_4",
    "lls_temperature_4"
  ],
  "states": [
    "hardware_key"
  ],
  "success": true
}
```

- `discrete_inputs` - int. A number of discrete inputs.
- `discrete_outputs` - int. A number of discrete outputs.
- `inputs` - string array. A list of available metering inputs.
- `states` - string array. A list of available state fields.

If there is more than one input of the same type, they are indexed (1, 2, 3...). In this case, only input with the maximum index is returned.

For example:

- If LLS Levels from 1 to 4 are available, `lls_level_4` is returned, and it is assumed that LLS levels 1 through 3 also exist.
- If AVL IOs from 1 to 100000 are available for a device, `avl_io_100000` is returned, and AVL IOs with smaller indexes also exist.


## errors

- 201 – Not found in the database – tracker ID does not exist.

- 204 – Entity not found – there is no tracker with such ID belonging to authorized user.
- 208 – Device blocked – tracker exists but was blocked due to tariff restrictions or some other reason.

## read

Retrieves parsed raw data - values received from tracking devices and decoded by the platform.

 The names and values of the inputs and state fields returned by this request align with the names visible in [Air Console](#) when connecting to a device. You can find them in the right column, where the incoming data is decoded.

## parameters

name	description	type	format
tracker_id	ID of the tracker (aka "object_id"). The tracker must be associated with the user whose hash is being used for the request, and not tariff-blocked.	int	123456
from	From date/time. Starting from what moment logs messages should be	date/time	"2023-08-24T08:04:36Z"



name	description	type	format
	<p>retrieved. It relates to the message time - when the packet was registered by a tracker. The time is specified along with time zone according to ISO8601.</p>		
to	<p>To date/time. Till which moment messages should be retrieved into log. It relates to the message time - when the packet was registered by a tracker. Specified date must be after "from" date. The time is specified along with time zone</p>	date/time	"2023-08-24T08:04:36Z"

name	description	type	format
	according to ISO8601.		
columns	List of CSV columns to retrieve	string array	<code>["flags.location_valid", "lat", "lng", "1", "inputs.board_voltage"]</code>
server_time_filter	Optional interval for additional filtering message by server time. If it is used - messages will be returned not only be message time - when the packet was registered by a tracker, they will be filtered by server time - when the message sent to the server.	string/object	<code>"2024-02-03T10:26:26+0500/2024-02-03T10:26:26+0500", "to": "2024-02-03T10:26:26+0500", {"interval": "2024-02-03T10:26:26+0500/2024-02-03T10:26:26+0500"}</code>



Instead of using from/to parameters it is possible to set interval parameter - ISO8601 formatted interval, for example 2023-08-24T08:04:36.306Z/PT24H.

The response is provided in a CSV format file, with columns that are predefined in the `columns` parameter of the API request. Here are the specifications for the table output:

- Rows are enclosed in double quotes. A double quote inside a string is output as two double quotes in a row. There are no other rules for escaping characters.
- Date and date-time are enclosed in double quotes.
- Numbers are output without quotes.
- Values are separated by a comma character `,`.
- Rows are separated using the Unix line feed (LF).
- `NULL` is represented as `\N`.
- Requested column can be a simple or complex.

**Simple columns:**

- `msg_time` - time of message sent by device. Always returned in CSV output and does not need to be requested separately. Indicated according to user account time zone.
- `server_time` - time of message processing by the server.
- `gps_fix_type` - enum. One of `UNKNOWN`, `NO_FIX`, `HAS_FIX`, `LAST_KNOWN_POSITION`.
- `lat` - float. Latitude.
- `lng` - float. Longitude.
- `speed` - decimal. Speed, km/h.
- `alt` - int. Altitude, meters.
- `satellites` - int. Satellites count (`-1` = unknown).
- `heading` - int. Heading degrees.
- `precision` - int. Location precision, meters.
- `hdop` - float. Horizontal dilution of precision (`-1` = unknown).
- `pdop` - float. Position dilution of precision (`-1` = unknown).
- `event_id` - int. Event ID.
- `mn_name` - string. Mobile network name.
- `mn_roaming` - int. Roaming status (`0` = no roaming, `1` = roaming, `-1` = unknown).
- `mn_code` - int. Mobile network operator code.
- `mn_csq` - int. Mobile network signal strength, CSQ, values from 0 to 31 (`99` = unknown).
- `mn_type` - enum. Mobile network type, one of `UNKNOWN`, `GSM`, `CDMA`, `WCDMA`, `LTE`, `NR`.

### Complex columns:

- `flags` - bitmap of flags:
  - bit 0 `location_valid`: 0 = location invalid, 1 = location valid.
  - bit 1 `lbs`: 0 = GPS, 1 = LBS.
  - bit 2 `soft_lbs`: 0 = device LBS, 1 = software LBS.
- `discrete_inputs` - map of discrete inputs states, inputs enumerated from 1.
- `discrete_outputs` - map of discrete outputs states, outputs enumerated from 1.
- `inputs` - map of metering inputs values. Inputs list depends on device.
- `states` - map of various states. States list depends on device.

To retrieve internal value from complex column, you need to use period symbol. For example: `flags.location_valid`, `inputs.board_voltage`. Unknown internal values will be returned as `NULL`.

The list of available internal values for a particular device is obtained using `get_inputs` method described above.

If you specify complex column without specifying internal value, then all internal values will be returned as JSON map (except flags that will be returned as integer).

You can append complex columns with an asterisk symbol:

- `inputs.*`
- `states.*`
- `discrete_inputs.*`
- `discrete_outputs.*`

In this case, the platform will search for all available columns in the specified data range and then request them from the database. In the resulting CSV output, instead of the column with an asterisk will be all the existing columns in alphabetical order. If there are no columns, they will not be shown in the response.

## example for standard searching by message time only

### cURL

```
curl -X 'POST' \
'https://api.eu.navixy.com/dwh/v1/tracker/raw_data/read' \
-H 'accept: text/csv' \
-H 'Content-Type: application/json' \
-d '{
  "hash": "6dc7d304dec4434f4c4202ec42817f83",
  "tracker_id": "123456",
  "from": "2023-11-30T07:13:00.000Z",
  "to": "2023-11-30T07:15:00.000Z",
  "columns":
  ["lat", "lng", "speed", "inputs.ble_lls_level_1", "inputs.hw_mileage", "dis
```

### response

```
"msg_time", "lat", "lng", "speed", "inputs.ble_lls_level_1", "inputs.hw_mil
1", "discrete_inputs.2"
"2023-11-30T13:13:14+0600",
54.22809, 69.5264283, 28, 2871, 24296.444, 0, 1
"2023-11-30T13:13:25+0600",
54.228095, 69.5278333, 32, 2871, 24296.536, 0, 1
"2023-11-30T13:13:36+0600",
54.227765, 69.5293916, 39, 2871, 24296.644, 0, 1
"2023-11-30T13:13:46+0600",
54.22744, 69.5310083, 39, 2871, 24296.756, 0, 1
"2023-11-30T13:13:55+0600",
54.227205, 69.5323383, 29, 2871, 24296.847, 0, 1
"2023-11-30T13:13:56+0600", 54.2271866, 69.5324516, 27, \N, \N, \N, \N
"2023-11-30T13:14:00+0600",
54.2270866, 69.5328033, 22, 2871, 24296.881, 0, 1
"2023-11-30T13:14:01+0600",
54.2270433, 69.53286, 23, 2871, 24296.887, 0, 1
"2023-11-30T13:14:02+0600",
54.2269866, 69.5328883, 22, 2871, 24296.893, 0, 1
"2023-11-30T13:14:04+0600",
54.2268766, 69.5328683, 22, 2871, 24296.906, 0, 1
"2023-11-30T13:14:05+0600",
54.2268266, 69.5328266, 23, 2871, 24296.912, 0, 1
"2023-11-30T13:14:13+0600",
54.2263733, 69.5321966, 33, 2871, 24296.977, 0, 1
"2023-11-30T13:14:18+0600",
54.2259866, 69.5318949, 34, 2871, 24297.014, 0, 1
"2023-11-30T13:14:22+0600", 54.2256266, 69.5318, 33, 2871, 24297.065, 0, 1
"2023-11-30T13:14:25+0600",
54.22534, 69.5318866, 35, 2871, 24297.097, 0, 1
"2023-11-30T13:14:31+0600",
54.224835, 69.532085, 33, 2871, 24297.155, 0, 1
"2023-11-30T13:14:42+0600",
54.2238583, 69.5320866, 38, 2871, 24297.264, 0, 1
"2023-11-30T13:14:52+0600",
54.2229033, 69.5321616, 36, 2871, 24297.36, 0, 1
```

```
"2023-11-30T13:15:00+0600",  
54.222275,69.5320816,36,2871,24297.44,0,1
```

### example for searching by server time additionally

This API is designed to accommodate scenarios where you retrieve information from trackers to your applications within specified time intervals. Occasionally, trackers may experience connectivity issues. During such occurrences, these trackers automatically store information in their memory buffers. Upon re-establishing a connection, devices promptly transmit their stored information to the platform.

For instance:

A tracker was connected from 10:00 to 10:30. It then loses GSM signal, storing information in its buffer from 10:30 to 12:00. At 12:00, it reconnects and begins sending packets from the buffer. These packets are timestamped with message times starting from 10:30, 10:31, and so forth. However, the server time reflects 12:00, 12:01, and so on. If your program requests data from 10:00 to 11:00 at 11:00 without utilizing the `server_time_filter` parameter, it will receive messages only from 10:00 to 10:30. The program might not be aware that it needs to re-request this data once all data from the buffer has been uploaded.

To address such situations, there is an optional filtering using the `server_time_filter` parameter. This ensures that your program will get all buffered information. This approach helps prevent potential data gaps and enhances the reliability of your application.

### cURL

```
curl -X 'POST' \  
'https://api.eu.navixy.com/dwh/v1/tracker/raw_data/read' \  
-H 'accept: text/csv' \  
-H 'Content-Type: application/json' \  
-d '{  
  "hash": "6dc7d304dec4434f4c4202ec42817f83",  
  "tracker_id": "123456",  
  "from": "2024-02-03T07:00:00.000Z",  
  "to": "2024-02-03T07:23:59.000Z",  
  "server_time_filter": {"from": "2024-02-03T10:30:00Z", "to":  
    "2024-02-03T12:30:00Z"}  
  "columns":  
    ["lat", "lng", "speed", "inputs.ble_lls_level_1", "inputs.hw_mileage", "dis
```

### errors

- 201 – Not found in the database – tracker ID does not exist.
- 204 – Entity not found – there is no tracker with such ID belonging to authorized user.

- 208 – Device blocked – tracker exists but was blocked due to tariff restrictions or some other reason.

Last update: February 28, 2024







# App: Courier on the map

**Delivery** is a special plugin which can be embedded to any other application or website and allows track user's task by external ID and bounded tracker in the real time.

## Usage

```
https://saas.navixy.com/pro/applications/delivery/?  
key=GENERATED_KEY
```

where key – is a session key generated with API call `/user/session/delivery/create`.

## Parameters

The plugin can be easily customized with the following parameters provided as GET params:

- `performer\_type` – You can use an employee or vehicle label as the tracker marker label. Values: `employee`, `vehicle`, `tracker`.
- `performer\_label` – You can set the custom tracker marker label.
- `external\_id` – The task external ID, specified in task creation/edit form.
- `hide\_task` (1,0) – Hides task. In this mode you can track only the tracker(courier).
- `display\_fields` – You can show only important information in the task info panel. Names of fields are listed through a comma. Fields: label, description, address, period.
- `prompt\_placeholder` – The task external ID prompt placeholder e.g "Order ID"
- `panel\_align` – Specifies the task info panel align. Values: `tl` – Top-Left corner, `tr` – Top-Right corner, `bl` – Bottom-Left corner, `br` – Bottom-Right corner.
- `panel\_scale` – Specifies the task info panel size. Values: `small`, `medium`, `big` – `medium` is the default value.
- `color` – Specifies the task marker and tracker marker color. Values: `FF0000` (red), `FF9900` (orange), `339966` ( green), `3366FF` (blue). `FF9900` (orange) by default.

Available colors: `000000`, `993300`, `333300`, `003300`, `003366`, `000080`, `333399`, `333333`, `800000`, `FF6600`, `808000`, `008000`, `008080`, `0000FF`, `666699`, `808080`, `FF0000`, `FF9900`, `99CC00`, `339966`, `33CCCC`, `3366FF`, `800080`, `969696`, `FF00FF`,

FFCC00, FFFF00, 00FF00, 00FFFF, 00CCFF, 993366, C0C0C0, FF99CC, FFCC99, FFFF99, CCFFCC, CCFFFF, 99CCFF, CC99FF, FFFFFFFF.

## Autoscaling

Autoscaling means that the scale of the map, and the center of the area are automatically selected so that all displayed objects are visible.

`autoscale`:

- `0` – do not scale
- `1` – scale (by default)

## Map scale

The zoom parameter allows specifying map scale by default. Parameter will be ignored with switched on autoscaling.

`map`:

- `roadmap` – Google
- `satellite` – Google satellite
- `osm` – Open Street map
- `doublgis` – 2Gis
- `osmmapnik` – OSM mapnik
- `wikimapia` – Wikimapia
- `mailru` – Mail.ru
- `yandexpublic` – Yandex Public map
- `cdcom` – Progorod

## API for keys

### Authorisation on API

To use the calls described further you have to be authorized in system as it is described according to the link: [API authorization](#)

## Creating a key

Use the following API call to create a new key

```
http://api.domain.com/user/session/delivery/create/?hash=USER\_HASH
```

answer example if the key is successfully generated:

```
{
  "success": true,
  "value": "206831ba32ec9d2a6f7b91b033a48912"
}
```



### Important

Previous key (if you already have got one), will be replaced with the new one. All the links like <http://ui.domain.com/pro/applications/locator/?key=> will not work anymore.

## Retrieving a key

To acquire the key you have created earlier, please use the method

```
http://api.domain.com/user/session/delivery/read/?hash=USER_HASH
```

The reply will look like as follows:

```
{
  "success": true,
  "value": "206831ba32ec9d2a6f7b91b033a48912"
}
```

Last update: January 28, 2022





# App: Web Locator

"Web Locator" is a special plugin which can be embedded to any other application or website and allows track user's objects on the map in real-time.

## Example

The following HTML texts is used to show on the map the objects from [demo account](#):

```
<iframe src="https://saas.navixy.com/pro/applications/locator/?  
key=14084cd4a31f702341afb3fd6f81e475"  
width="900" height="400">  
</iframe>
```

## Usage

To start using the Weblocator user needs to acquire the GENERATED\_KEY value. He or she can copy this value from their private user area in the Web-interface or use [appropriate API call](#). Once user generates the key value, it won't expire and can be used till user generates the newer key.

Insert the following HTML text on any web-page you require using the GENERATED\_KEY value.

```
<iframe src="https://saas.navixy.com/pro/applications/locator/?  
key=GENERATED_KEY"  
width="900" height="400">  
</iframe>
```

## Parameters

You can define window size, choose the background map layer, list the objects to show, use autoscaling to track multiple objects.

All parameters are transferred to the Web locator application by the GET method. For example:

```
?key=613e16fe56f14baa13c676eb9ddceb&width=600&height=400&map=1
```

Width and height of area are set in pixels.

## Objects list

You can limit the list of objects which will be displayed in the Web locator window. All user's account objects are displayed by default.

`names` - names of objects are listed through a comma

or

`objects` - numbers of objects are listed through a comma (tracker\_id)

## Autoscaling

Autoscaling means that the scale of the map, and the center of the area are automatically selected so that all displayed objects are visible.

`autoscale` - 0: do not scale, 1: scale (by default).

## Trace

Traces behind the assets will be shown on the map, as defined by the duration value (in seconds). Disabled by default.

`tail_size`: from 0 to 604800 (one week).

## Map scale

The `zoom` parameter allows to specify map scale by default. Parameter will be ignored with switched on autoscaling.

`zoom`: from 0 to 18

## Map choice

You can define a cartographic substrate



map :

- `roadmap` – Google
- `osm` – Open Street map
- `doublegis` – 2Gis
- `osmmapnik` – OSM mapnik
- `wikimapia` – Wikimapia
- `mailru` – Mail.ru
- `yandexpublic` – Yandex Public map
- `cdcom` – Progorod
- `satellite` – satellite

## API for keys

### Authorization on API

To use the calls described further you have to be authorized in system as it is described according to the link: [API authorization](#)

### Keys Generation

Use the following API call to generate a new key

```
http://api.domain.com/user/session/weblocator/create/?  
hash=USER_HASH
```

Important notice: previous key (if you already have got one), will be replaced with the new one. All the links like `http://ui.domain.com/pro/applications/locator/?key=<old key>` will not work anymore.

Answer example if the key is successfully generated:

```
{  
  "success": true,  
  "value": "206831ba32ec9d2a6f7b91b033a48912"  
}
```

## Acquiring key

To acquire the key generated earlier use the call

```
http://api.domain.com/user/session/weblocator/read/?hash=USER_HASH
```

The reply will look like as follows:

```
{  
  "success": true,  
  "value": "206831ba32ec9d2a6f7b91b033a48912"  
}
```

Last update: January 28, 2022





# Login redirect

There are a number of options to user login page URL, which you can submit as GET-parameters. You may use this feature for providing the links on external resources (e.g. your website) to let your users go straight to the section they need, use some language by default, etc.

## Page section

You can define the section which your users land by default with `partition` parameter:

- `user` – user login page (used by default)
- `demo` – access a chosen demo account
- `register_fast` – quick registration form
- `register_full` – full registration form
- `password_remind` – password reminder

## Language

Use `locale` parameter to define which language will be used:

- `en_EN` – English
- `es_ES` – Spanish
- `ru_RU` – Russian
- etc.

If this parameter is omitted, the language which was set by default for your service will be used.

## Examples

The next code will land user on login section with Spanish language:

```
http://<your_login_page_url>/login/?partition=demo&locale=es_ES
```

The code below lands user on quick registration form with default language set by default:

```
http://<your_login_page_url>/login/?  
partition=quick_register&locale=es_ES
```

Last update: October 31, 2021







## User apps

You can add your own application to the user interface. It will appear as an additional tab in the "Applications" menu. In order for the app to work within the platform, it needs to support iframe feature. If you don't have an iframe option, the app can be added as a separate link, in which case a new browser tab will open when you click on it. Some apps have been developed by our partners and are available in [Marketplace](#). To add them, contact the developer of the application.

**If your domain is using an HTTPS connection, the link to the application must also be HTTPS. Otherwise, you will encounter a mixed content error.**

## Authorization in the application

When you open an application through the Navixy interface, user session hash will be sent to the URL of the application by GET method. This hash can be used for authorization within the application.

## Cookie

By default, the web server sends the following cookies when an external application link opens:

- User session hash as `hash=a6aa75587e5c59c32d347da438505fc3`.
- Locale as `locale=en`.

**If you do not want the server to send cookies, inform technical support and this function will be disabled.**

## How to add an application


### Cloud version

Contact [Navixy technical support](#) and specify the following parameters:

- Application name.

- External URL link.
- Opening method - iframe or a new tab.
- Installation destination - user\_id or panel\_id.
- Cookies sending - user session hash and/or locale or nothing.

Our specialists will do everything necessary, and the application will be available in the user interface. The application can be installed for all users or for a specific one.

 **If the app will be installed to the specific user, please contact the support team every time you need to add this app to another user. If the app installed for whole panel - all new users will automatically get the app. Also, the app can be installed to whole panel instead of specific users.**

## Standalone version

You can find the instruction on installation of the software [here](#).

Last update: February 26, 2024

