Welcome to B2Field Developer Documentation

B2Field is a Feld Service Management software developed by SquareGPS company. Here you can find information about the integration of 3rd party solutions with the B2Field platform, API, and technical documentation for developers and partners.

Last update: September 16, 2020

Getting started

How to read this documentation

Coming soon.

Get involved

You can really help to improve this documentation o localizations of B2Field 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: September 10, 2020

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.

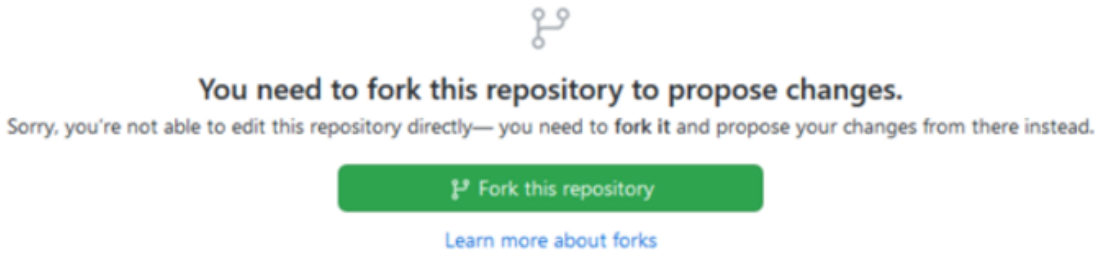
There are several ways:

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 multiply commits before creating a pull request.
4. Installing and editing documentation locally on yours PC.

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

Easy way

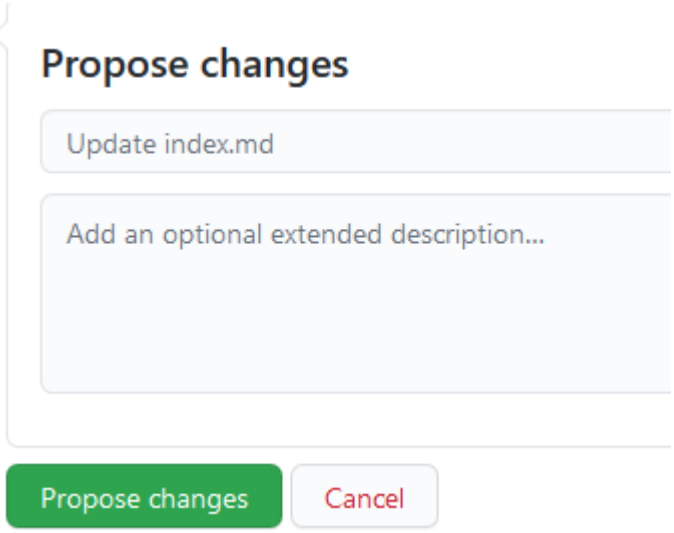
On each page in the upper right corner of the text top there is a link with a picture of a pencil :material-pencil:. After clicking on this link, you will be asked to create a fork of the repository (if you have not done this before).



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.



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 yours 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 B2Field 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

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

Resource description.

Resource specific actions:

* [/path/to/resource/method1](#method1)
* [/path/to/resource/method2](#method2)

## method1

Method description.

### Parameters

| name  | description | type  | restrictions |
| :---- | :----       | :---- | :----        |
|param1 | description | int   | [1..100], not null |

### Examples

=== "cURL"

    ```shell
```
```

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

=== "HTTP GET"

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

### Response

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

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

### Errors

Special error codes.

## method2

...
````

For real example see /user and source.

Last update: December 17, 2020
````

Contacts

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

You can call us or send email: b2field.com/contact.

Follow us in the social networks:

- GitHub
- https://www.facebook.com/B2Field/
- Instagram

Last update: September 10, 2020

Languages

B2Field 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 B2Field products both left-to-right and right-to-left languages are supported. The following B2Field 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 B2Field products to a new language or improve the existing language packs.

Last update: September 10, 2020

Translate B2Field

Localizing B2Field 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

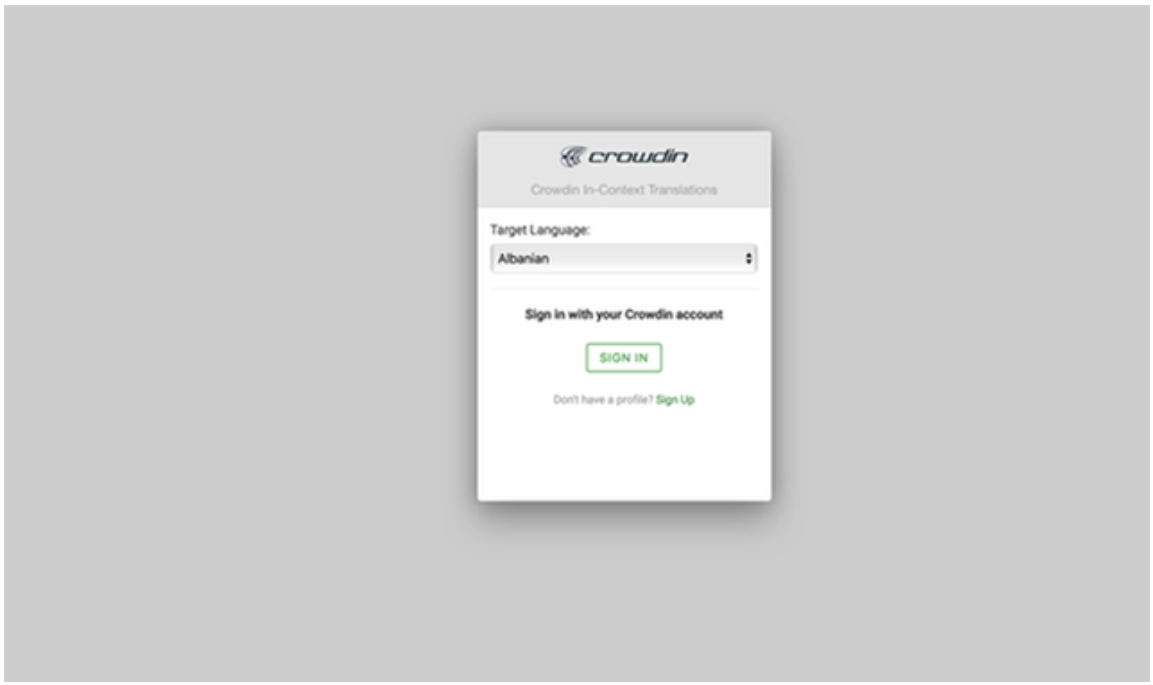
Crowdin In-context translation (only Web UI)

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:

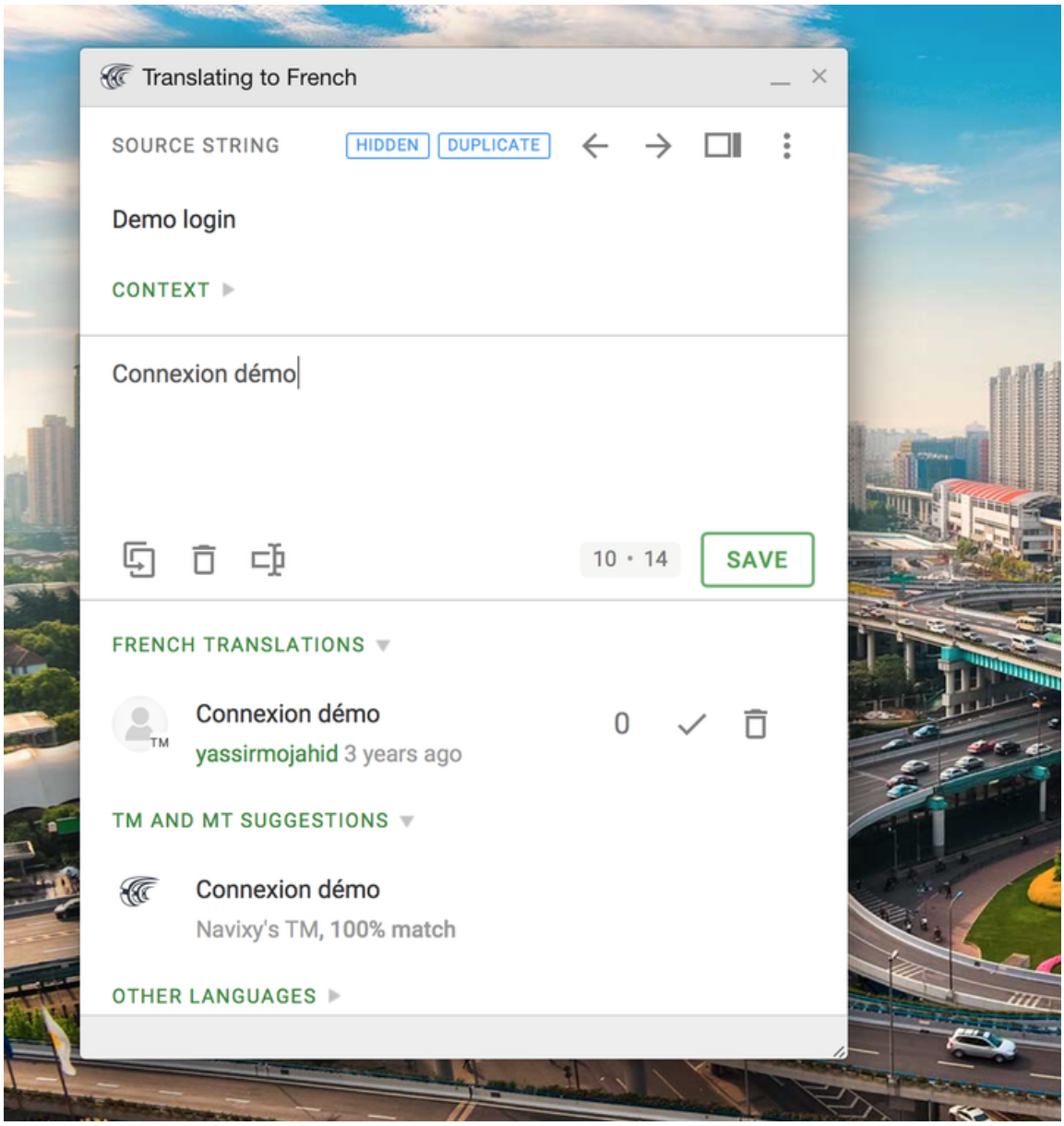
```
https://demo.navixy.com/?locale=ach#/login
```

You should see crowdin authorization dialog.



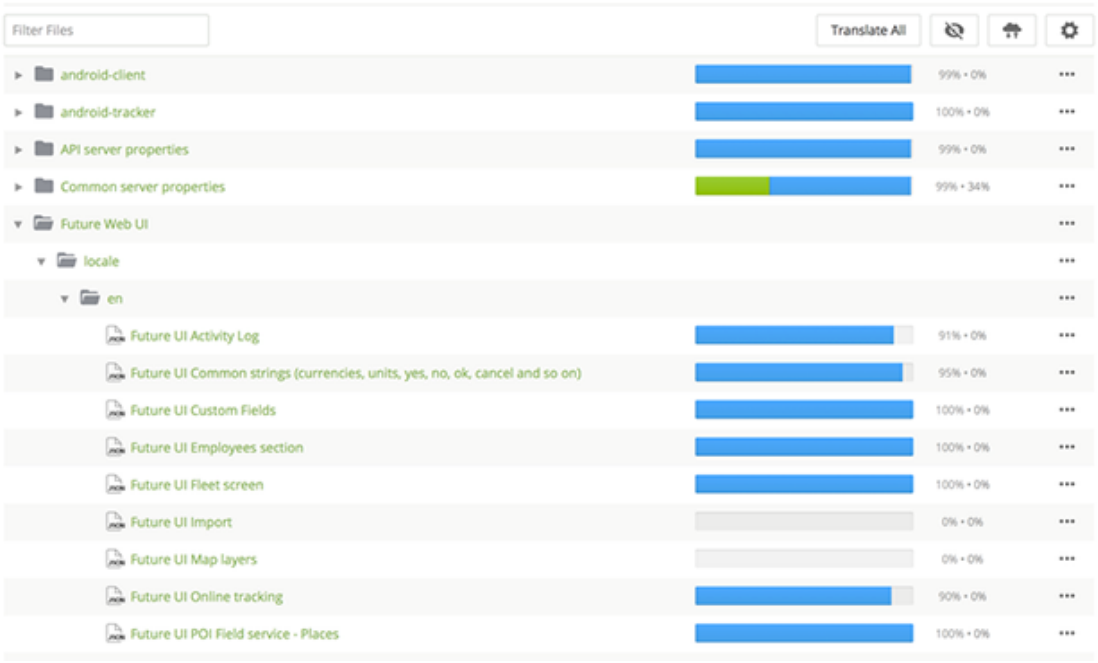
After authorization standard Navixy UI will appear in a special translation mode. Click on a little icon near each text item

opens translation dialog



Translate via Crowdin UI

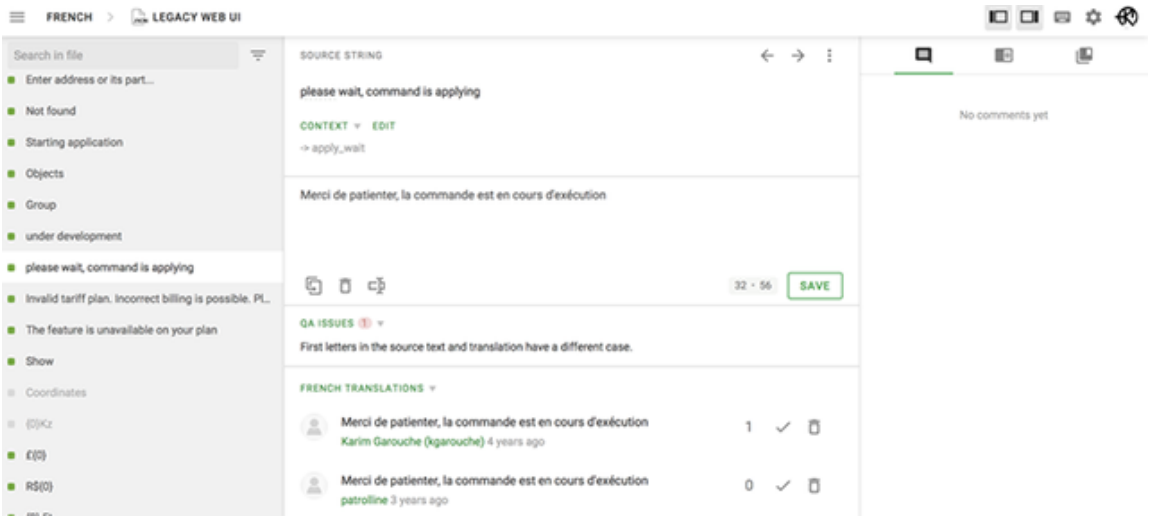
Crowdin UI is a most powerfull way to work with translations in Navixy and the only way if you want to translate not Backend and Mobile apps.



Translations in crowdin organized into several directories:

1. Common Server properties, API server properties, Tracking server properties, SMS server properties - translation strings for backend (Mainly for Reports, SMS and Email notifications)
2. Future Web UI, Legacy Web UI - translation strings for Navixy Web UI.
3. android-client, android-tracker, navixy-retracker-ios, navixy-viewer-ios - translation strings for mobile apps.

Each entry in each directory contains strings for translation, displayed in translation screen



If you have some problems with translation feel free to ask questions in comments.

Translations delivery

Usally 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 17, 2020

Postman

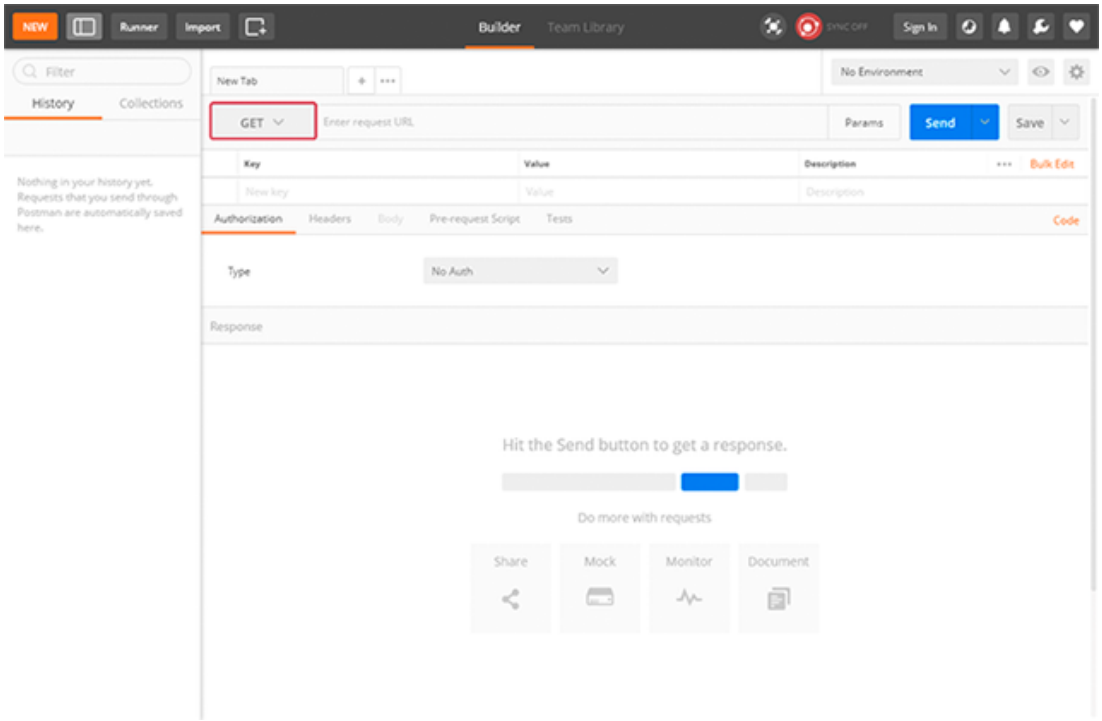
There are many tools that could be used to work with API requests. From simple input to browser's address line or cmd tool to more complex software. One of our personal favorites is Postman application. Postman is a collaboration platform for API development. It can be used for a variety of purposes 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

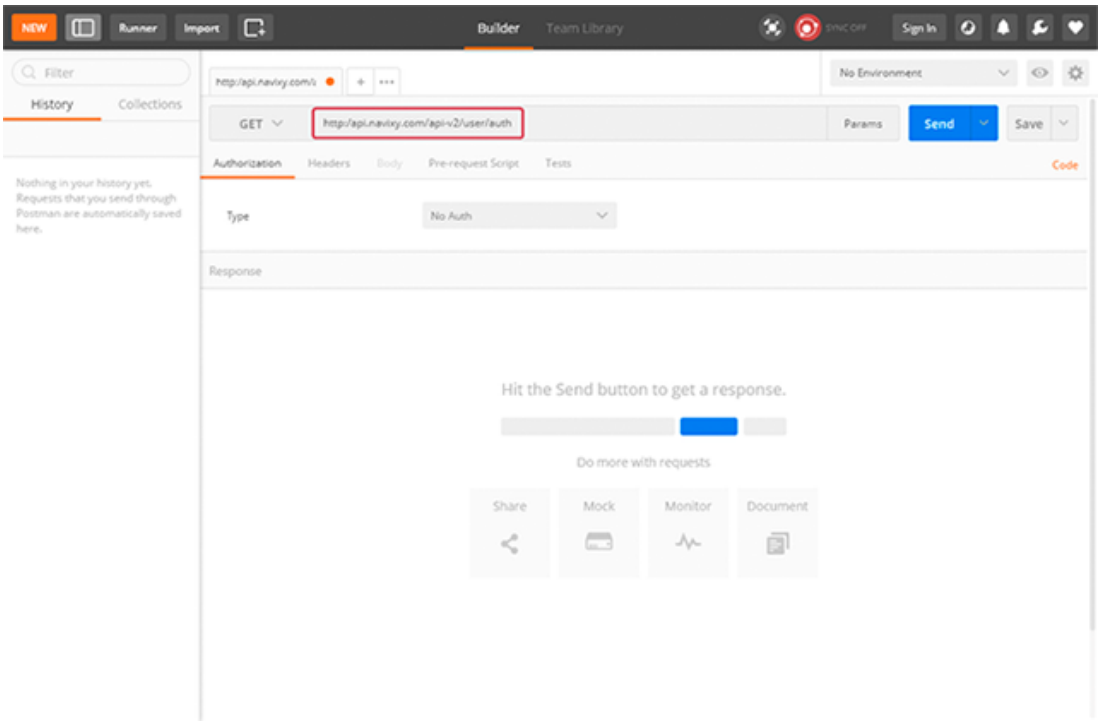
Postman API client allows you to easily send various API requests and helps you fill out parameters without worrying that you will miss a quote or bracket. This can be especially handy when working with large requests.

1. Select a request method:

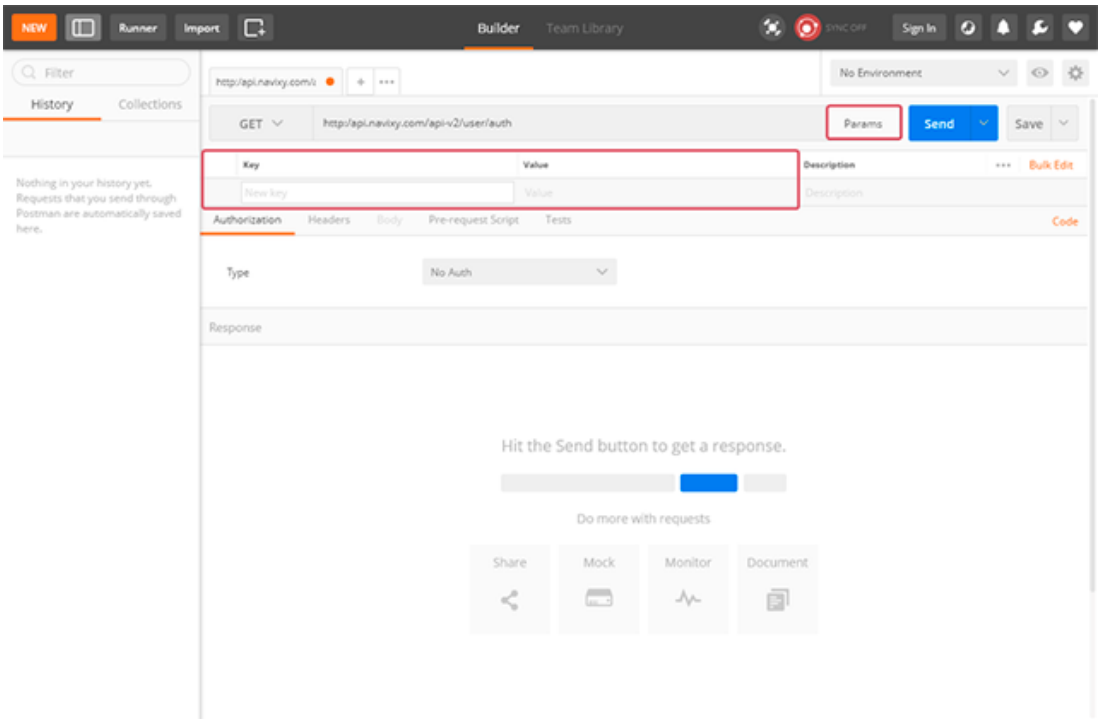


   Each API request uses an HTTP method. The most common methods for B2Field API are GET and POST. GET methods retrieve data from an API. POST sends new data to an API.

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/fsm/`
    - For US server - `https://api.us.navixy.com/v2/fsm/`

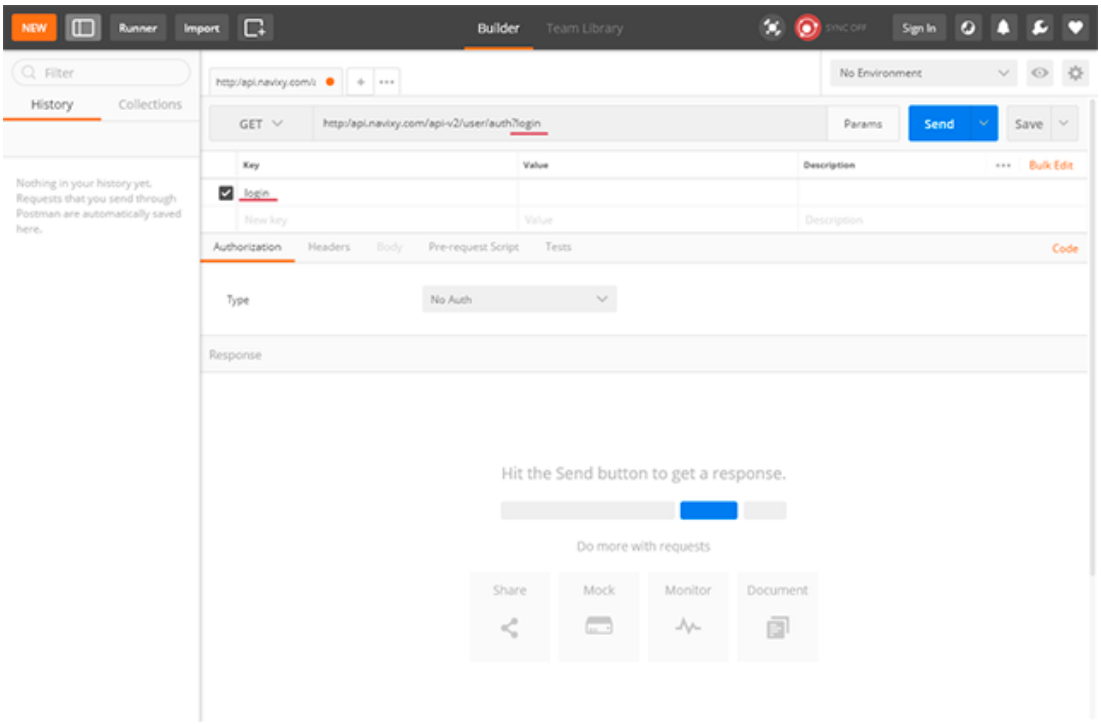


3. Click on the Params button, and you will see a table for key and value input:

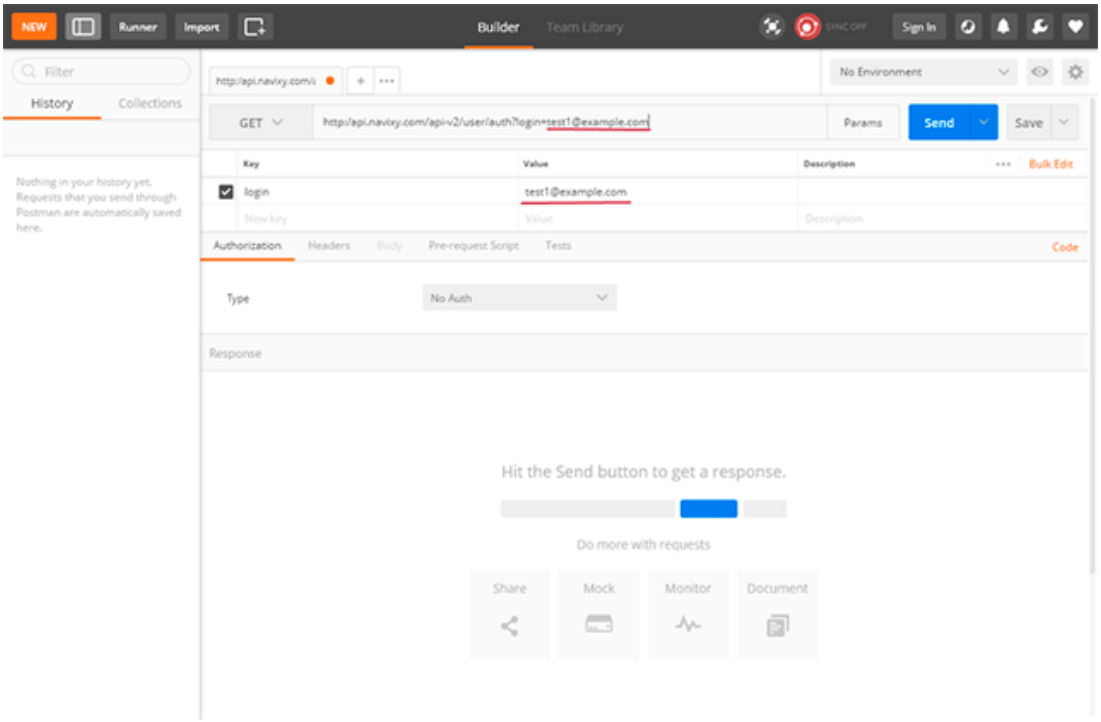


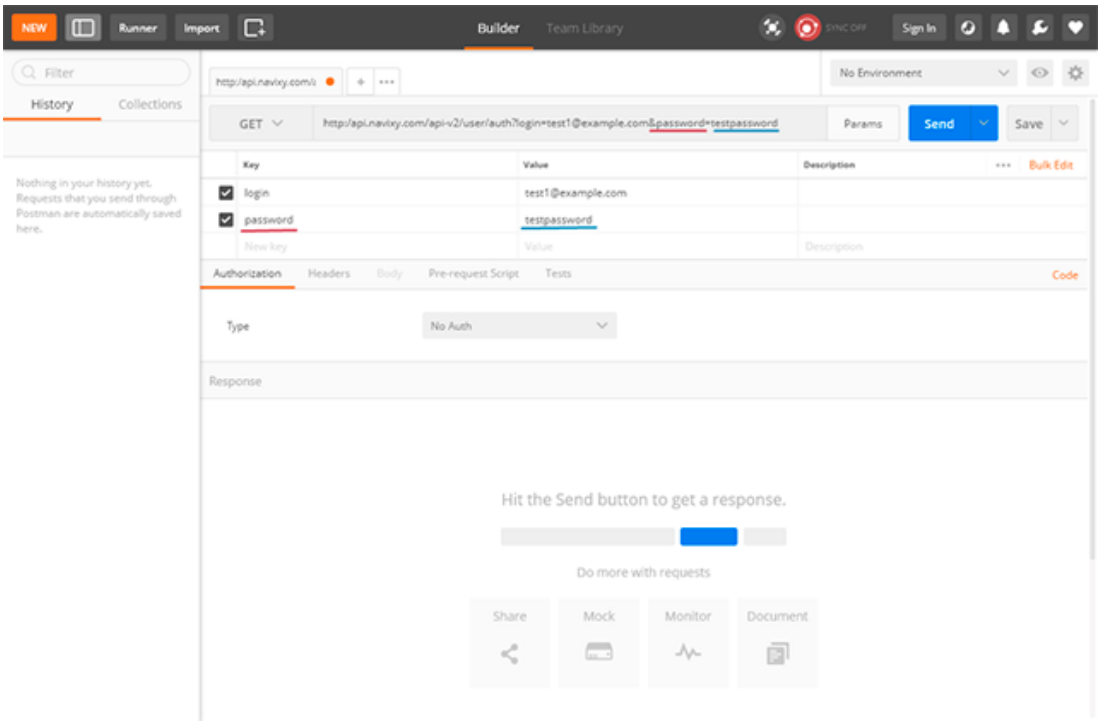
We will only ever need to fill 2 fields - Key (parameter name from documentation) and value. For user/auth request, we have 2 keys that should be transmitted - login and password.

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

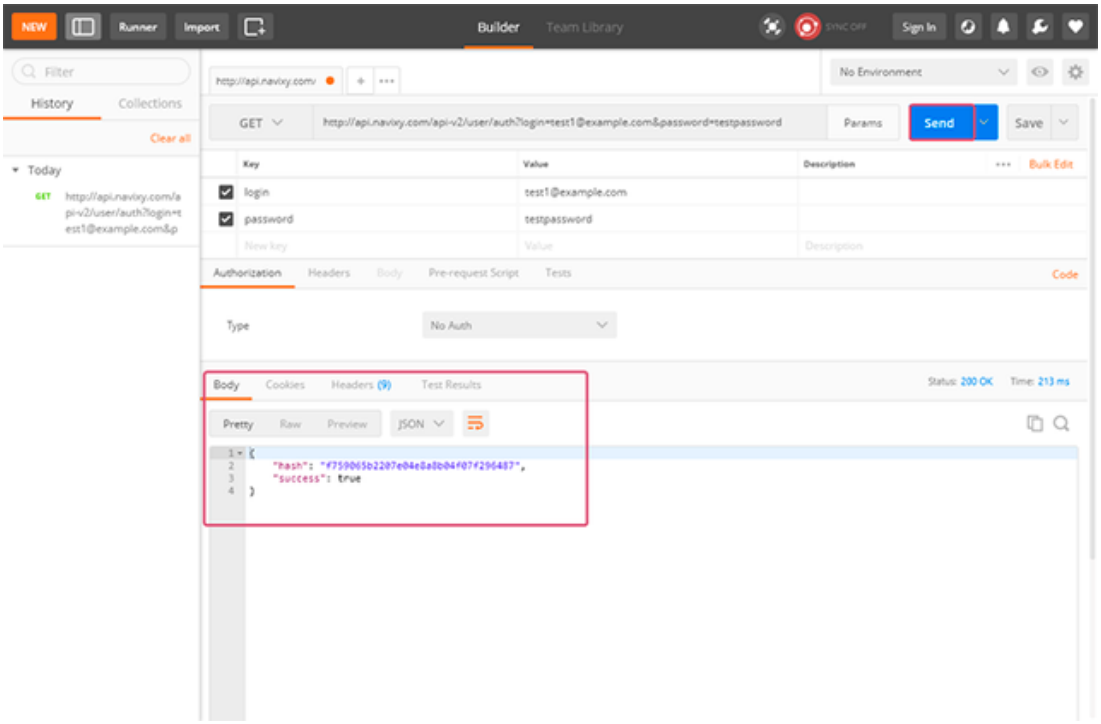


Similarly, with values and additional parameters:



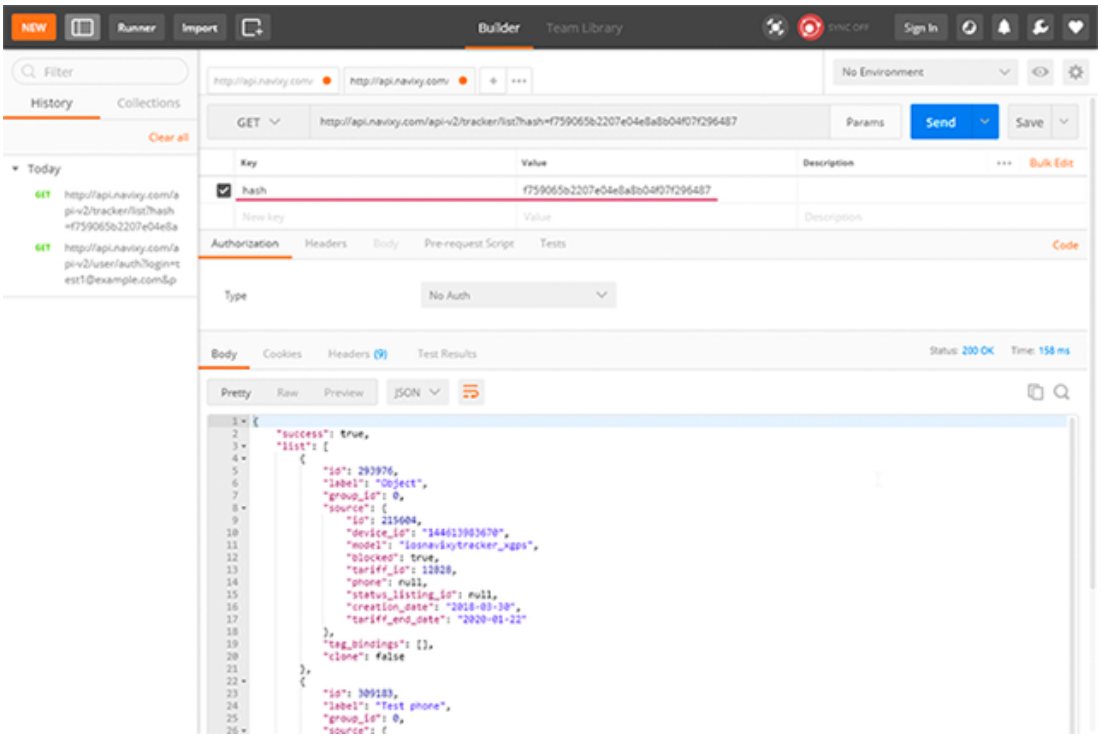


4. Press send, and you will see the reply, already split and highlighted for easier reading



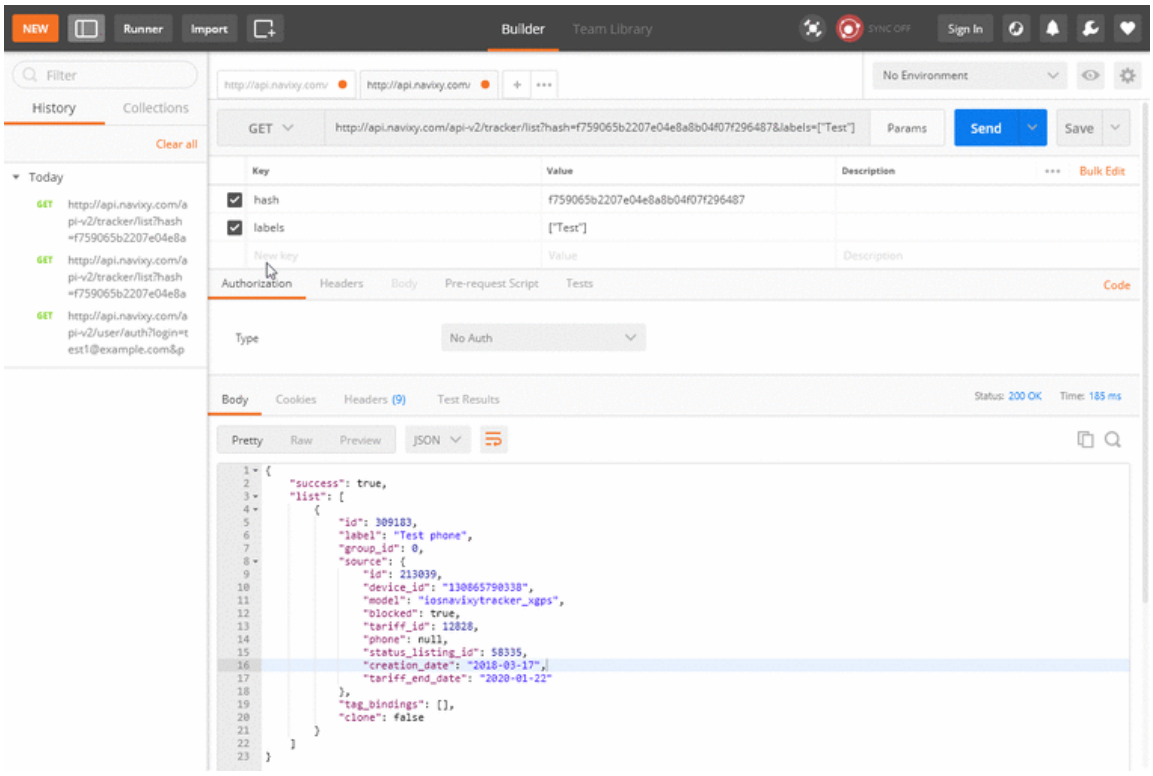
In this case, we have received a hash that should be copied and user for future requests.

Example: tracker/list request



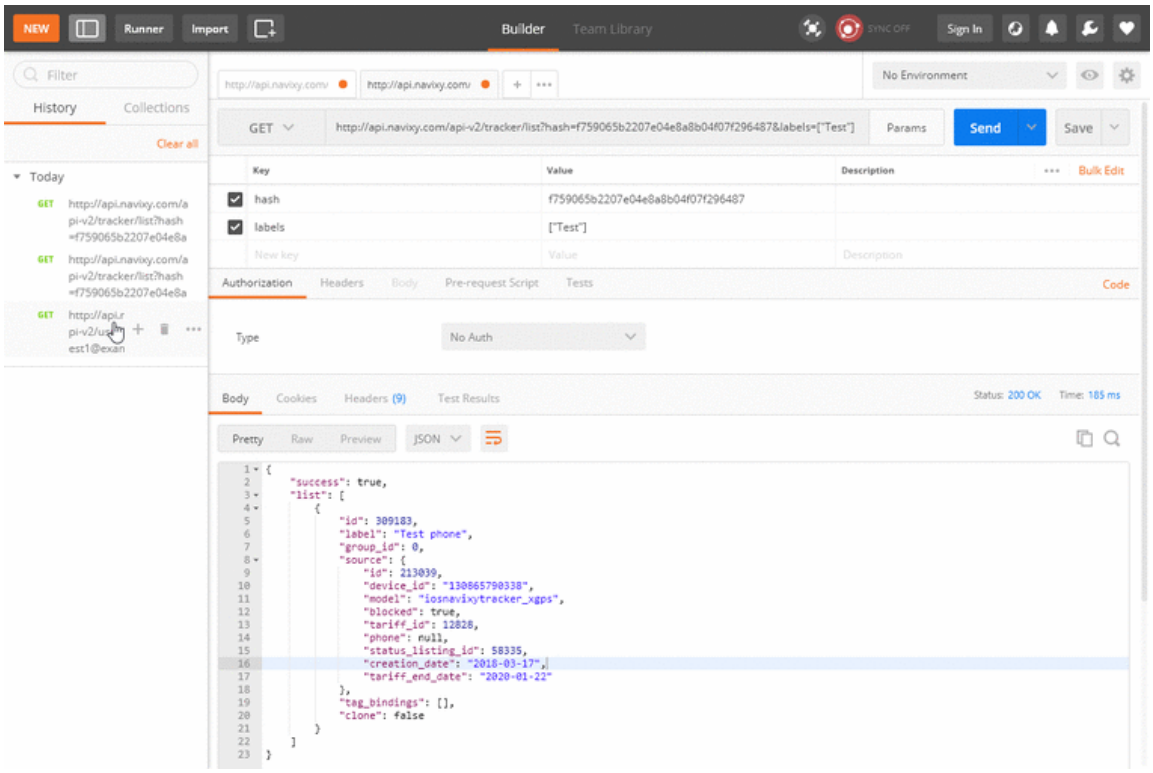
Working with parameters

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



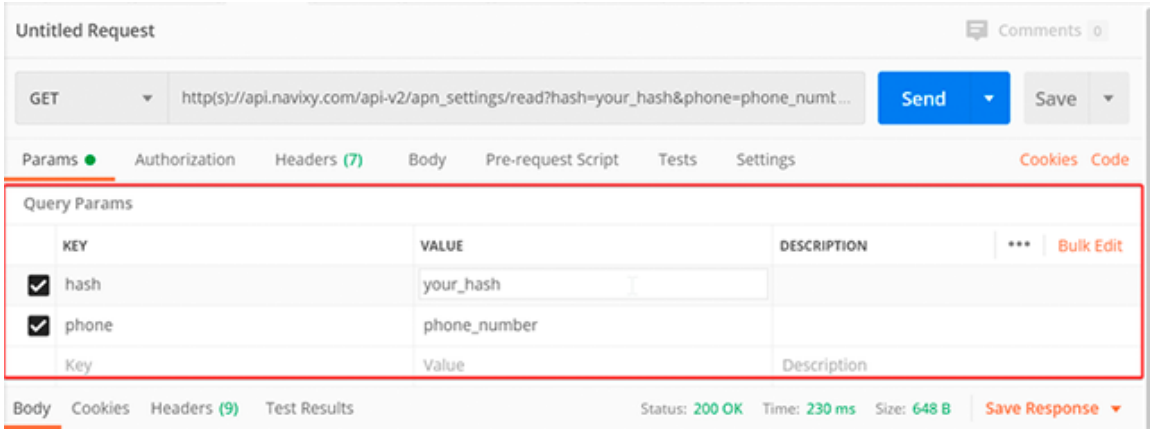
History of requests

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 editing



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 17, 2020

B2Field 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's describe standard workflow for API developer using very simple and most common example — requesting the track points data:

1. Determine URL to API calls.
2. Authorize with `user/auth`. This API method will return the hash you should use for all your next API calls.
3. Get objects lists with `tracker/list`.
4. Get track lists with `track/list`.
5. Get the track itself: `track/read`.

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/fsm` for European B2Field ServerMate platform.
- `https://api.us.navixy.com/v2/fsm` for American B2Field ServerMate platform.
- `https://api.your_domain/fsm` for the self-hosted (On-Premise) installations.

For example, to make `user/auth` API call on the European B2Field ServerMate, you should use the URL:

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

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=value' \
  -d 'hash=a6aa75587e5c59c32d347da438505fc3'
```

HTTP GET

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

Hash 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": "888888888888888"
      },
      "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": "333333333333333"
      },
      "tag_bindings": [
        {
          "tag_id": 9,
          "ordinal": 3
        }
      ],
      "clone": false
    }
  ]
}
```

Or error if hash is wrong:

```
{
  "success": false,
  "status": {
    "code": 4,
    "description": "User 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 user session hash (A String containing 32 hexademical 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 pseudogrammar that illustrates the construction of the `Authorization` request header:

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

Session hash can be obtained via `user/auth` API call:

cURL

```
$ curl -X POST '[api_base_url]/user/auth' \
       -H 'Content-Type: application/json' \
       -d '{"login": "demo", "password": "demo"}'
```

GET

This method is not recommended. Just for example:

```
[api_base_url]/user/auth?login=demo&password=demo
```

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 `yyyy-MM-dd HH:mm:ss` format (in user's timezone).
- `local_time` – is a string containing local time in `HH:mm:ss` format.
- `location` – is json object contains geographical coordinates, e.g.

  ```
  {"lat": 56.827001, "lng": 60.594296}
  ```

- `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`.

Constants

- **maxHistoryLimit** = 1000 – maximum count of history entries from listing requests
- **maxReportTimeSpan** = 120 days – maximum interval in for most requests

Error handling

If an error occurs, API returns special error response. You can also detect error by checking HTTP response code. If it's 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 user hash	
4	User not found or session ended	
5	Wrong request format	
6	Unexpected error	500
7	Invalid parameters	

code	description	HTTP code
8	Queue service error, try again later	503
9	Too large request	412
11	Access denied	403
12	Dealer not found	
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

code	description	HTTP code
209	Failed sending email	
210	Geocoding failed	
211	Requested time span is too big	
212	Requested limit is too big	
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	

code	description	HTTP code
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	
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 doesn't 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

code	description	HTTP code
248	Wrong password	
249	Operation available for clones only	403
250	Not allowed for deleted devices	403
251	Insufficient funds	403
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

code	description	HTTP code
267	Too many entities	
268	Over quota	402
269	Invalid file state	
270	Too many sensors of same type already exist	
271	File over max size	413

Last update: December 17, 2020

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's section has step by step examples of working with B2Field API.

From initial step of obtaining a session key 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 session hash
- How to get tracker list
- How to get track points
- How to obtain report's information

Last update: December 17, 2020

Obtaining session hash

"Hash" or "Session key" is a randomly generated string that is used to verify and authenticate actions. Nearly all API calls require a session key to operate.

To get hash use the user/auth call with credentials of a user:

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

The response will be like this:

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

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

Received hash should be saved and used in API calls. For security reasons, hash has a lifetime of 30 days and will expire in certain situations:

- After 30 days.
- User has changed their password.
- User logged out and ended the session.
- User was deleted.

Correct work with hash is crucial. There is no need to receive a new hash before each request, instead, your hash should be stored and reused. To prevent expiration, in most cases you just need to prolong the session.

To prolong the session, use the following API call:

```
https://api.navixy.com/v2/fsm/user/session/renew?hash=you_hash
```

You can disable the current hash with the following call:

```
https://api.navixy.com/v2/fsm/user/logout?hash=your_hash
```

Using hash

You must pass session hash with most API calls along other parameters required to make the call.

For example, if you want to make a call with the single parameter `id` equal to 1, and you obtained hash equal to `882fb333405d006df0d5a3f410115e92` (fake hash, just for example) you must pass the following parameters in HTTP request: `id=1&hash=882fb333405d006df0d5a3f410115e92` (example is for POST/ `x-www-form-urlencoded` or GET requests).

Otherwise, you will get an error response:

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

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

If session expired or was logged out, you will receive the following response:

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

It means that you need to obtain new user hash through user/auth API action.

Last update: October 1, 2020

How to get tracker list

Now we have a hash — let's start with essential basics.

B2Field 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/fsm/tracker/list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

HTTP GET

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

Last update: December 17, 2020

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 get hash.

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 in `yyyy-MM-dd HH:mm:ss` format (in user's timezone).
- `to` - a string containing end date/time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone).

Optional parameters:

- `track_id` - we can get them using track/list API call. If specified, only points belonging to the specified track will be returned. If not, any valid track points between `from` and `to` will be returned. All requested track ids must be unique and not null.
- `include_gsm_lbs` – boolean. It may contain `true` or `false`. If `false` && `track_id` not specified, GSM LBS points will be filtered out. It is `true` by default.
- `point_limit` – int. If it specified, the returned track would be simplified to contain this number of points. Min=2, Max=3000.
- `filter` – boolean. If `true`, the returned track will be filtered, applicable only for LBS tracks. It is `false` by default.

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` - string 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 -X POST 'https://api.navixy.com/v2/fsm/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 identical with track/read with the except of 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: October 1, 2020

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 in `yyyy-MM-dd HH:mm:ss` format (in user's timezone). Data in a report will be from that moment.
- `to` - A string containing date/time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone). 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/fsm/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/fsm/report/tracker/
retrieve' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "report_id":
"1234567"}'
```

HTTP GET

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

You will get the report in a JSON format:

Example


```
{
"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/fsm/report/tracker/delete'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "report_id":
"1234567"}'
```

HTTP GET

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

Last update: November 16, 2020

Bill

API path: `/bill`.

create

Creates new bill for the user. Required subuser rights: `payment_create`.

parameters

name	description	type	restrictions
payer	some payer description	string	Payer Name
sum	bill sum in default currency of the panel	int	1000

example

```
https://api.navixy.com/v2/fsm/bill/create?
hash=22eac1c27af4be7b9d04da2ce1af111b&payer=John Doe&sum=500
```

response

```
{
    "success": true,
    "value": 6421     // 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 subuser rights: payment_create

structure:

```
https://api.navixy.com/v2/fsm/bill/list?
hash=your_hash&limit=number_of_bills&offset=start_from
```

parameters

name	description	type	format
limit	maximum number of bills in list (maximum and default **10 000**) - optional	int	10000
offset	get bills starting from **offset** (default **0**) - optional	int	0

example

```
https://api.navixy.com/v2/fsm/bill/list?
hash=22eac1c27af4be7b9d04da2ce1af111b&limit=9500&offset=0
```

response

```
{
    "success": true,
    "count": 7,       // total number of bills
    "bills": [${bill}, ...]
}
```

where **bill** is

```
{
    "order_id": 63602,                  // unique id
    "created": "2012-03-05 11:55:03",   // creation date/time
    "sum": 150.0,                       // bill sum in rubbles
    "status": "created",                // bill order status
    "positions": ["The subscription fee for the services of Gdemoi
Account W3"],  // list of position names.
                                        // usually contains one
element for bill
    "link": "http://bill.b2field.com/xK1QEYK" // url to order
}
```

If bill created using /bill/create call then **positions** will contains exactly one element.

status may be one of:

- created – but not settled
- settled
- canceled

Note 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: September 10, 2020

Payment system

API path: `/payment_system`.

list

Return list of payment systems available for user.

required subuser rights: payment_create

response

```
{
    "success": true,
    "list": [<payment_system_settings>, ...]
}
```

where **payment_system_settings** is:

```
{
    "type": "rbkmoney", // payment system type
    "url": "https:_rbkmoney.com/acceptpurchase.aspx", // URL to
send payment info,
    "account": <string>, // (optional) dealer account in payment
system (eshopId for RBK)
    "currency": "EUR", // 3-letter ISO 4217 currency code
    "payment_code": "B2Field Demo", // (optional) code for payments
    "subscription_code": "4671292", // (string) subscription code.
same as "payment_code" for 2Checkout (formerly Avangate) but for
subscriptions
    "methods": [<string>, ...] // (optional) list of available
payment methods (may be empty)
    // for type == "ios_inapp" only:
    "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
    }
}
```

errors

- 201 – Not found in database.

estimate/get

Returns the estimate of the monthly payment amount

required subuser rights: payment_create

response

```
{
    "success": true,
    "value": 400.0 // payment amount, rounded up to hundreds for
rubles or to tens for other currencies
}
```

mobile/pay

Create bill using 'mobile' payment system (AKA Qiwi Bank)

required subuser rights: payment_create

parameters

name	description	type
phone	10-digit phone number without country code (e.g. 6156680000)	String
sum	amount of money to pay, e.g. 100.50 . minimum is 1.00, maximum is 99999.00	double

response

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

errors

- 13 – Operation not permitted. (if this payment system is not enabled for user's PaaS platform)
- 201 – Not found in database. (if payment system was not configured properly)
- 215 – External service error (if QIWI payment gateway returned an error)

Last update: December 17, 2020

Subscription

API path: `/subscription`.

Payment subscriptions

/subscription/avangate/

Working with 2Checkout (formerly Avangate) subscriptions (renewals).

cancel

Unsubscribe from auto-renewal by reference.

required subuser rights: payment_create

parameters

- **reference** - **string**. internal 2Checkout (formerly Avangate) subscription code. Get it from list call.

response

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

errors

- 215 – External service error

list

List active 2Checkout formerly Avangate subscriptions (renewals).

required subuser rights: payment_create

parameters

no parameters

response

```
{
    "success": true,
```

```
    "list": [
        {
            "reference": "5EAD4B0B2F" // pass it to /subscription/
avangate/cancel
            "code": "4679109" // 2Checkout (formerly Avangate)
product code
            "quantity": 123 // count
            "expiration_date": "2016-03-10 13:32:11" // next renew
date/time
        },
        ...
    ]
}
```

errors

- 215 – External service error

Last update: August 21, 2020

Transaction

list

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

required subuser rights: payment_create

parameters

- **from** – date/time. Start date/time for searching.
- **to** – date/time. End date/time for searching. must be after "from" date.
- **limit** – int (optional). Maximum number of returned transactions.

response

```
{
  "success": true,
  "list": [
     {
        "description": ,  // transaction description, e.g.
"Recharge bonus balance during tracker registration"
        "type": ,         // type, e.g. "bonus_charge"
        "subtype": ,      // subtype, e.g. "register"
        "timestamp": , // date/time at which transaction was
created, e.g. "2013-08-02 08:16:40"
        "user_id": ,         // user Id, e.g. 12203
        "dealer_id": ,       // dealer Id, e.g. 5001
        "tracker_id": ,      // tracker id, e.g., 3036, or 0 if
transaction is not associated with tracker
        "amount": ,       // amount of money in transaction, can
be negative. e.g. -10.0000 means 10 money units were removed from
user`s balance
        "new_balance": ,  // user`s money balance after
transaction, e.g. 800.0000
        "old_balance": ,  // user`s money balance before
transaction, e.g. 810.0000
        "bonus_amount": , // amount of bonus used in transaction,
can be negative. e.g. 10.0000 means 10 bonuses units were added to
user`s bonus balance
        "new_bonus": ,    // user`s bonus balance after
transaction, e.g. 10.0000
        "old_bonus":      // user`s bonus balance before
transaction, e.g. 0.0000
     }, ...
  ]
}
```

errors

- 211 – Requested time span is too big (more than **maxReportTimeSpan** config option)

Last update: October 23, 2020

Tariff

API path: `/tariff`.

Tariff JSON object structure:

```
{
    "id": 10, // (int) unique id
    "name": "Business", // (string) tariff description
    "group_id": 2, // (int) group of tariffs. user can change the
tariff only on the tariff in the same group.
    "active": true, // (boolean). user can change the tariff only
on the active tariff.
    "type": "monthly", // (string). tariff type. one of:
"monthly", "everyday", "activeday"
    "price": 13.0, // (double). price per month for "monthly" and
"everyday" tariff or price per "active" day for "activeday" tariff
    "early_change_price": 23.0, // (double) price of change tariff
from current to other
            // 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": 1000, // (int) maximum number of devices per
account
    "has_reports" : true // (boolean) true if reports are allowed,
false otherwise
    "paas_free": false, // (boolean) true if this tariff is free
for PaaS owner, false otherwise
    "store_period": "12m", // data storage period, e.g. "2h" (2
hours), "3d" (3 days), "5m" (5 months), "1y" (one year)
    "features": [
        "map_layers"
    ],
    "map_filter": {
        "exclusion": true,
        "values": []
    }
}
```

list

Get list of device's tariffs available to user.
If user's dealer if **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

- **device_type** – (string) one of 'tracker', 'camera' or 'socket'.

response

```
{
  "success": true,
  "list": [${tariff}, ...] // list of JSON objects
}
```

See **tariff** object structure here.

Last update: October 23, 2020

Tariff tracker

API path: `/tariff/tracker/`.

User of **dealer** can switch tracker from 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. new tariff must be differ from current.
4. **t1.dealer_id** = **t2.dealer_id** = **dealer.effectiveDealerId**, i.e. current and new tariffs must belongs 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. 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 elsewise

errors

- 201 – Not found in 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 tracker.tariff_id and belongs to user's **effective dealer**)

change

Change tariff of tracker (with **tracker_id**) to new tariff (with **tariff_id**).

required subuser rights: admin (available only to master users)

response

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

errors

- 221 (Device limit exceeded) – when new tariff device limit is less then count of trackers in cabinet.
- 238 (Changing tariff is not allowed) – user can't switch tracker to that tariff.
- 239 – New tariff doesn't exist.
- 240 (Not allowed to change 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 passed tracker (even when tariff last changed less or equal than **tariff.freeze.period** time ago).

parameters

- tracker_id

response

```
{
  "success": true,
  "list": [${tariff}, ...],
  "days_to_next_change": ${int} // days to next free change, or 0
if free change available.
}
```

See **tariff** object structure here.

Last update: October 23, 2020

Base

API path: `/base`.

nothing

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

example

```
https://api.navixy.com/v2/fsm/base/nothing?
hash=22eac1c27af4be7b9d04da2ce1af111b
```

response

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

send_email

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

structure:

```
https://api.navixy.com/v2/fsm/base/send_email?
hash=your_hash&from=sender_mail&to=recipient_mail&title=text_title&mes
```

parameters

name	description	type	format
from	from email address	string	from@mail.com
to	to email address	string	to@mail.com
title	title of the email	string	example title
message	text of the email	string	example message
service_id	service parameter	int	1

name	description	type	format
service_pass	service parameter	int	1

example

```
https://api.navixy.com/v2/fsm/base/send_email?
hash=22eac1c27af4be7b9d04da2ce1af111b&from=b2field@mail.com&to=user@ma
```

response

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

Last update: September 10, 2020

Data

/data/spreadsheet/parse

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

```
{
    "file_id": <string, unique file id>,
    "header": <optional, array of string>,
    "preview":<array of array of string, first N rows of file>
}
```

parameters

name	description	type
file	File to upload	File
preview_count	size of preview, min=1, max=20	Integer
parse_header	parse first row as header?	Boolean
header_map	if parse_header is true should contains map of matching column name to field identifier, `{"Label": "label", "Latitude": "lat"}`	Object

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

response

```
{
    "file_id": <string, unique file id>,
    "header": <optional, array of string>,
    "preview": <array of array of string, first N rows of file>
}
```

errors

234 – Invalid data format.

Last update: October 23, 2020

Dealer

API path: `/dealer`.

get_ui_config

Gets dealer info and dealer-specific UI settings by domain.

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

structure:

```
https://api.navixy.com/v2/fsm/dealer/get_ui_config?
domain=your_domain
```

parameters

name	description	type	format
domain	dealer’s monitoring interface domain, e.g. “b2field.com“	string	b2field.com

example

```
https://api.navixy.com/v2/fsm/dealer/get_ui_config?
domain=b2field.com
```

response

```
{
    "success": true,
    "dealer": {
        "id": 5001,                     // int. dealer id
        "ui_domain": "demo.b2field.com", // Dealer's UI domain
        "company_url": "b2field.com"     // Dealer's promo site URL
                                        // e.g. "http://
www.b2field.com" or "demo.b2field.com"
    },
    "settings": {         //may be null if dealer has not set any
custom settings
        "domain" : "demo.navixy.com",   // same as dealer.ui_domain
        "service_title": "Navixy Demo", // Title of the service
        "locale": "at_AT",              // default locale of the
dealer
        "demo_login": "demo",           // dealer's login for demo
```

```
user
                                        // (or empty string if no
demo user available)
        "demo_password": "demo",        // dealer's password for
demo user
                                        // (or empty string if no
demo user available)
        "maps": ["roadmap", "osm"],     // list of available
maps,
                // e.g. ["roadmap", "cdcom", "osm", "wikimapia",
"yandexpublic", "hybrid", "satellite"]
        "default_map": {  //default map settings
            "type": "roadmap",          // default map type
            "location": { //default map center location
                "lat": 57.0,            // latitude
                "lng": 61.0             // longitude
            },
            "zoom": 10                  // default map zoom level
        },
        "currency": "EUR",              // dealer's currency ISO
4217 code
        "payment_link": "http://site.de/pay.php", // PaaS-
dependent link that can be used
                                       // to refill user's
account. Can be null or empty.
        "promo_url": "http://site.de/about/",            //
customizable "About company" url
        "google_client_id": "clientID", // client id which which
must be used to work with google API or null
        "favicon": "paas/5001/custom.ico", // path or url to
dealer's interface favicon
        "logo": "paas/5001/logo.png",   // path or url to dealer's
logotype
        "app_logo": "paas/5001/app_logo.png",       //nullable,
path or url to dealer's mobile app logotype,
        "login_wallpaper": "paas/5001/login.png", // path or url
to dealer's interface login wallpaper
        "desktop_wallpaper": "http://test.com/test.jpg", // path
to dealer's interface wallpaper or null
        "monitoring_logo": "http://test.com/test.jpg", // path to
dealer's interface monitoring logo or null
        "login_footer": "All rights reserved.", // footer which
will be included in login page.
        "allow_registration": true,     // if true then
registration is available for dealer's users
                                        // all html special chars
escaped using HTML entities.
        "show_mobile_apps" : true,      // if true then mobile
applications are available for dealer's users
        "show_call_notifications" : true,      // if true then
call notifications are available for dealer's users
        "default_user_settings": {
            "geocoder": "google",             // default geocoder
            "route_provider": "progorod",     // default router
            "measurement_system": "metric",  // measurement system
            "translit": false
        },
```

```
        "display_model_features_link" : true, // when true show in
model info link to squaregps.com (UI option)
        "color_theme": "aqua",          // (string) color theme
code or empty string (for default theme)
        "app_color_theme": "blue_1",    // (string. 128 chars max)
mobile app color theme code or empty string (for default theme)
        "privacy_policy_link": "http://privacy-policy-url",
        "tos": "Terms Of Service text",
        "enable_trackers": true,        // if true, GPS monitoring
interface is available for dealer's users
        "enable_cameras": false,        // if true, camera
monitoring interface is available for dealer's users
        "tracker_model_filter": {       // a filter which
describes tracker models available for registration
            "exclusion": true,          // in this example all
models available
            "values": []
        },
        "internal": {  // additional options
            "light_registration": true,           // use "very
simple" registration with demo tracker
            "demo_tracker_source_id": 14,         // id of tracker
created on 'light_registration'
            "demo_tracker_label": "Demo tracker", // label of of
tracker created on 'light_registration'
            ...
        },
        "no_register_commands": false // if true then do not send
commands to devices on activation
    },
    "demo_ends": "2014-01-01",          // a date when demo for
this PaaS ends.
                                        // Is null when PaaS is
not on demo tariff
    "premium_gis": true,                // true, if dealer has
Premium GIS package
    "features": ["branding_web"]        // set of the allowed
features for dealer (all list see below in "Dealer features")
}
```

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 navixy_label)

name	description
navixy_label	show "Powered by Navixy" in UI (required for subpaas feature)

errors

- 12 – Dealer not found (if corresponding PaaS was not found in database)
- 201 – Not found in database (if there is no Ui settings data for corresponding PaaS)

Last update: September 10, 2020

Feedback

API path: `/feedback`.

```
<feedback> = {
  "text": <feedback text, string, may not be null>,
  "useragent": <optional, string>,
  "platform": <optional, string>,
  "screenshots": <optional, array of strings, base64-encoded
data:url image, example: [encoded image]>,
  "log": <optional, log file>
}
```

send_email

parameters

- feedback
- type – optional

Send email with feedback message on feedback.toEmail Where `type` is one of strings: `support_request` (default), `feature_request` and `review`.
Screenshot and log will be added to email as attachments.

response

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

Last update: August 21, 2020

File

API path: `/file`.

stats/read

Get user's files statistic.

response

```
{
    "success": true,
    "value": {
        "file_count": 24,        // count of all updloaded files
        "total_size": 40192953   // total files size in bytes
        "quota": 104857600       // space available to the user in
bytes
    }
}
```

Last update: October 23, 2020

Notification

API path: `/notification`.

list

List user notifications.

response

```
{
    "success": true,
    "list": [<notification>, ...]
}
```

where

```
<notification> =
    {
        "id": <int>,
        "message": <string>,
        "show_till": <date/time> // date until notification should
be showed, e.g. "2014-08-03 17:27:28"
    }
```

Last update: August 21, 2020

Timezone

API path: `/timezone`.

list

Information about all supported timezones for the specified locale. Does not require user authorization.

response

```
{
    "success": true,
    "list": [
        {
            "zone_id": <string>,     // timezone ID, which is used
throughout the API, e.g. "Africa/Dar_es_Salaam"
            "description": <string>, // Localized description of
the timezone, e.g. "Ekaterinburg"
            "base_offset": <double>,    // base timezone offset in
hours, e.g. 4 for Moscow. 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
timezone, e.g. "RU",
            "alt_ids": [<string>, <string>] //list of strings,
optional, alternative timezone IDs
        },
        ...
    ]
}
```

errors

- only standard errors

Last update: August 21, 2020

Custom Field File

API path: `/custom_field/file`.

create

parameters

- filename – **string**, optional
- size – **integer**
- metadata – **object**, optional
- type – **"image"** | **"file"**
- entity_type – **string**, see entity types

response

```
{
  "success": true,
  "value": {
    "file_id": 111,
    "url": "http://bla.org/bla",
    "expires": "2020-02-03 03:04:00",
    "file_field_name": "file1",
    "fields": {
      "token": "a43f43ed4340b86c808ac"
    }
  }
}
```

errors

- 268 – File cannot be created due to quota violation.
- 271 - File size is larger than the maximum allowed (by default 16 MB).

Last update: February 25, 2021

Entity actions

Entity describes a class of objects for which representation and editable fields can be customized. For example, you can add your own custom fields to **places** entity or rearrange existing fields.

entity is:

```
<entity> = {
    "id": 123, //identifier
    "type": "place", //currently, only "place" is supported
    "settings": {
        "layout": { //describes layout of fields for entity.
                "sections": [ //each section can contain one or
more fields. At least one section must exist in layout.
                  {
                    "label": "Section label",
                    "field_order": [ //built-in fields and ids of
custom fields (as strings)
                      "label",
                      "location",
                      "131212",
                      "tags",
                      "description"
                    ]
                  }
                ]
              }
    }
}
```

Entity Types: * **place** - a place object, the same that is available through place API

Builtin fields:

- label
- location
- tags
- description
- **task** - a task object, the same that is available through task API

Builtin fields:

```
* employee
* status
* label
* location
```

```
* period
* status_change_date
* arrival_date
* tags
* stay_duration
* description
* external_id
* form
```

list

Get list of entities which are available for customization.

parameters

none

response

```
{
    "success": true,
    "list": [ <entity>, ... ]
}
```

errors

Standard errors only.

read

Get entity by id or by type

parameters

name	description	type
id	ID of an entity	int
type	type of an entity	entity type string, see above

Exactly one of these parameters must be specified. They can't be both null or both non-null.

response

```
{
    "success": true,
```

```
    "entity": <entity>,
    "fields": [ //fields associated with this entity
        <field>,
        ...
    ]
}
```

errors

- 201 (Not found in database) – if there is no entity with such ID

update(entity)

Updates settings of customizable entity. Entity must have a valid id.

required subuser rights: places_custom_fields_update for entities with type `place`

WARNING: `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 is allowed. Fields cannot be duplicated, even in different sections.

parameters

name	description	type
entity	Entity object with valid id and settings	object

errors

- 201 (Not found in database) – if there is no entity with such ID
- 7 (Invalid parameters) - if entity object violates restrictions described above

response

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

Last update: February 25, 2021

Entity fields

API path: `/entity/fields`.

Fields actions

Field allows to add custom information to a customizable entity. Each field belongs to one entity.

field is:

```
<field> = {
    "id": 131312, //identifier, null when new object
    "label": "Additional info",
    "type":  "text", //type of field, see below
    "required": false, //whether field is required to be filled or
not
    "description": "Info about place", //Additional info about the
field, max 250 characters
    "params": { ... } //type-specific. If no specific params, this
field should be omitted
}
```

field types:

- `text` - text field up to 700 unicode symbols

Special params: none

- `bigtext` - bigger text field, up to 20000 unicode symbols with reduced search and sorting capabilities

Special params: none

- `email` - field for storing email, validated to contain valid email address

Special params: none

- `phone` - field for storing phone number, validated to contain valid phone number

Special params: none

- `decimal` - decimal number from -999999999999.999999 to 999999999999.999999 . Values are stored up to the sixth decimal place

Special params: none

- `integer` - integer number from `-2^63` to `2^63 - 1`

Special params: none

- `employee` - link to employee

Special params:

```
{
  "responsible": true //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 mobile app to view. Thus, field employee can view all places assigned to him to visit them, etc.

- `file` - link to file

Special params:

```
{
  "allowed_extensions": ["docx", "pdf"]
}
```

- `image` - link to image file

Special params: none

read(entity_id)

Get 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

response

```
```

```
{
    "success": true,
    "list": [ <field>, ... ]
}
```

errors

- 201 (Not found in database) – if there is no entity with such ID

update(entity_id, fields, delete_missing)

Update a set of custom fields associated with the specified entity.

required subuser 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 cannot change type of a 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.

WARNING 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	\<field>[]
delete_missing	(optional, default is false) delete fields not present in `fields` list	boolean

errors

- 201 (Not found in database) – if there is no entity with such ID
- 7 (Invalid parameters) - if fields violate restrictions described above

response

A list of **all** fields associated with the specified entity. Newly created fields will have their IDs filled.

```
{
    "success": true,
    "list": [ <field>, ... ]
}
```

Last update: February 25, 2021

Entity search Conditions

API path: `/entity/search_conditions`.

Search conditions are used to search and filter list of certain entities by built-in and/or custom fields.

Example:

```
<search_conditions> = [
    {"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 are represented by an array, each condition during search is 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 are joined using `AND` operator.

WARNING: A maximum of 72 conditions can be used at once, including nested conditions.

Condition types

AND

```
<and_condition> = {
    "type":"and",
    "conditions":[
            <list of other conditions here...>
```

```
    ]
}
```

Evaluates all specified conditions and joins them using `AND` boolean operator.

OR

```
<or_condition> = {
    "type":"or",
    "conditions":[
            <list of other conditions here...>
    ]
}
```

Evaluates all specified conditions and joins them using `OR` boolean operator.

NUMBER EQUALS

```
<eq_condition> = {
      "type": "eq",
      "field":"18", //built-in field or field id
      "value": 1111 //number value to which field is matched
against. Can be decimal.
                    //Must be between -2^63 and 2^63-1. No more
than 6 fraction digits
}
```

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.

CONTAINS STRING

```
<contains_condition> = {
    "type": "contains",
    "field":"label", //built-in field or field id
    "value": "who" //string value to which field is matched
against.
                   //Cannot  be null or empty, max length is 760
}
```

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.)

Last update: August 21, 2020

History

API path: `/history`.

Tracker history entry

```
{
    "id": 1,
    "type": "tracker",
    "is_read": false,
    "message": "Alarm",
    "time": "2020-01-01 00:00:00",
    "event": "offline",          // type of history event extension
    "tracker_id": 2,     // column object_id
    "rule_id": 3,        // column event_id
    "track_id": 4,
    "location":{
        "lat": 50.0,
        "lng": 60.0,
        "precision": 50
    },
    "address": "address",        // string. address of location or
"" (empty string)
    "extra": {
        "task_id": null , //related task identifier
        "parent_task_id": null, //related parent task identifier
(for task checkpoint related history entries)
        "counter_id": null, //related counter identifier
        "service_task_id": null, //related service task id
        "checkin_id": null, //related check-in marker
        "place_ids": null, //related place identifiers,
        "last_known_location": false, //true if location may be
outdated,
        "tracker_label": "Tracker label",//tracker label
        "emergency": false //true for events with a same flag,
optional
    }
}
```

Date/time type described in data types description section.

read

Returns history entry with the specified id.

parameters

- id – **int**. history entry ID

• add_tracker_label – **boolean**. optional, if true tracker label will be added to message

response

```
{
    "success": true,
    "value": ${history_entry}
}
```

where `history_entry` described in Tracker history entry.

errors

• 201 – Not found in database (when there are no history entries with that id)

mark_read

Mark history entry as read by **id** (see: Tracker history entry).

parameters

• id – **int**. Tracker history entry ID

response

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

errors

• 201 – Not found in database (when there are no history entries with that id)

mark_read_all

Mark all user's history entries read.

response

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

Last update: October 23, 2020

Tracker history

API path: `/history/tracker/`.

list

List less then or equal to **limit** of tracker events filtered by event types (**events**) between **from** date/time and **to** date/time sorted by **time** field.

parameters

- **trackers** – **[int]**. list of tracker's ids
- **from** – date/time. start date/time for searching
- **to** – date/time. end date/time for searching. must be after "from" date
- **events** – `["string"]` (optional, default: all). list of history types
- **limit** – **int** (optional, default: maxHistoryLimit. max count of entries in result
- **ascending** – **boolean** (optional, default: **true**). Sort ascending by time when it is **true** and descending when **false**.

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 by default. (Note for StandAlone: this value configured by maxHistoryLimit config option).

example

```
https://api.navixy.com/v2/fsm/history/tracker/list?
hash=user_hash&trackers=[tracker_id]&from=2018-02-19
10:29:00&to=2018-02-19 11:30:00&events=["event_type"]
```

response

```
{
    "success": true,
    "list": [ ${history_entry}, ... ], // list of zero or more
JSON objects
    "limit_exceeded": false // boolean. false when listed all
history entries satisfied to conditions
    // and true otherwise
}
```

where `history_entry` described in Tracker history entry.

errors

- 211 – Requested time span is too big (time span between **from** and **to** is more than maxReportTimeSpan days).
- 212 – Requested **limit** is too big (**limit** is more than maxHistoryLimit).
- 217 – List contains nonexistent entities – if one of the specified trackers does not exist or is blocked.

Last update: October 23, 2020

History type

API path: `/history/type`.

list

Returns available history types with localized descriptions.

parameters

- **locale** – locale code
- **only_tracker_events** – boolean (optional). Default - true.

response

```
{
    "success": true,
    "list": [<history_type>, ...]
}
```

where **history_type** is

```
{
    "type": "alarmcontrol",       // history type, e.g.
"alarmcontrol"
    "description": "Car alarm" // localized description, e.g. "Car
alarm"
}
```

Last update: August 28, 2020

History unread

API path: `/history/unread`.

list

List less than or equal to **limit** of the latest user's unread history entries.

parameters

- limit, int, optional
- from, date/time, optional

Default and max limit is maxHistoryLimit.

Type of **from** is date/time. Default **from** is **now** minus one year.

response

```
{
    "success": true,
    "list": [${history_entry}, ... ] //list of zero or more JSON
objects
}
```

where **history_entry** described in Tracker history entry

errors

- 212 – Requested limit is too big (more maxHistoryLimit config option)

count

Get count of user's unread history messages from **from** date.

parameters

- from – optional
- type – optional

Type of **from** is date/time. Default **from** is **now** minus one year.

response

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


Last update: October 23, 2020

Plugin

API path: `/plugin`.

Plugins are special software modules which modify the behavior of various API calls.

Plugin object structure

```
{
    "id": <plugin id, e.g. 1>, //int
    "type": <plugin type, e.g. "tracker_register">, //String
    "ui_module": <plugin ui module name, e.g.
"Registration.appPlugins.BundledSim">, //String
    "module": <plugin module name, e.g.
"com.navixy.plugin.tracker.register.bundled_sim">, //String
    "filter": { //a model filter which describes to which device
models this plugin is applicable
        "exclusion": true, //if true, "models" lists models NOT
supported by this plugin, if false, "models" contains all
supported models
        "values": <list of the regexes for models which are (not)
supported by this plugin, e.g. ["navixymobile",
"mobile_unknown.*"]> //string[]
    },
    "parameters" : { ... } //Plugin-specific parameters as JSON
object. This field is omitted if it's null (and it is null most of
the time)
}
```

Example

```
{
    "id": 4,
    "type": "tracker_report",
    "module": "com.navixy.plugin.tracker.report.trip",
    "ui_module": "Trip",
    "filter": {
        "exclusion": true,
        "values": []
    }
}
```

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.

response

```
{
    "success": true,
    "list": [ <plugin>, … ]
}
```

For "plugin" object structure, see plugin/.

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: October 23, 2020

Report plugins

Trips report

A report on detailed trip history.

parameters

Default **plugin_id**: 4.

Plugin-specific parameters:

name	description	type
hide_empty_tabs	If `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean
include_summary_sheet_only	If `true` the report will contain only a summary sheet for all chosen devices.	boolean
split	Trips will be split by stops if `true`.	boolean
show_idle_duration	Will show idle duration in report if `true`.	boolean
show_coordinates	Every address will contain longitude and latitude if `true`.	boolean
filter	If `true` 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
group_by_driver	Group trips by driver assigned to the device if `true`.	boolean

Stops report

A report on detailed stops history.

parameters

Default **plugin_id**: 6.

Plugin-specific parameters:

name	description	type
hide_empty_tabs	If `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean
show_coordinates	Every address will contain longitude and latitude if `true`.	boolean

Trips and stops by shifts report

A report on trips and stops by shifts.

parameters

Default **plugin_id**: 77.

Plugin-specific parameters:

name	description
hide_empty_tabs	If `true`, empty tabs will be hidden.
show_seconds	If `true` timestamps will be with seconds.
shifts	List of shifts with names, start and end time. e.g. `[{"name":"Shift1","start_time":"00:00","end_time":"23:59"}]`
filter	If `true` short trips will not coincide (shorter than 300m/have less tha 4 points total and if the device circles around one point (e.g., star pattern from GPS drifting)).
show_coordinates	Every address will contain longitude and latitude if `true`.

name	description
split_at_midnight	Split shifts at midnight if `true`.

- `shifts` is:

```
{
  "shifts": [{
    "name":"Shift1",
    "start_time":"00:00",
    "end_time":"23:59"
  }]
}
```

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 `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean
show_mileage	Adds mileage to the report if `true`.	boolean
show_not_visited_zones	Will show non visited zones if `true`.	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
zone_ids	List of zone ids.	array of int

POI visits report

A report on date, time, and the number of visits to POIs.

parameters

Default **plugin_id**: 85.

Plugin-specific parameters:

name	description	type
hide_empty_tabs	If `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean
show_mileage	Adds mileage to the report if `true`.	boolean
show_not_visited_places	Will show non visited POIs if `true`.	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.	array of int

Car security report

A report on alarms, tow alerts, AutoControl events, and crashes.

parameters

Default **plugin_id**: 15.

Plugin-specific parameters:

name	description	type
hide_empty_tabs	If `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean

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 `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean

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 `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean

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 `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean

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 `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean
group_by_type	If `true` events will group by type.	boolean

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 `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean
show_detailed		boolean

name	description	type
	If `true` will contain detailed engine hours tab.	
include_summary_sheet_only	If `true` the report will contain only a summary sheet for all chosen devices.	boolean
filter	If `true` 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

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 `true` timestamps will be with seconds.	boolean
graph_type	The type of X-axis. Can be "time" or "mileage".	string enum
detailed_by_dates	If `true` show final data on fuel traffic for each day in the period.	boolean
include_summary_sheet_only	If `true` the report will contain only a summary sheet for all chosen devices.	boolean
use_ignition_data_for_consumption	Calculate consumption only when the ignition was on if `true`.	boolean

name	description	type
include_mileage_plot	Optional. Used if `graph_type = time`. Show mileage plot if `true`.	boolean
filter	If `true` 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_speed_plot	If `true` show speed plot.	boolean
smoothing	Smooth graph if `true`. Smoothing reduces the accuracy of calculating refills or drains.	boolean
surge_filter	If `true` enables surge filter.	boolean
surge_filter_threshold	Defines a level of surge filter. Can be 0.01 - 0.99.	float
speed_filter	If `true` enables speed filter.	boolean
speed_filter_threshold	Defines a speed filter threshold.	int

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		boolean

name	description	type
	If `true`, a table with statistics for every single day in selected date range will be added to the report.	
filter	If `true` 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 `true` the report will contain only a summary sheet for all chosen devices.	boolean

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 `true`, empty tabs will be hidden.	boolean
details_interval_minutes	The interval in minutes. Can be `[30, 60, 180, 360]`.	int
graph_type	The type of X-axis. Can be "time" or "mileage".	string 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
  }]
}
```

Speed violation

A report on speeding instances.

parameters

default **plugin_id**: 27.

Plugin-specific parameters:

name	description	type
hide_empty_tabs	If `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean
min_duration_minutes	A minimum time in seconds when speed is more than `max_speed` 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 `true`.	boolean
filter	If `true` 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

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 `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean

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 `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean

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 `true`, empty tabs will be hidden.	boolean
details_interval_minutes	The interval in minutes. Can be `[5, 30, 60, 180, 360]`.	int
graph_type	The type of X-axis. Can be "time" or "mileage".	string enum

name	description	type
smoothing	Smooth data if `true`.	boolean
show_address	Address of each reading appears in report if `true`.	boolean
filter	If `true` 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
  }]
}
```

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 `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean
min_working_period_duration		int

name	description	type
	A minimum time in seconds the equipment works to determine activity. Min = 1.	
show_idle_percent	If `true` show percentage of idling.	boolean
filter	If `true` 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
  }]
}
```

Tasks report

A report on tasks statuses.

parameters

default **plugin_id**: 42.

name	description	type
hide_empty_tabs	If `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean
show_external_id	Show external ID of task if `true`.	boolean
show_description	Show description of task if `true`.	boolean

name	description	type
show_forms	Show forms when the task has it if `true`.	boolean
show_places_and_zones	Show places and geofences if `true`.	boolean

Form completion statistics report

A report on form fields completion rate.

parameters

default **plugin_id**: 70.

name	description	type
hide_empty_tabs	If `true`, empty tabs will be hidden.	boolean

Work statuses report

A report on status changes history.

parameters

default **plugin_id**: 47.

name	description	type
hide_empty_tabs	If `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean

Check-in report

A report on markers for Check-in function.

parameters

default **plugin_id**: 80

Plugin-specific parameters:

name	description	type
show_coordinates	If `true`, coordinates will be added to the report.	boolean

Driver shift change report

A report on driver identification.

parameters

default **plugin_id**: 66.

Plugin-specific parameters:

name	description	type
hide_empty_tabs	If `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean

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 `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean
filter	If `true` 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_summary_sheet_only	If `true` the report will contain only a summary sheet for all chosen devices.	boolean
group_type	A group type. Can be "province" or "country".	string enum

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 `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean
group_by_type	Groups events by type if `true`.	boolean
event_types	A list of event types that will be considered.	array of string

- the object with all `event_types` is:

```
{
  "event_types":
["auto_geofence_in","auto_geofence_out","door_alarm","forward_collisio
}
```

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 `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` 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

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 `true`, empty tabs will be hidden.	boolean
show_seconds	If `true` timestamps will be with seconds.	boolean

Eco-driving report

A report on safety driving.

parameters

default **plugin_id**: 46.

Plugin-specific parameters:

name	description	type
harsh_driving_penalties	A list of penalties for harsh driving.	array of objects

name	description	type
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 `speed_limit` to determine violation.	int
use_vehicle_speed_limit	If `true` vehicle speed limit used instead of `speed_limit` parameter.	boolean
show_seconds	If `true` 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.

Stay in zones report

Custom report for AO NIPIGAZ

parameters

default **plugin_id**: 84

plugin-specific parameters:

name	description	type
show_seconds	If true, time values in report should have format with seconds. Default is **false**.	boolean
show_tags	If true, tags fields will be added to the report. Default is **false**.	boolean
min_minutes_in_zone	Minimum time in zone (geofence). Default is **5**.	int, min value 1
zone_ids	IDs of user zones, required, min size 1, max size 30	list of ints

Last update: May 12, 2021

Report schedule

API path: `/report/schedule`.

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 array of string. List of emails.
- `email_format` - string 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.

create

Create 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/fsm/report/schedule/create'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "shedule":
{"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

Delete report schedule with the specified id.

required sub-user rights: `reports`

parameters

name	description	type
schedule_id	Id of the report schedule to delete.	int

examples

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/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`

examples

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/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 "id", "fire_time", "last_result".	JSON object

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/report/schedule/update'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "shedule":
{"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: October 23, 2020

Report tracker

API path: `/report/tracker`.

delete

Delete 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/fsm/report/tracker/delete'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "report_id":
"1234567"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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".	string 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/fsm/report/tracker/
download' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "report_id":
"1234567", "format": "pdf"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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 date/time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone).	string
to	A string containing date/time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone). 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 geocoder/.	string
trackers	List of trackers' ids to be included in report (if report is by trackers).	array of int
employees	List of employees' ids to be included in report (if report is by employees).	array of int
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]}`.	JSON object
plugin	A plugin object (see below).	JSON 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
show_seconds	Flag to define whether time values in report should have format with seconds. `true` - show seconds, `false` - don't show seconds.	boolean

Plugin example:

```
{
  "details_interval_minutes":60,
  "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/fsm/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/fsm/report/tracker/list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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` - array of int. 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` - array of int. 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` - string enum. Type of created report.
- `from` - string. "from" parameter from generate.
- `to` - string. "to" parameter from generate.

errors

- No specific errors.

retrieve

Retrieve 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/fsm/report/tracker/
retrieve' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "report_id":
"1234567"}'
```

HTTP GET

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

response

Example


```
{
"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/fsm/report/tracker/status'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "report_id":
"1234567"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: October 13, 2020

Subuser

API path: `/subuser`.

Contains API calls related to sub-users, that is, additional users who have access to your account and monitoring assets. Sub-users is convenient way for corporate clients to provide multiple employees, who have different roles and priveleges, with access to the monitoring system.

"Usual" user account is 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's 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.

```
<subuser> = {
      "id": 103, //sunuser id, can be null (when creating new sub-
user)
      "activated": true,                // true if user is
activated (allowed to login)
      "login": "user@test.com",         // User email as login.
Must be valid unique email 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_type": "legal_entity",     // either "legal_entity",
"individual" or "sole_trader"
      "phone": "491761234567",          // User's or contact phone
(10-15 digits)
      "post_country": "Germany",        // country part of user's
post address
      "post_index": "61169",            // index part of user's
post address
      "post_region": "Hessen",          // region part of user's
post address
```

```
      "post_city": "Wiesbaden",          // city from postal address
      "post_street_address": "Marienplatz 2", // street address
      "registered_country": "Germany",   // country part of user's
registered address
      "registered_index": "61169",       // index part of user's
registered address
      "registered_region": "Hessen",     // region part of user's
registered address
      "registered_city": "Wiesbaden",    // city from registered
address
      "registered_street_address": "Marienplatz 2", // User's
registered address
      "state_reg_num": ${string},        // State registration
number. E.g. EIN in USA, OGRN in Russia. 15 characters max.
      "tin": ${string},                  // Taxpayer identification
number aka "VATIN"
      "legal_name": "E. Biasi GmbH",     // user legal name (for
"legal_entity" only)
      "iec": ${string},                  // Industrial Enterprises
Classifier aka "KPP" (used in Russia. for "legal_entity" only)
      "security_group_id": 333,          // Id of the security
group to whic sub-user belongs to. Can be null, which means
default group with no privileges
      //this fields are read-only, they should not be used in user/
update
      "creation_date": "2016-05-20 00:00:00" // date/time when
user was created
    }
```

delete

Delete sub-user. This operation cannot be reversed.

required tariff features: multilevel_access – for ALL trackers **required subuser rights:** admin (available only to master users)

parameters

- **subuser_id** - **int**. id of the sub-user belonging to current account

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 database – if sub-user with such id does not exist or does not belong to current master user.

list

List all subusers belonging to current user.

required tariff features: multilevel_access – for ALL trackers **required subuser rights:** admin (available only to master users)

parameters

none

response

```
{
    "success": true,
    "list": [<subuser>, ... ] //list of all sub-users belonging to
this master account
}
```

Subuser object is 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 subuser rights:** admin (available only to master users)

parameters

- **user** - **JSON object**. object without "id" field
- **password** - **printable string**. 6 to 20 characters. New sub-user's password.

response

```
{
    "success": true,
```

```
    "id": <id of the created sub-user>
}
```

Subuser object is 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)
- 201 – Not found in database – when specified security_group_id does not exist
- 206 – login already in use (if this login email already registered)

update

Update subuser data.

required tariff features: multilevel_access – for ALL trackers **required subuser rights:** admin (available only to master users)

parameters

- **user** - **JSON object**. object with "id" field

response

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

Subuser object is 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)
- 201 – Not found in database – if sub-user with such id does not exist or does not belong to current master user. Also when specified security_group_id does not exist

Last update: October 23, 2020

Subuser security group

API path: `/subuser/security_group/`.

Contains API calls related to security groups, that is, groups of sub-users with the specified set of rights and privileges.

Security group object structure

```
${security_group} = {
      "id": 103, //group id, can be null (when creating new
security group)
      "label": "Managers", //group label
      "privileges": {
        "rights": [
          "tag_update", "tracker_register" //a set of rights
granted to security group (see below)
        ],
        "store_period": "1d" // optional, 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 f 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

create

Create new security group.

required tariff features: multilevel_access – for ALL trackers **required subuser rights:** admin (available only to master users)

parameters

- **group** - **JSON object**. ${security_group} without "id" field

response

```
{
    "success": true,
    "id": ${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

Delete 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 subuser rights:** admin (available only to master users)

parameters

- **security_group_id** - **int**. id of security group, which must be deleted.

response

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

errors

- 13 – Operation not permitted – if user has insufficient rights
- 201 – Not found in 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 subuser rights:** admin (available only to master users)

parameters

none.

response

```
{
    "success": true,
    "list": [${security_group}, ... ] //list of all sub-users
belonging to this master account
}
```

Security group object is 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)

update

Update existing security group.

required tariff features: multilevel_access – for ALL trackers **required subuser rights:** admin (available only to master users)

parameters

- **group** - **JSON object**. ${security_group} with "id" field

response

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

errors

- 13 – Operation not permitted – if user has insufficient rights
- 201 – Not found in 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: October 23, 2020

Subuser session

API path: `/subuser/session/`.

create

Create 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 subuser rights:** admin (available only to master users)

parameters

- **subuser_id** - **int**. id of the sub-user belonging to current account

response

```
{
    "success": true,
    "hash" : ${hash of the created subuser session}
}
```

Subuser object is 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)
- 201 – Not found in database – if sub-user with such id does not exist or does not belong to current master user.

Last update: October 23, 2020

Subuser tracker

API path: `/subuser/tracker`.

Contains API calls to control which tracker is available to which sub-user.

bind

Give access for sub-user to the specified trackers.

required tariff features: multilevel_access – for ALL trackers **required subuser rights:** admin (available only to master users)

parameters

- **subuser_id** - **int**. id of the sub-user belonging to current account.
- **trackers** - **array of int**. array of tracker id-s to associate with the specified sub-user. All trackers must belong to current master user.

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 database – if sub-user with such 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

Get a list of tracker ids to which this sub-user has access.

required tariff features: multilevel_access – for ALL trackers **required subuser rights:** admin (available only to master users)

parameters

- **subuser_id** - **int**. id of the sub-user belonging to current account.

response

```
{
    "success": true,
    "list" : [${tracker_id1}, ...] //list of tracker ids to which
this sub-user has acccess
}
```

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 database – if sub-user with such id does not exist or does not belong to current master user.

unbind

Disable access for sub-user to the specified trackers.

required tariff features: multilevel_access – for ALL trackers **required subuser rights:** admin (available only to master users)

parameters

- **subuser_id** - **int**. id of the sub-user belonging to current account.
- **trackers** - **array of int**. array of tracker id-s to associate with the specified sub-user. All trackers must belong to current master user.

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 database – if sub-user with such 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: October 23, 2020

Tag

API path: `/tag`.

tag object

```
<tag> =
   {
      "id": 3,
      "avatar_file_name": "asdf.jpg",
      "name": "hop",
      "color": "FF0000"
    }
```

tagged entity types

- place
- task
- task_schedule
- employee
- vehicle
- zone
- tracker

create

Create new tag.

required subuser rights: tag_update

parameters

- **tag** - JSON object.

response

```
{
    "success": true,
    "id": 111 //id of the created tag
}
```

errors

General types only.

delete

Delete tag with the specified id.

required subuser rights: tag_update

parameters

- **tag_id** - (int) id of the tag to delete.

response

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

errors

- 201 – Not found in database (if there is no tag with such id)

list

Get all tags belonging to user with optional filtering.

parameters

- **filter** - (string) optional filter for tag name, 3-60 characters or null.

response

```
{
    "success": true,
    "list": [ <tag>, ... ]
}
```

errors

General types only.

search

Search entities that bound with all of specified tags.

parameters

- **tag_ids** - (Array or int) tag IDs.
- **entity_types** - (Array of tagged entity types) optional, filter for entity types.

response

```
{
    "success": true,
    "result": {
      "place": [...], //array of place objects
      "task": [...], //array of task objects
      "task_schedule": [...], //array of task schedule objects
      "employee": [...], //array of employee objects
      "vehicle": [...], //array of vehicle objects
      "zone": [...], //array of zone objects
      "tracker": [...] //array of tracker objects
    }
}
```

errors

General types only.

update

Update existing tag.

required subuser rights: tag_update

parameters

- **tag** - JSON object.

response

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

errors

- 201 – Not found in database (if there is no tag with such id)

Last update: November 20, 2020

Tag avatar

API path: `/tag/avatar`.

assign

required subuser rights: tag_update

parameters

- **tag_id**
- **icon_id**

Assign icon_id (from standard icon set) to this tag. Icon_id can be null – this means that uploaded avatar should be used instead of icon.

response

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

errors

- 201 – Not found in database (when vehicle with **tag_id** not found in db)

upload

Upload 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/fsm/static/tag/avatars/abcdef123456789.png`.

required subuser 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** or **image/pjpeg**
- **image/png**

- **image/gif**

parameters

- **tag_id** – tag id
- **file** – image file
- **redirect_target** – (optional) URL to redirect. If **redirect_target** passed return redirect to `<redirect_target>?response=<urlencoded response json>`

response

```
{
    "success": true,
    "value": <string> // avatar file name
}
```

errors

Here is the list of errors that might occurred:

- 201 – Not found in database (when tag with **tag_id** not found in db)
- 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: September 9, 2020

User

API path: `/user`.

User specific actions:

- /user/activate
- /user/auth
- /user/get_info
- /user/get_tariff_restrictions
- /user/resend_activation

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 not found or session ended).

response

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

auth

Try to authenticate user.

It does not need authentication/hash and is available at `UNAUTHORIZED` access level.

parameters

name	description	type	restrictions
login	User email as login (or demo login)	string	not null
password	User password	string	

name	description	type	restrictions
			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 -X POST 'https://api.navixy.com/v2/fsm/user/auth' \
-H 'Content-Type: application/json' \
-d '{ "login": "test@email.com", "password": "password123456" }'
```

response

```
{
    "success": true,
    "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

Only session `hash`.

example

```
```

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

Get basic user info.

response

```
{
    "success": true,
    "paas_id": 7,
    "paas_settings": ${pass_settings},
    "user_info": {
        "id": 43568,                        // user id
        "login": "demo@navixy.com",         // user's login (in most
cases it's an email address)
        "title": "John Smith",              // user first and last
name or organization title
        "phone": "79123456789",             // user phone (if not
empty)
        "creation_date": "2016-05-20 01:10:34", // user
registration date/time
        "balance": 74.31,                   // user balance, max. 2
digits after dot. For subusers, this field should be ignored
        "bonus": 0,                         // user bonus, max. 2
digits after dot. For subusers, this field should be ignored
        "locale": "en_US",                  // user locale, for
example "en_EN"
        "demo": true,                       // true if this is a demo
user, false otherwise
        "verified" : true,                  // true if user email
already verified
        "legal_type" : "individual",        // string. "individual",
"legal_entity" or "sole_trader"
        "default_geocoder": "google",       // user's default
geocoder ("google", "yandex", "progorod", "osm", or "locationiq")
        "route_provider": "google",         // user's route provider
("progorod", "google" or "osrm")
        "time_zone": "America/New_York", // user timezone name
        "measurement_system" : "metric", // user's measurement
system ("metric", "imperial" or "us")
        "tin": "2345678239",                // Taxpayer
identification number aka "VATIN" or "INN"
        "iec": ${string},                   // Industrial Enterprises
Classifier aka "KPP". Used in Russia for legal entities. Optional.
        // postal address
        "post_country": "USA",              // country
        "post_region": "NY",                // region part of post
address (oblast, state, etc.)
        "post_index": "10120",              // post index or ZIP code
        "post_city": "New York",            // city part of post
address
        "post_street_address": "1556 Broadway, suite 416" // tail
part of post address
        // legal (juridical) address
```

```
        "registered_country": "USA",           // country
        "registered_region": "NY",             // region part of
post address (oblast, state, etc.)
        "registered_index": "10120",           // post index or
ZIP code
        "registered_city": "New York",         // city part of
post address
        "registered_street_address": "1556 Broadway, suite 416" //
tail part of post address
        "first_name": "John",
        "middle_name": "Walker",
        "last_name": "Smith",
        "legal_name": "QWER Inc." // juridical name (optional)
    },
    "master": { // returned only if current user is sub-user. All
fields have same meaning as in "user_info", but for master user's
account
    "id": 1234, // same as in "user_info"
    "demo": false, // same as in "user_info"
    "legal_type": "individual", // same as in "user_info"
    "first_name": "David", // same as in "user_info"
    "middle_name": "Middle", // same as in "user_info"
    "last_name": "Blane", // same as in "user_info"
    "legal_name": "Blah LLC", // same as in "user_info"
    "title": "David Blane", // same as in "user_info"
        "balance": 0.0, // master user balance, max. 2 digits
after dot. Only returned if subuser has "payment_create" right
        "bonus": 89.78, // master user bonus, max. 2 digits after
dot. Only returned if subuser has "payment_create" right
    },
    "tariff_restrictions": ${tariff_restrictions},
    "premium_gis": true,
    "features": ["branding_web"],
    "privileges": { // only returned for subusers. Describes
effective subuser privileges
        "rights": [
            "tag_update"
        ]
    }
}
```

where

- `paas_settings` same as `settings` in /dealer/get_ui_config response,
- `tariff_restrictions` is JSON object same as in /user/get_tariff_restrictions response,
- `features` is a set of allowed Dealer features.

get_tariff_restrictions

Gets user tariff restrictions.

parameters

Only session `hash`.

example

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

response

```
{
    "success": true,
    "value": ${tariff_restrictions}
}
```

where `tariff_restrictions` is JSON object:

```
{
    "allowed_maps": [${map_name}, ...] // [string]. list of
allowed maps, e.g. ["roadmap","osm"]
}
```

logout

Destroys current user session.

parameters

Only session `hash`.

example

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

response

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

resend_activation

Send 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

example

```
$ curl -X POST 'https://api.navixy.com/v2/fsm/user/logout' \
-H 'Content-Type: application/json' \
-d '{ "login": "users_login" }'
```

response

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

errors

- 201 (Not found in database) – user with 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", // timeout between sending activation
links in ISO-8601 duration format
    "remainder": "PT4M31.575S" // remaining time to next try in
ISO-8601 duration format
}
```

- 265 (Already done) – user already activated and verified

Last update: August 21, 2020

User password

API path: `/user/password`.

change

Changes password of user with the provided session hash (it is contained in password restore link from email sent to user by "user/restore_password".

NOTE: this call will receive only session hash from password restore email. Any other hash will result in result error code 4 (user not found or session ended)

parameters

- **password** (string) – New password for the user, 6 to 20 printable characters

response

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

errors

- 101 – In demo mode this function is disabled (if specified session hash belongs to demo user)

set

Changes password for logined user.

parameters

- **old_password** (string) – Current password of the user
- **new_password** (string) – New password for the user, 6 to 20 printable characters

response

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

errors

- 101 – In demo mode this function is disabled (if specified session hash belongs to demo user)
- 225 – New password must be different (if **old_password** = **new_password**)

- 248 – Wrong password (if **old_password** is wrong)

Last update: October 23, 2020

User personal info

API path: `/user/personal_info`.

update

Update user personal info.

Require 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** – optional. string. Country part of user's post address.
- **post_index** – optional. string. Index part of user's post address.
- **post_region** – optional. string. Region part of user's post address.
- **post_city** – optional. string. City from post address.
- **post_street_address** – optional. string. 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 USA, OGRN in Russia.

- **tin** – string. Taxpayer identification number.

response

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

errors

- 222 (Plugin not found) – when plugin 45 not available for user

Last update: October 23, 2020

User audit

API path: `/user/audit`.

checkin

This method is called when user has opened UI.

response

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

Last update: August 21, 2020

User audit log

API path: `/user/audit/log`.

audit_object type is JSON object:

```
{
    "id": 1, // ID of the audit record
    "user_id": 3, // Master user's ID
    "subuser_id": 3, // ID of the subuser who made an action
    "entry_category": "user", // Category of the entry on which an
action was made
    "entry_id": null, // Nullable. ID of the entry on which an
action was made
    "action": "login", // Action on entry
    "payload": null, // Nullable json-object. Additional
information about action
    "host": "192.168.88.1", // Host from which an action was made.
IPv4 or IPv6
    "user_agent": "Apache-HttpClient/4.1.1 (java 1.5)", // User
agent
    "action_date": "2018-09-03 11:32:34" // Date and time of the
action
}
```

list

Gets list of audit records available for current user.

required subuser rights: admin (available only to master users)

parameters

- **from** – **string**. Include audit objects recorded after this date, e.g. `2014-07-01 00:00:00`.
- **to** – **string**. Include audits before this date, e.g. `2014-07-01 00:00:00`.
- **subuser_ids** – **int[]**. (optional) Include audits for specific subusers, e.g. `[2, 3]`.
- **actions** – **string[]**. (optional) Include audits for specific actions only, e.g. `["user_checkin"]`. Set of valid values is formed by combinations of entry categories and actions.
- **limit** – **int**. Pagination. Maximum number of audit records to return, e.g. `10`.
- **offset** – **int**. Pagination. Get audits starting from, e.g. `0`.
- **sort** – **string[]**. (optional) Set of sort options. Each option is a pair of property name and sorting direction, e.g. `["action_date=acs", "user=desc"]`. 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 (subuser) 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.

response

```
{
    "success": true,
    "list": [ <audit_object>, ... ]
}
```

Last update: August 21, 2020

User session

API path: `/user/session`.

renew

Prolongs current user session.

response

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

Last update: August 21, 2020

Delivery

API path: `/user/session/delivery`.

Calls to work with "delivery" type sessions. Those are special sessions to integrate order (task) tracking functionality into external systems.

create

Create new user delivery session. In demo session allowed to create a new session only if it not already exists.

required subuser rights: admin (available only to master users)

response

```
{
    "success": true,
    "value": "42fc7d3068cb98d233c3af749dee4a8d" // created session
hash key
}
```

errors

- 101 (In demo mode this function is disabled) – current session is demo but weblocator session already exists.
- 236 – Feature unavailable due to tariff restrictions

read

Return current user delivery session key.

response

```
{
    "success": true,
    "value": <string> // session hash key
}
```

errors

- 201 – Not found in database (if there is no delivery session)

Last update: October 23, 2020

Push token

API path: `/user/session/push_token`.

bind

Binds Push token with current session.

parameters

- **application** (string) – Application ID, for now it's "navixy_iphone_viewer" or "navixy_android_viewer"
- **token** (string) – Push token
- **category_filter** (string) – Push notifications category filter, default is *

response

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

Using `category_filter` you can filter out unwanted notifications categories.

If `category_filter` equals to `*` this means all categories are 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 are 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

delete

Deletes push token that bound with the session.

response

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

errors

General types only.

Last update: October 23, 2020

User sessions weblocator

API path: `/user/sessions/weblocator`.

create

Create new user weblocator session. In demo session allowed to create a new session only if it not already exists.

required subuser rights: admin (available only to master users)

response

```
{
    "success": true,
    "value": "42fc7d3068cb98d233c3af749dee4a8d" // created session
hash key
}
```

errors

- 101 (In demo mode this function is disabled) – current session is demo but weblocator session already exists.
- 236 – Feature unavailable due to tariff restrictions

read

Return current user weblocator session key.

response

```
{
    "success": true,
    "value": <string> // session hash key
}
```

errors

- 201 – Not found in database (if there is no weblocator session).

Last update: August 21, 2020

User settings

API path: `/user/settings`.

CRUD actions for user settings.

`settings` type is JSON object:

```
{
    "time_zone": "Europe/Amsterdam",  // ISO timezone id
    "locale": "nl_NL",                // locale code
    "measurement_system": "metric"    // measurement system
("metric", "imperial", "us" or "metric_gal_us")
    "geocoder": "osm",                // preferred geocoder type
("google", "yandex", "progorod", "osm" or "locationiq")
    "route_provider": "google",       // preferred route finding
provider ("google", "progorod" or "osrm")
    "translit": false                 // true if sms notification
should be transliterated, false otherwise
}
```

`balance_alert_settings` type is JSON object:

```
{
    "emails": ["email1@example.com", "email2@example.com"]  //
array of emails to send alert message about balance
                                                            //
empty array means disclaimer of notifications
}
```

`file_storage_settings` type is JSON object:

```
{
    "auto_overwrite": <true|false> // default - false,
}
```

read

Read current user's settings.

response

```
{
    "success": true,
    "settings": ${settings},                          // JSON
object
    "file_storage_settings": ${file_storage_settings},   // JSON
object
```

```
    "balance_alert_settings": ${balance_alert_settings}, // JSON
object
    "first_user_balance_warning_period": "7d",           // first
interval to send alert
    "second_user_balance_warning_period": "2d"           // second
interval to send alert
}
```

Where `settings`, `balance_alert_settings` and `file_storage_settings` described above.

required subuser rights for **balance_alert_settings** and **file_storage_settings** fields: admin (available only to master users)

update

Update current user's settings.

parameters

- **time_zone** – ISO timezone id
- **locale** – locale code
- **measurement_system** – measurement system ("metric", "imperial", "us" or "metric_gal_us"). If field is not passed then default (**metric**) system will be used.
- **geocoder** – preferred geocoder type ("google", "yandex", "progorod", "osm" or "locationiq")
- **route_provider** – preferred route finding provider ("google", "progorod" or "osrm")
- **translit** – true if sms notification should be transliterated, false otherwise
- **balance_alert_settings** – JSON object containing array of emails
- **file_storage_settings** – JSON object

required subuser rights for **balance_alert_settings** and **file_storage_settings**: admin (available only to master users)

See examples above.

response

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

file_storage/update

Update current user's file storage settings

required subuser rights: admin (available only to master users)

parameters

- `file_storage_settings` – JSON object.

errors

- 13 – Operation not permitted – if user has insufficient rights

Last update: October 23, 2020

User UI settings

API path: `/user/settings/ui`

The user interface settings are 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 is provided but not guaranteed - when the quota is exceeded, data could be deleted.

read

Read setting value by key.

parameters

key - string. Length should be between 1 and 50 is 50 symbols, should only contain English letters, digits, '_' and '-'.

responses:

```
{
  "success": true,
  "value": "previously saved value"
}
```

When nonexistent key is provided:

```
{
  "success": false,
  "status": {
    "code": 201,
    "description": "Not found in database"
  }
}
```

errors

Standard errors

update

Set setting value.

parameters

key - string. Length should be between 1 and 50 symbols. Should only contain English letters, digits, '_' and '-'. **value** - string. Length should be between 0 and 8192 symbols.

responses:

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

errors

- Standard errors
- 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: August 21, 2020

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, 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": 56.5,
        "lng": 60.5,
        "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/
01gw2j5q7nm4r92dytolzd6koxy9e38v.png/lala.jpg"
    }],
    "form_id": 23423,
    "form_label": "Service request form"
}
```


- `id` - int. An id of a check-in.

- `marker_time` - string 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` - string enum. Can be "image" | "file".
  - `created` - string date/time. Date when file created.
  - `uploaded` - string 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` - string 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/fsm/checkin/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "checkin_id":
1}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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": 56.5,
         "lng": 60.5,
         "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/
01gw2j5q7nm4r92dytolzd6koxy9e38v.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.	array of
from	Optional. Start date/time for searching.	date/tim
to	Optional. End date/time for searching. Must be after "from" date.	date/tim
conditions	Optional. Search conditions to apply to list. See Search conditions. Allowed fields are `employee`, `location`, `marker_time`, `comment`.	array of
sort	Optional, offset, default is 0. List of sort expressions. See below.	array of
location	Optional, location with radius, inside which check-ins must reside	Locatio example { "lat" 56.823 "lng":

name	description	type
		`60.5941` `"radius`
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
show_nearby_geo_entities	Optional. If true, the call will search for places and zones where the location of the check-in falls and add their description to the response.	boolean

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
tracker_id	number	
marker_time	DateTime	
location	string	address
comment	string	
form	string	label

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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).

When parameter `show_nearby_geo_entities` is set, `<checkin>` will contain additional fields `places` and `zones`.

```
 "places": [{
      "id": integer,
      "label": string
  }, …],
  "zones": [{
      "id": integer,
```

```
        "label": string
    }, …]
```

errors

- 7 – Invalid parameters.
- 211 – Requested time span is too big (more than **maxReportTimeSpan** config option).
- 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 id-s.

required sub-user rights: `checkin_update`.

parameters

name	description	type
checkin_ids	List of check-in ids.	array of int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/checkin/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"checkin_ids": [2132,4533]}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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.

Last update: March 2, 2021

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/fsm/department/list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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.
- 211 – Requested time span is too big (more than **maxReportTimeSpan** config option).
- 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 department object without `id` field.	JSON object

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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.
- 211 – Requested time span is too big (more than **maxReportTimeSpan** config option).
- 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 department object.	JSON object

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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/fsm/department/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"department_id": 111}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: November 25, 2020

Working with employees

Employees are 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_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],
    "fuel_consumption": 8.2,
    "fuel_cost": 27.1
}
```


- `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. `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_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.
- `tags` - array of int. List of tag ids.
- `fuel_consumption` - decimal. Fuel consumption rate of employee's vehicle, measured in liters per 100 km.
- `fuel_cost` - decimal. The cost of a liter of fuel used by employee's vehicle.

API actions

API base path: `/employee`.

list

Gets all employees belonging to user.

response

```
{
    "success": true,
    "list": [<employee>]
}
```

- `list` - a list of employee objects.

examples

cURL

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

HTTP GET

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

errors

General types only.

create

Create new employee. If tracker id is specified, tracker's label will be changed to employee full name.

required sub-user rights: `employee_update`.

parameters

name	description	type
employee	An employee object without `id` field. Non-null.	JSON object
force_reassign	if true, specified tracker will be assigned to employee even if it already assigned to another	Boolean

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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 of the created employee
}
```

- `id` - int. An id of the created employee.

errors

- 247 – Entity already exists, if `tracker_id` !=null and exists an employee that already bound to this `tracker_id` .

read

Gets employee by its id.

parameters

name	description	type
employee_id	Id of an employee.	int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/employee/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"employee_id": 111}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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_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],
      "fuel_consumption": 14.2,
      "fuel_cost": 9.99
  }
}
```


- `value` - an employee object.

errors

- 201 – Not found in the database (if there is no employee with such an id).

update

Update existing employee. If it had tracker assigned and tracker id had changed, tracker label will be prepended with "Deleted ". New tracker's label will be changed to employee full name.

required sub-user rights: `employee_update`.

parameters

- **employee** – an employee object Non-null.
- **force_reassign** – if true, specified tracker will be assigned to employee even if it already assigned to another | name | description | type | | :--- | :--- | :--- | | employee | An employee object with `id` field. Non-null. | JSON object | | force_reassign | if true, specified tracker will be assigned to employee even if it already assigned to another | Boolean |

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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 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 with the specified id. If it had tracker assigned, tracker label will be prepended with "Deleted "

required sub-user rights: `employee_update`.

parameters

name	description	type
employee_id	Id of an employee to delete.	int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/employee/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"employee_id": 111}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/employee/delete?
hash=a6aa75587e5c59c32d347da438505fc3&employee_id=111
```

response

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

errors

- 201 – Not found in the database (if there is no employee with such an id).

batch_convert

Convert batch of tab-delimited employees and return list of checked employees with errors.

Required sub-user rights: `employee_update`.

parameters

name	description	type
batch	Batch of tab-delimited employees.	string
file_id	Preloaded file ID.	string
fields	Optional. Array of field names. Default is `["first_name", "middle_name", "last_name", "email", "phone"]`.	array of string
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.

Note that employees created this way must have either phone or email specified.

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_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],
       "fuel_consumption": 10.0,
       "fuel_cost": 0.94,
       "errors": <array_of_objects>
    }],
    "limit_exceeded": false
}
```

- `list` - list of checked employees.
    - `errors` - optional array of errors.
- `limit_exceeded` - boolean. `true` if given batch constrained by a limit.

errors

- 234 - (Invalid data format).

Last update: February 11, 2021

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 base path: `/employee/avatar`.

assign

Assign `icon_id` (from standard icon set) to this employee. 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 to whom the icon will assign.	int
icon_id	Id of the icon.	int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/employee/avatar/assign'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"employee_id": 2132, "icon_id": 3654}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/employee/avatar/assign?
hash=a6aa75587e5c59c32d347da438505fc3&employee_id=2132&icon_id=3654
```

response

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

errors

- 201 – Not found in the database (when employee with `employee_id` not found).

upload

Uploads avatar image for specified employee. Then it will be available from /employee/avatars/ e.g. `https://api.navixy.com/v2/fsm/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** or **image/pjpeg**.
- **image/png**.
- **image/gif**.

parameters

name	description	type
employee_id	Id of the employee to whom the icon will assign.	int
file	Image file.	string
redirect_target	Optional. URL to redirect. If passed returns redirect to `?response=`.	string

response

```
{
    "success": true,
    "value": "picture.png"
}
```

- `value` - string. Uploaded file name.

errors

- 201 – Not found in the database (when employee 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: November 16, 2020

About forms

Forms used to provide additional information, such as user 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.

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` - string 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` - string 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` - string enum. Can be "image" or "file".
- `created` - string date/time. Date when file created.
- `uploaded` - string 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` - string 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
id	Id of the form.	int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/form/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id": 2}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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".	string enum

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/form/download' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id": 2,
"format": "pdf"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: November 25, 2020

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` - string 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` - array of string enum. 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` - array of int. 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` - array of int. 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: November 16, 2020

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 an 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` - string 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` - string 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 base path: `/form/template`.

list

Gets all form templates belonging to current master user.

examples

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/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 `id`, `created`, `updated` fields.	JSON object

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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/fsm/form/template/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"template_id": 111}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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 `created`, `updated` fields.	JSON object

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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/fsm/form/template/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"template_id": 111}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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/fsm/form/template/stats/
read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"template_id": 111}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: November 25, 2020

Working with places

"Places" are business-specific points of interest 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 place, in various reports name of the place will be specified after the address.

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.

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 place.
- `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.
- `fields` - optional object. A map, each key of which is a custom field id *as a string*. See entity/fields
- `label` - string. The name of the place.

- `description` - optional string. Description of the place.
- `tags` - optional array of int. A list of tag_ids. Non-empty.
- `external_id` - optional string. Max length 32.

API actions

API base path: `/place`.

read

Gets place by ID.

parameters

name	description	type
place_id	ID of the place.	int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/place/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "place_id":
122304}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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 place with such ID.

list

Get places belonging to user.

parameters

name	description	type
place_ids	Optional. List of place IDs.	array of int
filter	Optional. Filter for all built-in and custom fields. If used with conditions, both filter and conditions must match for every returned place.	string
conditions	Optional. Search conditions to apply to list. Array of search conditions, see Search conditions.	array of objects
order_by	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 entity/.	string
ascending	Optional. If `false` – descending order.	boolean
limit	Optional. Limit.	int
offset	Optional. offset, default is 0.	int

examples

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/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 places count.

errors

General types only.

create

Creates new place.

required sub-user rights: `place_update`.

parameters

name	description	type
place	A place object without `id` field.	JSON object
ignore_missing_fields	Optional (default is false). If `true`, place can be created even without all required custom fields.	boolean

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/place/read' \
    -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 place.

errors

- 268 (Over quota) – if the user's quota for places exceeded.

update

Updates existing place.

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/fsm/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 place with such ID.

delete

Deletes place with the specified ID.

required sub-user rights: `place_update`.

parameters

name	description	type
place_id	ID of the place to delete.	int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/place/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "place_id":
122304}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/place/delete?
hash=a6aa75587e5c59c32d347da438505fc3&place_id=122304
```

response

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

errors

- 201 (Not found in the database) – if there is no place with such ID.

batch_convert

Converts batch of tab-delimited places and return list of checked places with errors.

Required sub-user rights: `place_update`.

parameters

name	description	type
batch	Batch of tab-delimited places.	string
file_id	Preloaded file ID.	string
fields	Optional. Array of field names, default is `["label", "address", "lat", "lng", "radius", "description", "tags"]`.	array of strings
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 array of strings. Tag names of the place.
- `limit_exceeded` - boolean. `true` if given batch constrained by a limit.

errors

- 234 (Invalid data format).

upload

Upload places.

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 places data.	File upload
error_policy	`ignore` or `fail`	string
duplicate_policy	`skip` or `update` or `fail`, **belongs only to external_id duplicates**	string
default_radius	Optional, radius for point, meters, default is 100.	int
geocoder	Geocoder type.	string
redirect_target	Optional URL to redirect. If `redirect_target` passed return redirect to `<redirect_target>?response=<urlencoded_response_json>`.	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 place contains external_id and place with this ID already exists and duplicate_policy=fail.
- 268 (Over quota) – if the user's quota for places exceeded.

Last update: February 4, 2021

Changing place 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

API path: `/place/avatar`.

upload

Uploads avatar image for specified place.

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/fsm/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* or *image/pjpeg*
- *image/png*
- *image/gif*

PARAMETERS

name	description	type
place_id	ID of the place.	int
file	Image file.	File upload
redirect_target	Optional URL to redirect. If **redirect_target** passed return redirect to `<redirect_target>?response=<urlencoded_response_json>`.	string

RESPONSE

```
{
    "success": true,
    "value": "Avatar file name"
}
```

- `value` - string. Avatar file name.

errors

- 201 (Not found in the database) – when place 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 place. `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
place_id	ID of the place.	int
icon_id	Optional. ID of the icon from standard icon set.	int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/place/avatar/assign' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "place_id":
122304, "icon_id": 1}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/place/avatar/assign?
hash=a6aa75587e5c59c32d347da438505fc3&place_id=122304&icon_id=1
```

response

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

errors

- 201 (Not found in the database) – when place with `place_id` not found.

Last update: November 16, 2020

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.

Task object

```
{
    "id": 111,
    "user_id": 3,
    "tracker_id": 22,
    "location": {
        "lat": 56.5,
        "lng": 60.5,
        "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>,
    "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` - string date/time. When task created. *IGNORED* in create/update.
- `from` - string date/time. Date AFTER which task zone must be visited.
- `to` - string 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` - string 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` - string 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` - string 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` - array of int. List of tag ids.
- `form` - form object. If present.
- `fields` - optional object. A map, each key of which is a custom field id *as a string*. See entity/fields

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/fsm/task/assign' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id":
23144, "tracker_id": 132421}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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 `["label", "from", "to", "address", "lat", "lng", "description"]`.	array of string
geocoder	Geocoder type.	string 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"	string enum
employee_ids	Optional. List of employee Ids to automatic assign	array of int
vehicle_ids	Optional. List of vehicle Ids to automatic assign	array of int

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": 56.5,
```

```
                "lng": 60.5,
                "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/fsm/task/count' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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	`task` object without fields which are **IGNORED**	JSON object
create_form	If `true` then check additional `form_template_id` field in `task` object and create form if it is not null. Default value is `false` for backward compatibility.	boolean

Minimal JSON object to create a new task must contain:

```
{
    "tracker_id": 22,
    "location": {
    "lat": 56.83717295,
    "lng": 60.59761920,
    "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` - string date/time. Start date of the interval - when the specified location has to be visited (in the user's time zone).
- `to` - string 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/fsm/task/create' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b" "task":
{"tracker_id": 22, "location": {"lat": 56.83717295, "lng":
60.59761920, "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/fsm/task/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id":
23144}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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. `["unassigned","failed"]`.	array of string
trackers	Optional. Ids of the trackers to which task assigned.	array of int
from	Optional. Show tasks which are actual AFTER this date, e.g. "2020-07-01 00:00:00".	string date/time
to	Optional. Show tasks which are actual BEFORE this date, e.g. "2020-07-01 00:00:00".	string date/time
conditions	Optional. Search conditions to apply to list. Array of search conditions.	array of SearchCondition
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.	array of string
tag_ids	Optional. Tag IDs assigned to the task.	array of int
location	Optional. Location with radius, inside which task zone centers must reside. Example: `{ "lat": 56.823777, "lng": 60.594164, "radius": 350 }`	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	number	
employee	number	id
status	string	
label	string	
location	string	address
from	string date/time?	
to	string date/time?	
status_change_date	string date/time?	
arrival_date	string date/time?	
stay_duration	Seconds	
description	string	
external_id	string?	
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=acs", "address=desc", "employee=desc"].

SORT FIELDS

Name	Type	Comment
id	number	
employee	string	full name or tracker label
status	string	
label	string	
location	string	address
from	string date/time?	
to	string date/time?	
status_change_date	string date/time?	
arrival_date	string date/time?	
stay_duration	Seconds	
description	string	
external_id	string?	
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/fsm/task/list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

HTTP GET

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

response


```
{
    "success": true,
    "list": [{
        "id": 111,
        "user_id": 3,
        "tracker_id": 22,
        "location": {
            "lat": 56.5,
            "lng": 60.5,
            "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` - string date/time. Date of creation of a task, unchangeable field.
  - `from` - string date/time. Start date of the interval - when the specified location has to be visited (in the user's time zone).
  - `to` - string 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` - string 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` - string 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` - string 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` - string 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/fsm/task/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id":
23144}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/task/read?
hash=a6aa75587e5c59c32d347da438505fc3&task_id=23144
```

response

```
{
    "success": true,
    "value":  <task, 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		int

name	description	type
	Zero-based index at which checkpoint should be inserted into route.	

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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/fsm/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	`task` object without fields which are *IGNORED*.	JSON object
create_form	If `true` then check additional `form_template_id` field in `task` object and create, replace or delete task's	boolean

name	description	type
	form. Default value is `false` for backward compatibility.	

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/task/update' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b" "task": {"id":
22379, "location": {"lat": 56.83717295, "lng": 60.59761920,
"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: December 17, 2020

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` - string date/time. When route created. *IGNORED* in route/create, route/update.
- `from` - string date/time. Date AFTER which first checkpoint zone must be visited, depends on first checkpoint `from`, *IGNORED* in route/create, route/update.
- `to` - string 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` - string date/time. When route status changed. *IGNORED* in route/create, route/update.
- `origin` - string. A route origin. *IGNORED* in route/create, route/update.
- `tags` - array of int. List of tag ids.
- `checkpoint_ids` - array of int. List of route checkpoint ids in order of execution. *IGNORED* in route/create.

API actions

API base path: `/task/route`

assign

(Re)assign route to new tracker (or make it unassigned).

required sub-user rights: `task_update`.

parameters

name	description	type
route_id	ID of the route 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 none.	int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/task/route/assign' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "route_id":
11231, "tracker_id": 223465}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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 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 *IGNORED*.	JSON object
checkpoints	Checkpoints array of checkpoints object without fields which are *IGNORED*.	array of JSON objects
create_form	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.	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": 56.83717295,
        "lng": 60.59761920,
        "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` - string date/time. Start date of the interval - when the specified location has to be visited (in the user's time zone).
- `to` - string 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/fsm/task/route/create' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
223652, "label": "Name", "description": "Description example",
"checkpoints": [{"tracker_id": 223652, "location": { "lat":
56.83717295, "lng": 60.59761920, "radius": 150}, "label": "Name",
"description": "Description example", "from": "2014-02-03
04:05:06", "to": "2014-03-04 05:06:07"}], "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` - array of int. 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
route_id	ID of the route to delete.	int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/task/route/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "route_id":
23144}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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. `["unassigned","failed"]`. Default all.	array of string enum
trackers	Optional. List of `tracker_id` to which task assigned.	array of int
from	Optional. Show tasks which are actual AFTER this date, e.g. "2020-06-01 00:00:00".	string date/ time
to	Optional. Show tasks which are actual BEFORE this date, e.g. "2020-07-01 00:00:00".	string date/ time

name	description	type
filter	Optional. Filter for task label and description. If **trackers**, **filter**, **from** or **to** is not passed or *null* then appropriate condition not used to filter results.	string

examples

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/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 *IGNORED*.	

name	description	type
		JSON object
checkpoints	List of checkpoint_entry objects. Should be null if **route**'s field **checkpoint_ids** 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 `true` then check additional `form_template_id` field in every **checkpoint** object and create, replace or delete checkpoint's form. Default value is `false` for backward compatibility.	boolean

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: November 25, 2020

Optimizing routes

To minimize transit time and costs, it may be beneficial to reorder route checkpoints so total travel time between them is minimal. Our platform provides a way to perform such optimisation. You don't even need to create route and checkpoints, you just provide data required to optimize and algorithm returns order in which points should be visited.

API actions

API path: `/task/route/points/optimize`.

optimize

Suggest optimal order for given route points. Suggested order will correspond to route points time windows: points with earlier time windows will have lower ordinal numbers. If time windows overlaps each other, such points can have any order due to maximize summary efficiency of the route.

required sub-user rights: `task_update`.

parameters

- **start_point** - (object) coordinates of location, from where performer will come. Count of points must be in the range [2..15], example:

```
{ "lat": 15.233, "lng": -5.554 }
```

* **route_points** - (array of objects) points, which performer must visit, example:

```
[
  {"location": {"lat": 11.111, "lng": 11.111}, "from": "2019-04-05
13:45:00", "to": "2019-04-05 14:00:00"},
  {"location": {"lat": 22.222, "lng": -2.222}, "from": "2019-04-05
13:45:00", "to": "2019-04-05 14:00:00"},
  {"location": {"lat": -3.333, "lng": 33.333}, "from": "2019-04-05
15:45:00", "to": "2019-04-05 16:00:00"},
  {"location": {"lat": -4.444, "lng": -4.444}, "from": "2019-04-05
16:45:00", "to": "2019-04-05 17:00:00"},
  {"location": {"lat": 55.555, "lng": 55.555}, "from": "2019-04-05
18:45:00", "to": "2019-04-05 19:00:00"}
]
```

response

```
```

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

In the `result` returning an order in which 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"
```

errors

- 7 - Invalid parameters.
- 264 - Timeout not reached (too high api call rate).
- General types only.

Last update: November 25, 2020

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": 56.5,
        "lng": 60.5,
        "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>
}
```

- `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` - string date/time. When checkpoint created. *IGNORED* in checkpoint/create, checkpoint/update.

- `from` - string date/time. Date AFTER which checkpoint zone must be visited.
- `to` - string 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` - string enum. Checkpoint status. *IGNORED* in checkpoint/create, checkpoint/update.
- `status_change_date` - string 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` - string date/time. Wen 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` - array of int. List of tag ids.
- `form` - form object. If present.

API actions

API base path: `/task/checkpoint`.

create

Creates a new checkpoint.

required sub-user rights: `task_update`.

parameters

name	description	type
checkpoint	A `checkpoint` object without fields which are *IGNORED*.	JSON object

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/task/checkpoint/create'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "checkpoint":
{"tracker_id": 22, "label": "Shop", "description": "Buy things",
"parent_id": 1, "order": 0, "location": { "lat": 56.5, "lng":
60.5, "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

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/fsm/task/checkpoint/delete'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"checkpoint_id": 23144}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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. `[1,2]`.	array of int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/task/checkpoint/list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"checkpoint_ids": [1,2]}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/task/checkpoint/list?
hash=a6aa75587e5c59c32d347da438505fc3&checkpoint_ids=[1,2]
```

response


```
{
    "success": true,
    "list": [{
         "id": 111,
         "user_id": 3,
         "tracker_id": 22,
         "location": {
             "lat": 56.5,
             "lng": 60.5,
             "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/fsm/task/checkpoint/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"checkpoint_id": 111}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/task/checkpoint/read?
hash=a6aa75587e5c59c32d347da438505fc3&checkpoint_id=111
```

response

```
{
    "success": true,
    "value":  {
          "id": 111,
          "user_id": 3,
          "tracker_id": 22,
          "location": {
              "lat": 56.5,
              "lng": 60.5,
              "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/fsm/task/checkpoint/
transmute' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"checkpoint_id": 111}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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 `checkpoint` object without fields which are *IGNORED*.	JSON object

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/task/checkpoint/update'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "checkpoint":
{"id": 111, "label": "Shop", "description": "Buy things",
"parent_id": 1, "order": 0, "location": { "lat": 56.5, "lng":
60.5, "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}}'
```

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: November 25, 2020

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.

API actions

API path: `/task/form`.

Contains API calls related to forms associated with tasks.

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/fsm/task/form/create' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id":
11231, "template_id": 12548}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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/fsm/task/form/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id":
11231}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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".	string enum

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/task/form/download' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id":
11231, "format": "pdf"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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

- **form_template_id** (*integer, optional*). The returned list will contain forms, related to that template.
  warning: at least one of **form_template_id** and **task_ids** parameters must be not null.
- **task_ids** (*list of integers, optional*). Maximum size of list is 500 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 **form_template_id** and **task_ids** parameters must be not 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": , // a sequence of characters for
```

```
partial matching (against the name of the associated employee)
    "form_description": ,
    "form_label": ,
    "submit_address": ,
    "task_id": // strict match
    "task_address": ,
    "task_label": ,
    "tracker_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/fsm/task/form/list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "ascending":
"true", "include_unsubmitted": "true"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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/fsm/task/form/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "task_id":
"12546"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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 unsibmitted, associated with this form's fields.
    - `id` - int. File id.
    - `type` - string enum. Can be "image" or "file".
    - `created` - string date/time. Date when file created.
    - `uploaded` - string 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` - string 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: November 25, 2020

Updating task form values

Task form values can be submitted using web API only if there was a submission using Mobile Tracker App (Android / iOS). The use case is to "fix" incorrectly filled data. This action not intended to fill empty form from scratch.

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/fsm/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: November 25, 2020

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`.

Contains API calls which manipulate files attached to form's fields.

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/fsm/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/fsm/task/form/file?
hash=a6aa75587e5c59c32d347da438505fc3&task_id=11231&field_id=file1&siz
```

response

```
{
  "success": true,
  "value": {
    "file_id": 111,
    "url": "http://bla.org/bla",
    "expires": "2020-02-03 03:04:00",
    "file_field_name": "file1",
    "fields": {
      "token": "a43f43ed4340b86c808ac"
    }
  }
}
```

- `file_id` - int. This value will be submitted as form's field value.
- `url` - string. An url to which POST form-data with file contents should be executed.
- `expires` - string 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.
- `token` - string. Used for authentication of upload.

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

```
POST /bla HTTP/1.1
Host: bla.org
Content-Length: 1325
Origin: http://bla.org
... other headers ...
Content-Type: multipart/form-data; boundary=----
WebKitFormBoundaryePkpFF7tjBAqx29L

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

a43f43ed4340b86c808ac
------WebKitFormBoundaryePkpFF7tjBAqx29L
Content-Disposition: form-data; name="file";
filename="actual_file_name.png"
Content-Type: image/png

... contents of file goes here ...
------WebKitFormBoundaryePkpFF7tjBAqx29L--
```

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 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: November 25, 2020

Scheduling 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": 56.5,
        "lng": 60.5,
        "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` - array of int. List of tag ids.
- `form_template_id` - int. Form template id. Nullable.

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": 56.5,
        "lng": 60.5,
        "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` - array of int. 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` - array of int. 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` - array of int. Days of month on which tasks will
be created (1..31).
```

API actions

API base path: `task/schedule`.

create

Creates new task schedule entry.

required sub-user rights: `task_update`.

parameters

name	description	type
schedule	`schedule_entry` object without fields which are *IGNORED*.	JSON object

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/task/schedule/create' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "schedule":
{"tracker_id": 22, "location": {"lat": 56.5, "lng": 60.5,
"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/fsm/task/schedule/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"schedule_id": 23144}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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.	array of int
filter	Optional. Filter for task schedule label and description.	string

examples

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/task/schedule/list?
hash=a6aa75587e5c59c32d347da438505fc3
```

response


```
{
    "success": true,
    "list": [{
         "id": 111,
         "user_id": 3,
         "tracker_id": 22,
         "location": {
             "lat": 56.5,
             "lng": 60.5,
             "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/fsm/task/schedule/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id":
"12314"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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": 56.5,
            "lng": 60.5,
            "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	`schedule_entry` object without fields which are *IGNORED*.	JSON object

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/task/schedule/update' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "schedule":
{"tracker_id": 22, "location": {"lat": 56.5, "lng": 60.5,
"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: November 25, 2020

Scheduling routes

These actions allow creating scheduled routes similarly to regular routes.

API actions

API base path: `/task/schedule/route`.

create

Creates route schedule with checkpoints.

parameters

name	description	type
route	Route schedule entry without fields which are *IGNORED*.	JSON object
checkpoints	Array of route's checkpoints without fields which are *IGNORED*.	JSON object

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/task/schedule/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":
56.5, "lng": 60.5, "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.

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/fsm/task/schedule/route/
delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id": 23144}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/task/schedule/route/delete?
hash=a6aa75587e5c59c32d347da438505fc3&id=23144
```

response

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

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
	Route schedule entry without fields which are *IGNORED*.	JSON object
checkpoints	Array of route's checkpoints without fields which are *IGNORED*.	JSON object

example

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/task/schedule/create' \
    -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]}}, "checkpoint": {"id": 111, "tracker_id": 22, "label":
"Shop", "description": "Buy things", "parent_id": 1, "order": 0,
"location": { "lat": 56.5, "lng": 60.5, "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 }
```

Last update: November 25, 2020

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/fsm/task/schedule/
checkpoint/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"checkpoint_id": 11231}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/task/schedule/checkpoint/delete?
hash=a6aa75587e5c59c32d347da438505fc3&checkpoint_id=11231
```

response

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

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/fsm/task/schedule/
checkpoint/transmute' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"checkpoint_id": 11231}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/task/schedule/checkpoint/transmute?
hash=a6aa75587e5c59c32d347da438505fc3&checkpoint_id=11231
```

response

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

Last update: November 25, 2020

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.	array of int
from	Show tasks that will be created AFTER this date, e.g. "2014-07-01 00:00:00", should not before now	string date/ time
to	Show tasks will be created BEFORE this date, e.g. "2014-07-01 00:00:00", should not before `from`	string date/ time
filter	Optional. Filter for task schedule label and description.	string
types	Optional. Tasks or routes. For example: `["task", "route"]`	array of string enum

- 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/fsm/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": 56.5,
             "lng": 60.5,
             "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: November 25, 2020

Task history

Our platform tracks changes to task fields and state for your convenience.

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` - string date/time. Date when history event happened.
- `operation` - string 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/fsm/task/checkpoint/delete'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"checkpoint_id": 23144}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: November 25, 2020

APN settings

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 '+' sign.	string	"1234567890"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/apn_settings/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "phone":
"1234567890"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: October 1, 2020

Delivery info

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/fsm/delivery/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"external_id": "259876"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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/fsm/delivery/list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"external_id": "259876"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/delivery/list?
hash=a6aa75587e5c59c32d347da438505fc3&external_id=259876
```

response


```
{
    "success": true,
    "list": [{
        "task" : {
          "id": 111,
          "user_id": 3,
          "tracker_id": 22,
          "location": {
            "lat": 56.5,
            "lng": 60.5,
            "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": 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
          }]
        },
        "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: November 25, 2020

Geocoder

API path: `/geocoder`.

Geocoder types

- google.
- yandex.
- progorod.
- osm.
- locationiq.

API actions

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./ 60.0, 61.0"
lang	Language in which results should be.	string (enum)	"en"
geocoder	Optional. Geocoder type that will be preferably used for searching.	string (enum)	"google"
bounds	Optional. JSON object. The bounding box, specified by coordinates of northwest	bounds_object	`{"nw":{"lat":60.0,"lng":61.0},"se":`

name	description	type	format
	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.		`{"lat": 55.0,"lng": 60.0}}`
lang	Optional. ISO 639 language code.	locale	"en_US"
with_details	Optional. If `true` then the response will contain details.	boolean	`true`

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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": 56.26697,
            "lng": 19.55436,
            "address": "750 Avenue E,San Francisco",
            "details": {
              "country": "USA",
              "province": "CA",
              "locality": "San Francisco",
              "street": "Avenue E",
              "house": "750",
              "postcode": "94130",
              "bounds": {
                "nw":{
                  "lat": 62.23621,
                  "lng": 58.56997
```

```
                },
                "se":{
                  "lat": 31.98753,
                  "lng": 42.23694
                }
              }
            }
    }]
}
```

- `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: data types description section section).	location	`{"lat": , "lng": }`
geocoder	Optional. Geocoder type that will be preferably used for searching.	string (enum)	"google"
lang	Optional. ISO 639 language code.	locale	"en_US"

name	description	type	format
with_details	Optional. If `true` then the response will contain details.	boolean	`true`
goal	Helps to choose the target geocoder. Now supported `ui`, `ui_user_action`. Use `ui_user_action` for requests initiated by user, otherwise `ui`.	string (enum)	"ui"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/geocoder/
search_location' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "location":
"{"lat": 56.827001, "lng": 60.594296}}'
```

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": 62.23621,
          "lng": 58.56997
        },
        "se":{
          "lat": 31.98753,
          "lng": 42.23694
        }
      }
    }
}
```


- `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: October 1, 2020

Map layer

API path: `/map_layer`.

Map layer object structure:

```
{
    "id": 123456,
    "label": "test"
}
```

- `id` - int. Map layer entity ID.
- `label` - string. Map layer name.

API actions

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/fsm/map_layer/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id":
"123456"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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.

examples

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/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 **redirect_target** passed return redirect to `<redirect_target>?response=<urlencoded_response_json>`	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		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/fsm/map_layer/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "id":
"123456"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: October 23, 2020

Retranslator

API path: `/retranslator`.

CRUD actions for retranslators.

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

create

Creates new retranslator.

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

parameters

name	description	type
retranslator	Retranslator object without `id` field.	JSON object

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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
retranslator_id	ID of the retranslator that will be deleted.	int	123456

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/retranslator/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b",
"retranslator_id": "123456"}'
```

HTTP GET

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

HTTP GET

```
https://api.navixy.com/v2/fsm/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 `id` field.	JSON object

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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.

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/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: October 1, 2020

Tracking route

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. `[{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
provider_type	Optional. If not specified, the default user provider is used. One of "progorod", or "google", "osrm".	string enum

- `location` object described in data types description section.

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/route/get' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "start":
{"lat": 56.827001, "lng": 60.594296}, "end": {"lat": 52.835601,
"lng": 60.514721}}'
```

response

```
{
    "success": true,
    "distance": 2546,
    "time": 194,
    "list": [{"lat": 56.827001, "lng": 60.594296}, {"lat":
52.835601, "lng": 60.514721}],
    "key_points": [{
      "id": 123,
      "lat": 56.827,
      "lng": 60.594296
    }]
}
```


- `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: October 23, 2020

Tracking route Google

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.

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/route/google/get' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "start":
{"lat": 56.827001, "lng": 60.594296}, "end": {"lat": 52.835601,
"lng": 60.514721}}'
```

response

```
{
    "success": true,
    "distance": 13482,
    "time": 844,
    "list": [{"lat": 56.827001, "lng": 60.594296}, {"lat":
52.835601, "lng": 60.514721}],
```

```
    "key_points": [{
      "id": 123,
      "lat": 56.827,
      "lng": 60.594296,
      "distance": 482,
      "time": 144
      }]
}
```

- `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` - array of string enum. 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` - array of string enum. 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: October 23, 2020

Tracking route OSRM

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/fsm/route/osrm/get' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "start":
{"lat": 56.827001, "lng": 60.594296}, "end": {"lat": 52.835601,
"lng": 60.514721}}'
```

response

```
{
    "success": true,
    "distance": 2546,
    "time": 194,
    "list": [{"lat": 56.827001, "lng": 60.594296}, {"lat":
52.835601, "lng": 60.514721}],
```

```
    "key_points": [{
      "id": 123,
      "lat": 56.827,
      "lng": 60.594296
    }]
}
```

- `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` - string 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: October 23, 2020

Tracking route progorod

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. `[{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
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= `false` If it is `false` then use `mode=optimal` and use traffic=0, else `mode=comfort` and use traffic=1.	boolean

Where **location** described in data types description section. Order of waypoints may be changed.

response

```
{
    "success": true,
    "distance": 2546,
    "time": 194,
    "list": [{"lat": 56.827001, "lng": 60.594296}, {"lat":
52.835601, "lng": 60.514721}],
    "key_points": [{
      "id": 123,
      "lat": 56.827,
      "lng": 60.594296
    }]
}
```


- `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` - string enum. Type of error. One of: "not_set", "malformed" and "isolated".
- `point` - string 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: October 1, 2020

Status

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.

API base path: `/status/`

Status object structure

```
{
    "id": 5,
    "label": "Busy",
    "color": "E57373"
}
```

- `id` - int. A unique identifier of the status. Read-only.
- `label` - string. Human-readable label for the status.
- `color` - string. Hex-representation of RGB color used to display this status.

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 status listing. Read-only.
- `label` - string. Human-readable label for the status listing.
- `employee_controlled` - boolean. If `true` employees can change their own status, e.g. using mobile tracking app.
- `supervisor_controlled` - boolean. If `true` supervisors can change status, e.g. using mobile monitoring app.
- `entries` - array of int. List of IDs of statuses which belong to this listing. Order matters, and is preserved.

create

Creates new possible status for the specified status listing.

required sub-user rights: `tracker_update`

parameters

name	description	type
listing_id	ID of the listing for this status to attach to.	int
status	Status object without ID field.	JSON object

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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 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 statuses exceeded.

delete

Deletes status entry.

required sub-user rights: `tracker_update`

parameters

name	description	type
status_id	ID of the status belonging to authorized user.	int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/status/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "status_id":
"123"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/status/delete?
hash=a6aa75587e5c59c32d347da438505fc3&status_id=123
```

response

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

errors

- 201 (Not found in the database) – if 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 statuses belonging to the specified status listing.

parameters

name	description	type
listing_id	ID of the listing for this status to attach to.	int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/status/list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "listing_id":
"12345"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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 status properties.

required sub-user rights: `tracker_update`

parameters

name	description	type
status	Status object with ID field.	JSON object

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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 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: October 23, 2020

Tracker status

This resource contains methods to read and assign status of a particular tracker.

API base path: `/status/tracker/`

assign

Assign a 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 status. Must belong to status listing assigned to this tracker.	int	5

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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/fsm/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` - string date/time. Change date and time.
  - `origin` - string enum. Origin – who changed the status ("employee" or "supervisor").

errors

- 13 (Operation not permitted) – if status listing 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 listing 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.	array of int	`[123456, 234567]`

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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": 55.60920599,
                    "lng": 37.71843797,
                    "address": "Moscow, Orekhovyy Bul'var, 14a"
                },
                "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": 56.8267226,
                    "lng": 60.5947458,
                    "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` - string date/time. Date and time of change.
  - `origin` - string enum. Origin – who changed the status ("employee" or "supervisor").

errors

- 217 (Requested limit is too big) – limit is more than maxHistoryLimit.
- 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 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/fsm/status/tracker/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
"123456"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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` - string date/time. Date and time of change.
  - `origin` - string 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: October 23, 2020

Listing

API base path: `/status/listing/`

create

Creates new empty status listing.

required sub-user rights: `tracker_update`

parameters

name	description	type
listing	status_listing object without "id" and "entries" fields.	JSON object

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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 status listing.

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 listings exceeded.

delete

Deletes status listing.

required sub-user rights: `tracker_update`

parameters

name	description	type
listing_id	ID of the listing for this status to attach to.	int

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/status/listing/delete'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "listing_id":
"12345"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/status/listing/delete?
hash=a6aa75587e5c59c32d347da438505fc3&listing_id=12345
```

response

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

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.

list

Gets status listings belonging to authorized user.

examples

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/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 status listing properties.

required sub-user rights: `tracker_update`

`entries` field allows changing order of statuses attached to this listing.

parameters

name	description	type
listing	status_listing object with "id" and "entries" fields.	JSON object

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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 status 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.
- 262 (Entries list is missing some entries or contains nonexistent entries) – if entries does not contain full set of status IDs associated with this status listing, or if it contains nonexistent status IDs.

Last update: November 25, 2020

Tracker status listing

Contains api calls which link together trackers and status listings.

API base path: `/status/listing/tracker`

assign

Assigns a status listing (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 status listing. Omit this parameter completely, if you want remove the assignment.	int	12345

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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/fsm/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 listing 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: October 23, 2020

Track

API path: `/track`.

download

Downloads track points as KML/KMZ file for the specified track ID, 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 time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone).	string date/ time	`"2020-09-23 03:24:00"`
to	To time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone). Specified date must be after "from" date.	string date/ time	`"2020-09-23 06:24:00"`
track_ids	Optional. If specified, only points belonging to the specified tracks will be returned. If not, any valid track points between "from" and "to" will be returned.	array of int	`[123456, 234567]`
include_gsm_lbs	Optional. If `false` && track_ids not specified, GSM LBS points will be filtered out. Default= `true`.	boolean	`true`
point_limit	Optional. If specified, the returned track will be	int	`300`

name	description	type	format
	simplified to contain this number of points. Min=2, Max=3000. If not specified, the server settings to decimates track will be used.		
filter	Optional. If specified, the returned track will be filtered, applicable only for LBS tracks now.	boolean	`true`
format	File format, "kml" or "kmz", default is "kml".	string enum	`"kml"`
split	If `true`, split tracks by folders with start/end placemarks and track line. Default=`false`.	boolean	`false`

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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"}'
```

response

KML/KMZ file or JSON response if requested time period exceeds limit specified in a tracker's tariff:

```
{
    "success": true,
    "list": [],
    "limit_exceeded": 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.
- 211 (Requested time span is too big) – if interval between "from" and "to" is too big (maximum value specified in API config).

list

Gets a list of track descriptions for 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 time in `yyyy-MM-dd HH:mm:ss` format in user's timezone.	string date/ time	"2020-09-23 03:24:00"
to	To time in `yyyy-MM-dd HH:mm:ss` format in user's timezone. Specified date must be after "from" date.	string date/ time	"2020-09-23 06:24:00"
filter	Optional, default= `true`. If `true`, tracks which are too short (in terms of length and number of points) will be omitted from resulting list.	boolean	true
split	Optional, default= `true`. If `false`, all tracks will be merged into single one.	boolean	true
include_gsm_lbs	Optional, default= `true`. If `false`, GSM LBS tracks will be filtered out.	boolean	true

name	description	type	format
cluster_single_reports	Optional, default=`false`. If `true`, single point reports will be clustered by its coordinates.	boolean	false
count_events	Optional, default=`false`. If `true`, number of events occurred during each non-singlepoint track will be returned.	true	

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/track/list' \
    -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"}'
```

response

```
{
    "success": true,
    "limit_exceeded": false,
    "list": [{"track_info": }]
}
```

- `limit_exceeded` - boolean. `true` if the requested time period exceeds limit specified in a tracker's tariff.
- `list` - array of JSON objects. List of zero or more JSON objects.

where is either , , or :

`regular` object:

```
{
    "id": 123456
    "start_date": "2020-09-23 03:39:44",
    "start_address": "1255 6th Ave, New York, NY 10020, USA",
    "max_speed": 62,
    "end_date": "2020-09-23 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
}
```

- `id` - int. Track id.
- `start_date` - string 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 in km/h, e.g. 96.
- `end_date` - string 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. Number of events on this track. Field will be omitted if "count_events" is `false`.
- `norm_fuel_consumed` - float. A consumed fuel on track, litres. Field will be omitted if no vehicle bound to tracker or no normAvgFuelConsumption defined in a vehicle.
- `type` - string enum. Used to distinguish this track type from the others.
- `gsm_lbs` - optional boolean. GSM LBS point flag.

`single_report` object. Returned if device was creating reports in "interval" mode (e.g. M7 tracker in interval mode):

```
{
    "id": 123456,
    "type": "single_report",
    "start_date": "2020-09-24 03:39:44",
    "start_address": "1255 6th Ave, New York, NY 10020, USA",
    "avg_speed": 34,
    "gsm_lbs": false,
    "precision": 10
}
```

- `id` - int. Track id.
- `type` - string enum. Used to distinguish this track type from the others.
- `start_date` - string date/time. Point creation date, in user's timezone e.g. "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.
- `precision` - optional int. Location precision, meters.

`merged` object. Only returned if "split" is set to `false`:

```
{
    "start_date": "2020-09-24 03:39:44",
    "start_address": "1255 6th Ave, New York, NY 10020, USA",
    "max_speed": 62
    "end_date": "2020-09-24 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
}
```

- `start_date` - string 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 in km/h, e.g. 96.
- `end_date` - string 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. Number of events on this track. Field will be omitted if "count_events" is `false`.
- `norm_fuel_consumed` - float. A consumed fuel on track, litres. Field will be omitted if no vehicle bound to tracker or no normAvgFuelConsumption defined in a vehicle.
- `type` - string enum. Used to distinguish this track type from the others.
- `gsm_lbs` - optional boolean. GSM LBS flag.

`cluster` object. Only returned if "split" is set to `true`:

```
{
    "start_date": "2020-09-24 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": 56.829274,"lng": 60.597125}, {"lat":
56.829279,"lng": 60.597123}],
    "type": "cluster",
    "gsm_lbs": false
}
```


- `start_date` - string 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` - string date/time. Track end date, in user's timezone e.g. "2011-06-18 05:18:36".
- `precision` - optional int. Location precision, meters.
- `points` - array of points in a cluster.
- `type` - string enum. Used to distinguish this track type from the others.
- `gsm_lbs` - optional boolean. GSM LBS flag, true if cluster contains only GSM LBS points.

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).

read

Gets track points for the specified track ID, 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

name	description	type	format
from	From time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone).	string date/ time	"2020-09-23 03:24:00"
to	To time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone). Specified date must be after "from" date.	string date/ time	"2020-09-23 06:24:00"
track_id	Optional. If specified, only points belonging to the specified track will be returned. If not, any valid track points between "from" and "to" will be returned.	int	234567
include_gsm_lbs	Optional, default=`true`. If `false` && track_id not specified, GSM LBS points will be filtered out.	boolean	true
point_limit	Optional. If specified, the returned track will be simplified to contain this number of points. Min=2, Max=3000	int	3000
filter	Optional. If specified, the returned track will be filtered, applicable only for LBS tracks now.	boolean	false

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/track/read' \
    -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"}'
```

response

```
{
    "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 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` - string 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.

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).

Last update: November 23, 2020

Waybill

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 time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone).	string date/ time	"2020-09-23 03:24:00"
to	To time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone). Specified date must be after "from" date.	string date/ time	"2020-09-23 06:24:00"
filter	Optional, default= `true`. If `true`, tracks which are too short (in terms of length and number of points) will be omitted from resulting list.	boolean	true
split	Optional, default= `true`. If `false`, all tracks will be merged into single one.	boolean	false
include_gsm_lbs	Optional, default= `true`. If `false`, GSM LBS tracks will be filtered out.	boolean	false

name	description	type	format
cluster_single_reports	Optional, default=`false`. If `true`, single point reports will be clustered by its coordinates.	boolean	false
type	Should be one of "form3", "form3ext", "form4c".	string enum	"form4c"
fill_history	If `false`, only basic info about driver/garage/ vehicle will be filled (no trips or parkings).	boolean	false
fill_odometer	Optional, default=`false`. If `true`, 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"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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": "form3", "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: October 1, 2020

Waybill settings

API base path: `track/waybill/settings/`

read

Get last waybill number. Waybill number saved when new waybill had downloaded. If it had only digits, then it was incremented before saving.

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/track/waybill/settings/
read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: October 1, 2020

Working with trackers

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": 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 employee's department id, 0 if no department, -1 if no employee assigned. Read only
- `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`

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.	string	"fast.tmobile.com"
apn_ user		string	"tmobile"

name	description	type	format
	The user of GPRS APN of the sim card inserted into device.		
apn_password	The password of GPRS APN of the sim card inserted into device.	sting	"tmobile"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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

- 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).
- 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).

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/fsm/tracker/corrupt' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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/fsm/tracker/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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.

get_diagnostics

Gets last 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/fsm/tracker/
get_diagnostics' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/tracker/get_diagnostics?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

response

```
{
  "success": true,
  "user_time": "2014-07-09 07:50:58",
  "inputs": [
    {
      "label": "Sensor #1",
      "name": "can_fuel",
      "type": "fuel",
      "value": 100.0,
      "units_type": "litre",
      "units": "litres",
      "converted_units_type": null,
      "converted_value": null
    }
  ],
  "states": {
```

```
    "obd_vin": "123",
    "obd_mil_status": "false"
  },
  "update_time": "2014-03-06 13:57:00"
}
```

- `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` - string of enum. Name of sensor's raw input. E.g. "can_fuel" (see below list of values).
  - `type` - string of enum. Type of quantity, measured by a sensor. E.g. "fuel".
  - `value` - float. Reading's value, measured in units from an eponymous field. E.g. 100.0.
  - `units_type` - string of enum. Unit of measurement of input to the sensor. E.g."litre".
  - `units` - string. User label for sensor's units.
  - `converted_units_type` - string of 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`:

- **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_level**.
- **fuel_frequency**.
- **fuel_temperature**.
- **lls_temperature_x** (range for x: [1 – 16]).
- **lls_level_x** (range for x: [1 – 16]).

- **fuel_consumption**.
- **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).
- **rs232_x** (range for x: [1 – 6]).
- **board_voltage**.
- **can_engine_temp**.
- **can_engine_hours**.
- **can_mileage**.
- **can_throttle**.
- **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).
- **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]).
- **temp_sensor**.
- **ext_temp_sensor_x** (range for x: [1 – 10]).

List of state names for the field `states`:

- **obd_vin** (value type: string).
- **obd_mil_status** (value type: boolean).
- **obd_dtc_number** (DTC codes number; value type: integer).
- **obd_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).
- **vibration_state** (value type: boolean).
- **idling_state** (value type: boolean).
- **external_power_state** (connected/disconnected; value type: boolean).
- **case_intrusion_state** (value type: boolean).
- **driver_ident_state** (identified/not identified; value type: boolean).
- **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_hand_brake_state** (value type: boolean).
- **can_hood_state** (value type: boolean, `true` means "open").
- **can_airbag_state** (value type: boolean, `true` means "malfunction").
- **can_trunk_state** (value type: boolean, `true` means "open").
- **can_seat_belt_driver_state** (value type: boolean, `true` means "untied").
- **can_seat_belt_passenger_state** (value type: boolean, `true` means "untied").
- **can_door_state** (value type: boolean).

- **can_door_driver_state** (value type: boolean, `true` means "open").
- **can_door_passenger_state** (value type: boolean, `true` means "open").

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).

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/fsm/tracker/get_fuel' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/tracker/get_fuel?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

response


```
{
    "success": true,
    "user_time": "2014-07-09 07:50:58",
    "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": "2014-03-06 13:57:00"
}
```

- `user_time` - date/time. Current time in user's timezone.
- `inputs` - array of last readings of fuel-related sensors. Items are objects of the same type as used in tracker/get_diagnostics.
- `update_time` - date/time. Date and time when the data 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).

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/fsm/tracker/get_inputs' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/tracker/get_inputs?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

response

```
{
    "success": true,
    "user_time": "2014-07-09 07:50:58",
```

```
    "inputs": [true, true, false],
    "states": [
        {
            "type": "ignition",
            "name": "DIN1",
            "status": true,
            "input_number": 1
        }
    ],
    "update_time": "2014-03-06 13:57:00"
}
```

- `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` - string of 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

- 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).

get_counters

Gets last values of the tracker's 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	999119

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/tracker/get_counters' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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` - string of 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).

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/fsm/tracker/
get_last_gps_point' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/tracker/get_last_gps_point?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

response


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

- `value` - track point object.
    - `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. "" if no address recorded for the point.
    - `heading` - int. Direction bearing in degrees (0-360).
    - `speed` - int. Speed in km/h.
    - `precision` - int. Optional. Precision in meters.
    - `gsm_lbs` - boolean. Optional. `true` if location detected by GSM LBS, optional.
    - `parking` - boolean. Optional. `true` if point does not belong to track.

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/fsm/tracker/get_readings' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/tracker/get_readings?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

response


```
{
    "success": true,
    "user_time": "2014-07-09 07:50:58",
    "inputs": [{
      "label": "Sensor #1",
      "name": "can_fuel",
      "type": "fuel",
      "value": 100.0,
      "units_type": "litre",
      "units": "litres",
      "converted_units_type": null,
      "converted_value": null
    }]
}
```


- `user_time` - date/time. Current time in user's timezone.
- `inputs` - list of `sensor value` objects. See tracker/get_diagnostics.

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).

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/fsm/tracker/get_state' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/tracker/get_state?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=265489
```

response


```
{
    "success": true,
    "user_time":"2014-07-09 07:50:58",
    "state": {
        "source_id": 65894,
        "gps": {
            "updated": "2013-02-19 10:48:08",
            "signal_level": 25,
            "location": {
                "lat": 56.826068,
                "lng": 60.594338
            },
            "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"
            }
        }
    }
}
```

- `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.

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).

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.	array of int	`[999119, 999199]`

name	description	type	format
list_blocked	Optional. If `true` call returns list of blocked tracker IDs instead of error 208. Default is `false`.	boolean	true/false
allow_not_exist	Optional. If `true` call returns list of nonexistent tracker IDs instead of error 217 or 201. Default is `false`.	boolean	true/false

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/tracker/get_states' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "trackers":
"[999119, 999199]"}'
```

response


```
{
    "success": true,
    "user_time":"2014-07-09 07:50:58",
    "states": [{
      "source_id": 65894,
      "gps": {
        "updated": "2013-02-19 10:48:08",
        "signal_level": 25,
        "location": {
          "lat": 56.826068,
          "lng": 60.594338
        },
        "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": [123456],
    "not_exist": [234567]
}
```


- `user_time` - date/time. Current time in user's timezone.
- `states` - object. A map containing state object for requested trackers. 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. `true` to compact view. Default is `false`.	boolean	true/false
codes	Optional. Array of model codes. If passed only given models will be returned.	array of string	`[model_1, model_2, ...]`

examples

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/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", "analog_1"],
    "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. Does a switching LED supported by this tracker.
- `has_location_request` - boolean. Does the tracker have an opportunity to request a location with a command by SMS.
- `has_gprs_location_request` - boolean. Does the tracker have an opportunity to request a location with a command over a GPRS connection.
- `has_gsm_lbs_location_request` - boolean. Does the tracker have an opportunity to request a location by LBS with a command over a GPRS connection.
- `has_chat` - boolean. Does chat available for the device.
- `has_odometer` - boolean. Does the tracker have an integrated odometer.
- `has_lbs` - boolean. Does the tracker send information about cell info.
- `has_motion_sensor` - boolean. Does the tracker have an integrated motion sensor.
- `has_hardware_key` - boolean. Does the tracker have 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.

list

Gets user's trackers with optional filtering by labels.

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.	array of string	`["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/fsm/tracker/list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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.

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 **tag_binding** objects.	array of Json objects	`[{"tag_id" : 1, "ordinal" : 1}, {"tag_id" : 2, "ordinal" : 2}]`

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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 sms.	string	"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/fsm/tracker/
location_request' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"123456"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/tracker/location_request?
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).
- 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.

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 group/list.	int	0
imei	Tracker's IMEI.	string	"35645587458999"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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/fsm/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	Device ID that was used to register, e.g. IMEI. It can be used instead of `tracker_id` for models with a fixed ID.	string	"4568005588562"
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/fsm/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/fsm/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.
- 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).

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.

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 group/list.	int	0
model	A code of one of the supported models. See tracker/list_models.	string	"pt10"
plugin_id	An id of a registration plugin which will be used to register the device. See Registration plugins.	int	37
device_id	**Must** be specified if device model uses fixed device id. See tracker/list_models.	string	"4568005588562"

name	description	type	format
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 `null` will be used the platform settings. Default: `null`.	boolean	true/false
create_employee	If true, an employee will be created with name equal to label, phone number equal to `notification_phone`, and email equal to `notification_email`. If parameter is not specified or equals `null`, it is treated as false	boolean	true/false

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.

You can try to "auto-detect" APN settings by phone number using apn_settings/read API call.

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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/fsm/tracker/register?
hash=a6aa75587e5c59c32d347da438505fc3&label=Courier&group_id=0&plugin_
```

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.
- 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).

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

name	description	type	format
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/fsm/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.

Last update: December 17, 2020

Alarm mode for tracker

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/fsm/tracker/alarm_mode/
read' \
    -H 'Content-Type: application/json' \
    -d '{"tracker_id": "123456", "hash":
"a6aa75587e5c59c32d347da438505fc3"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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	`true` if alarm mode should be enabled.	boolean	true/ false

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/tracker/alarm_mode/set'
\
    -H 'Content-Type: application/json' \
    -d '{"tracker_id": "123456", "enabled": "true", "hash":
"a6aa75587e5c59c32d347da438505fc3"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: October 23, 2020

APN settings by tracker ID

API base path: `/tracker/apn_settings`

APN is short of Access Point Name and provides a device with the information needed to connect to wireless service. Using this call you can get APN settings by a tracker ID.

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/fsm/tracker/apn_settings/
read' \
    -H 'Content-Type: application/json' \
    -d '{"tracker_id": "123456", "hash":
"a6aa75587e5c59c32d347da438505fc3"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: October 23, 2020

Avatar for the tracker

API base path: `/tracker/avatar`

upload

Uploads avatar image for specified tracker. Then it will be available from `https://api.navixy.com/v2/fsm/[api_static_path]/tracker/avatars/<file_name>` e.g. `https://api.navixy.com/v2/fsm/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** or **image/pjpeg**
- **image/png**
- **image/gif**

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 redirect_target passed return redirect to ?response=.	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: October 23, 2020

Chat

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 `true`.	boolean	true/false

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/tracker/chat/list' \
    -H 'Content-Type: application/json' \
    -d '{"tracker_id": "123456", "hash":
"a6aa75587e5c59c32d347da438505fc3"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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.	array of int	`[999199, 999919]`

examples

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/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.	array of int	`[123,213]`

Use only one parameter.

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/tracker/chat/mark_read'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "message_id":
"123"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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	"Hello World"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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 authorized user and not be blocked. Max size - 300.	array of int	`[999199, 999919]`
message	Message text, not null, max size - 20000.	string	"Hello World"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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	Array of Ids of the tracker (aka "object_id"). Tracker must belong to authorized user and not be blocked. Max size - 300.	array of int	[999199, 999919]

examples

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/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: October 23, 2020

Contact

Deprecated

This API action deprecated and should not be used.

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/fsm/tracker/contact/list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: October 23, 2020

Datalogger

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: October 23, 2020

Engine immobilizer

API base path: `/tracker/engine_immobilizer`

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.

read

Request 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/fsm/tracker/
engine_immobilizer/read' \
    -H 'Content-Type: application/json' \
    -d '{"tracker_id": "123456", "hash":
"a6aa75587e5c59c32d347da438505fc3"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/tracker/engine_immobilizer/read?
tracker_id=123456&hash=a6aa75587e5c59c32d347da438505fc3
```

response

```
{
   "success": true,
   "enabled": true
}
```

- `enabled` - `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

Request 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	`true` if immobilizer should be enabled.	boolean	true/ false

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/tracker/
engine_immobilizer/set' \
    -H 'Content-Type: application/json' \
    -d '{"tracker_id": "123456", "enabled": "true", "hash":
"a6aa75587e5c59c32d347da438505fc3"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: October 23, 2020

Group

group

Tracker group is used to organize trackers in user interface. Currently, its function is purely visual. In FSM api, groups are read-only and represent departments of the employees assigned to trackers. Group id `0` means employee has no department, group id `-1` means tracker has no employee assigned.

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`

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/fsm/tracker/group/list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

HTTP GET

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

response

```
{
    "success": true,
    "list": [
        {
            "title": "test",
            "color": "FF6DDC",
            "id": 129301
        }
    ]
}
```

errors

General types only.

Last update: December 17, 2020

LED

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/fsm/tracker/led/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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, `true` – ON, `false` – OFF.	boolean	true

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/tracker/led/update' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489", "value": "true"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: October 23, 2020

Mobile app register

Deprecated

This API action deprecated and should not be used.

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 must exist. See group/list.	int	0
device_id	**Must** be specified if device model uses fixed device id. See tracker/list_models.	string	"4568005588562"

name	description	type	format
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 `null` will be used the platform settings. Default: `null`.	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: October 23, 2020

Output control

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. `[true, true, false]` means output 1 is on, output 2 is on, output 3 is off.	array of boolean	`[true, true, false]`

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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	`true` if the requested output should be enabled, or `false` if it should be disabled.	boolean	true

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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/fsm/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: October 23, 2020

Sensor readings

API base path: `/tracker/readings`

list

Gets last values for all metering sensors and state values. Includes CAN, OBD, and fuel.

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/fsm/tracker/readings/list'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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": "2019-03-16 11:15:19"
        }
  ],
  "states": [
    {
       "field": "obd_mil_status",
```

```
        "value": 12345.23,
        "update_time": "2019-03-16 11:15:19"
      }
    ]
}
```

- `states.value` - can be string, int, float, boolean, or null.

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: October 23, 2020

Retranslator

tracker_retranslator_binding is:

```
{
    "retranslator_id": 4548
    "fake_device_id": "AI568T"
}
```

- `retranslator_id` - int. An id of the retranslator.
- `fake_device_id` - string. Optional. If this field 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	Optional. If this field is set retranslator use it instead of real device ID to forward data.	string	"AI568T"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/tracker/retranslator/
bind' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489", "retranslator_id": "123"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/tracker/retranslator/bind?
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 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 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/fsm/tracker/retranslator/
list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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/fsm/tracker/retranslator/
unbind' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3", "tracker_id":
"265489", "retranslator_id": "123"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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 clone.

Last update: October 23, 2020

Trusted number

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/fsm/tracker/trusted_number/
list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "a6aa75587e5c59c32d347da438505fc3"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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 "+".

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.	array of string	`["496156680001", "496156680000"]`

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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: October 23, 2020

Unconfirmed commands

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/fsm/tracker/command/
unconfirmed/count' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
"123456"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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/fsm/tracker/command/
unconfirmed/reset' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
"123456"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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: October 23, 2020

Counter actions

API path: `/tracker/counter`.

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"]`.	string enum	"odometer"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsmtracker/counter/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
"123456", "type": "odometer"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsmtracker/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
type	Counter type. One of `["odometer", "fuel_consumed", "engine_hours"]`.	string 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/fsmtracker/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/fsmtracker/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 exist.
- if type value is not one of list above.

Last update: October 23, 2020

Counter Value

API path: `/tracker/counter/value`.

get

Gets 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"]`.	string enum	"odometer"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsmtracker/counter/value/
get' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
"123456", "type": "odometer"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsmtracker/counter/value/get?
hash=a6aa75587e5c59c32d347da438505fc3&tracker_id=123456&type=odometer
```

response

```
{
    "success": true,
    "value": 18.9
}
```

- `value` - float. The last valuer 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.

list

Get values for counters of passed `type` and `trackers`.

parameters

name	description	type	format
trackers	List of the tracker's Ids belonging to authorized user.	array of int	`[123456, 234567]`
type	Counter type. One of `["odometer", "fuel_consumed", "engine_hours"]`.	string enum	"odometer"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsmtracker/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.

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"]`.	string enum	"odometer"
value	A new value of counter.	float	233.21

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsmtracker/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.

Last update: October 23, 2020

Rule

A rule element consists of following fields:

Rule object

```
{
    "id": 1,
    "name": "rule",
    "description": "description",
    "zone_ids": [12, 15],
    "trackers": [123456, 234567],
    "type": "alarmcontrol",
    "primary_text": "ON",
    "secondary_text": "OFF",
    "param": 1,
    "alerts": {
      "sms_phones": ["98829991"],
      "phones": ["98829991"],
      "emails": ["example@test.com"],
      "push_enabled": true
    },
    "suspended": false,
    "schedule": [{
      "type":"weekly",
      "from":{"weekday":1,"time":"00:00:00"},
      "to":{"weekday":7,"time":"23:59:59"}],
    "extended_params": {
      "alarmcontrol": {
      "enabled": true,
      "sms": false,
      "call": false,
      "email": true,
      "push": true,
      "always_notify": false
    },
    "auto_created": true
}
```

- `id` - int. An id of a rule.
- `name` - string. Name of a rule.
- `description` - string. Rule's description.
- `zone_ids` - array of int. List of geofence ids.
- `trackers` - array of int. List of bound tracker ids.
- `type` - enum. One of pre-defined types of rules. See rule types.
- `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` - array of string. Phones for SMS notifications.
  - `phones` - array of string. Phones for voice calls.
  - `emails` - array of string. Emails for notifications.
  - `push_enabled` - boolean. If `true` push notifications available.
- `suspended` - boolean. `true` if the rule suspended.
- `shedule` - optional object. The rule will work in specified period.
- `extended_params` - optional. An object specified for concrete rule type. See rule types.
- `auto_created` - optional, boolean. `true` means that the rule created automatically.
- **schedule_interval** is one of:
  - **weekly_schedule_interval**

```
{
  "type": "weekly",
  "from": <weekday_time>,
  "to": <weekday_time>,
  "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
}
```

where **weekday_time** is:

```
{
    "weekday": 1,
    "time": "01:00:00"
}
```

- `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	format
rule_id	Id of a rule.	int	10
trackers	Ids of trackers. Trackers which do not exist, owned by other user or deleted ignored without errors.	`[999199, 999119]`	

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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

- **rule** - JSON object.

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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":"","append_zone_title":"","name":"Status
changing","trackers":[123456],"extended_params":
{"emergency":false,"zone_limit_inverted":false,"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 exists.

delete

Deletes rule with rule_id and all related objects from the database.

required sub-user rights: `tracker_rule_update`

parameters

name	description	type	format
rule_id	Id of a rule.	int	10

examples

cURL

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

HTTP GET

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

HTTP GET

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

response

```
{
   "success": true,
   "list": [{
        "id": 1,
        "name": "rule",
        "description": "description",
        "zone_ids": [12, 15],
        "trackers": [123456, 234567],
        "type": "alarmcontrol",
        "primary_text": "ON",
```

```
        "secondary_text": "OFF",
        "param": 1,
        "alerts": {
          "sms_phones": ["98829991"],
          "phones": ["98829991"],
          "emails": ["example@test.com"],
          "push_enabled": true
        },
        "suspended": false,
        "schedule": [{
          "type":"weekly",
          "from":{"weekday":1,"time":"00:00:00"},
          "to":{"weekday":7,"time":"23:59:59"}],
        "extended_params": {
          "alarmcontrol": {
          "enabled": true,
          "sms": false,
          "call": false,
          "email": true,
          "push": true,
          "always_notify": false
        },
        "auto_created": true
    }]
}
```


- `list` - list of rules

unbind

Unbinds trackers from rule with `rule_id`.

required sub-user rights: `tracker_rule_update`

parameters

name	description	type	format
rule_id	Id of a rule.	int	10
trackers	Ids of trackers. Trackers which do not exist, owned by other user or deleted ignored without errors.	`[999199, 999119]`	

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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

- **rule** - JSON object.

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/tracker/rule/update' \
    -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":"","append_zone_title":"","name":"Status
changing","trackers":[123456],"extended_params":
{"emergency":false,"zone_limit_inverted":false,"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: October 23, 2020

Rule types

Rule types with all parameters to create. The rule availability depends on the device and rule integration for it.

Common parameters

Common parameters exist in all rule types.

name	description	type
description	Rule's description.	string
primary_text	Primary text of rule notification.	string
alerts	Alerts object with destinations for notifications. Described in rule_object.	JSON object
suspended	`true` if the rule suspended.	boolean
name	Name of a rule.	string
trackers	List of bound tracker ids.	array of int
extended_params	Extended parameters for the rule. Described below.	JSON object
schedule	The rule will work in specified period. Described in rule_object.	JSON object
zone_ids	List of bound zones.	array of int

Common extended parameters

`extended_parameters` that used in all rule types.

name	description	type
emergency	If `true` enables emergency notification.	boolean
zone_limit_inverted	The rule tracked inside or outside zones. Default is: `false`.	boolean

Geofence entrance or exit

A rule that triggers on entering/exiting geofences.

parameters

name	description	type
type	Default `type`: "inoutzone".	string enum
secondary_text	Secondary text of rule notification.	string

extended parameters

name	description	type
append_zone_title	Show or not a label of zone in a notification.	boolean

Speed exceeding

A rule that triggers on speed exceeding.

parameters

name	description	type
type	Default `type`: "speedup".	string enum
param	Speed limit.	int

Parking state detection

A rule that triggers on detection of parking state.

parameters

name	description	type
type	Default `type` : "track_change".	string enum
secondary_text	Secondary text of rule notification.	string

Deviation from the route

A rule that triggers on deviations from the route.

parameters

name	description	type
type	Default `type` : "route".	string enum

extended parameters

name	description	type
allow_exit_at_endpoints	If `true` disables notifications on deviations from the start and end points of a route.	boolean

Work status change

A rule that triggers on status changing.

parameters

name	description	type
type	Default `type` : "work_status_change".	string enum

extended parameters

name	description	type
status_ids	List of tracked status ids.	array of int

Task performance

A rule that triggers on task status changes.

parameters

name	description	type
type	Default `type`: "work_status_change".	string enum

extended parameters

name	description	type
statuses	List of tracked statuses. Possible statuses are "arrived", "done","delayed", "failed".	array of string
on_form_submission	If `true` form submission will track.	boolean
on_repeated_form_submission	If `true` form resubmission will track.	boolean

Excessive idling (hardware related)

A rule that triggers on excessive idling registered by hardware.

parameters

name	description	type
type	Default `type`: "idling".	string enum
secondary_text	Secondary text of rule notification.	string

Excessive idling (platform related)

A rule that triggers on excessive idling registered by the platform.

parameters

name	description	type
type	Default `type`: "idling_soft".	string enum
secondary_text	Secondary text of rule notification.	string
param	Idle duration to send notification.	int

Fuel level change

A rule that triggers on a fuel level change.

parameters

name	description	type
type	Default `type`: "fuel_level_leap".	string enum
secondary_text	Secondary text of rule notification.	string

extended parameters

name	description	type
sensor_id	Id of tracked sensor.	int

Harsh driving

A rule that triggers on harsh driving.

parameters

name	description	type
type	Default `type`: "harsh_driving".	string enum

Pressing SOS button

A rule that triggers on SOS button pressing.

parameters

name	description	type
type	Default `type`: "sos".	string enum

Auto geofencing

A rule that triggers on auto geofencing.

parameters

name	description	type
type	Default `type`: "auto_geofence".	string enum

Fall detection

A rule that triggers on fall detection.

parameters

name	description	type
type	Default `type`: "g_sensor".	string enum

Unauthorized movement

A rule that triggers on unauthorized movement.

parameters

name	description	type
type	Default `type`: "parking".	string enum

Car crash

A rule that triggers on a car crash.

parameters

name	description	type
type	Default `type`: "crash_alarm".	string enum

Autocontrol related

Autocontrol related tracked rules.

parameters

name	description	type
type	Default `type`: "autocontrol".	string enum

extended parameters

name	description	type
alarmcontrol	Activation of car alarms. Described below	JSON object
battery_off	Disabling of external power supply. Described below	JSON object
door_alarm	Opening doors/trunk. Described below	JSON object
hood_alarm	Opening hood. Described below	

name	description	type
		JSON object
ignition	Ignition. Described below	JSON object
parking	Unauthorized movement. Described below	JSON object
gsm_damp	GSM-signal dumping (low signal level). Described below	JSON object
security_control	Switching ON/OFF security mode. Described below	JSON object

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
}
```

Advanced driver assistance systems (ADAS)

A rule that triggers on warnings from driver-assistance systems (ADAS).

parameters

name	description	type
type	Default `type` : "driver_assistance".	string enum

extended parameters

name	description	type
lane_departure_enabled	If `true` lane departure tracked.	boolean
forward_collision_enabled	If `true` lane departure tracked.	boolean
headway_warning_enabled	If `true` lane departure tracked.	boolean

name	description	type
peds_in_danger_zone_enabled	If `true` lane departure tracked.	boolean
peds_collision_warning_enabled	If `true` lane departure tracked.	boolean
traffic_sign_recognition_enabled	If `true` lane departure tracked.	boolean

Identification via RFID/iButton

A rule that triggers on a driver identification.

parameters

name	description	type
type	Default `type`: "driver_identification".	string enum
secondary_text	Secondary text of rule notification.	string

Driver change

A rule that triggers on driver change.

parameters

name	description	type
type	Default `type`: "driver_change".	string enum

Fatigue driving

A rule that triggers on fatigue driving.

parameters

name	description	type
type	Default `type`: "fatigue_driving".	string enum

Social distancing monitoring

A rule that triggers on social distancing violation.

parameters

name	description	type
type	Default `type`: "proximity_violation".	string enum
secondary_text	Secondary text of rule notification.	string

Tracker switched OFF or lost connection

A rule that triggers on device switch OFF and lost connection.

parameters

name	description	type
type	Default `type`: "offline".	string enum
secondary_text	Secondary text of rule notification.	string
param	Offline time to notification.	int

GSM signal dump

A rule that triggers on GSM signal dump.

parameters

name	description	type
type	Default `type`: "gsm_damp".	string enum

Low battery

A rule that triggers on low internal battery.

parameters

name	description	type
type	Default `type`: "lowpower".	string enum

Bracelet sensor

A rule that triggers on bracelet sensor opening/closing.

parameters

name	description	type
type	Default `type`: "bracelet".	string enum

Car alarm triggered

A rule that triggers on car alarm.

parameters

name	description	type
type	Default `type`: "alarmcontrol".	string enum

Tracker detach from the objects

A rule that triggers on a tracker detach from the object.

parameters

name	description	type
type	Default `type`: "detach".	string enum

External power cut

A rule that triggers on external power cut.

parameters

name	description	type
type	Default `type`: "battery_off".	string enum
secondary_text	Secondary text of rule notification.	string

Door opening in alarm mode

A rule that triggers on door opening in alarm mode.

parameters

name	description	type
type	Default `type`: "door_alarm".	string enum

Hood opening in alarm mode

A rule that triggers on hood opening in alarm mode.

parameters

name	description	type
type	Default `type`: "hood_alarm".	string enum

Location report on demand

A rule that triggers on location requests.

parameters

name	description	type
type	Default `type`: "location_response".	string enum

Connection/disconnection to the OBD2 port

A rule that triggers on connection/disconnection to the OBD2 port.

parameters

name	description	type
type	Default `type` : "obd_plug_unplug".	string enum
secondary_text	Secondary text of rule notification.	string

Tracker switch ON/OFF

A rule that triggers on tracker switch ON/OFF.

parameters

name	description	type
type	Default `type` : "on_off".	string enum
secondary_text	Secondary text of rule notification.	string

Locking/unlocking (padlock)

A rule that triggers on locking/unlocking(padlock).

parameters

name	description	type
type	Default `type` : "locking_unlocking".	string enum
secondary_text	Secondary text of rule notification.	string

Backup battery low

A rule that triggers on backup battery low.

parameters

name	description	type
type	Default `type`: "backup_battery_low".	string enum

Case intrusion

A rule that triggers on case intrusion.

parameters

name	description	type
type	Default `type`: "case_intrusion".	string enum

GPS signal lost/recover

A rule that triggers on GPS signal lost/recover.

parameters

name	description	type
type	Default `type`: "gps_lost_recover".	string enum
secondary_text	Secondary text of rule notification.	string

Padlock tampering

A rule that triggers on padlock tampering.

parameters

name	description	type
type	Default `type`: "strap_bolt".	string enum
secondary_text	Secondary text of rule notification.	string

Vibration sensor

A rule that triggers on vibration sensor.

parameters

name	description	type
type	Default `type`: "vibration".	string enum
secondary_text	Secondary text of rule notification.	string

Light sensor

A rule that triggers on a light sensor.

parameters

name	description	type
type	Default `type`: "light_sensor".	string enum
secondary_text	Secondary text of rule notification.	string

Call button pressed

A rule that triggers on call button pressing.

parameters

name	description	type
type	Default `type`: "call_button_pressed".	string enum

Tracker switched ON

A rule that triggers on tracker switch ON.

parameters

name	description	type
type	Default `type`: "poweron".	string enum

GPS antenna disconnected

A rule that triggers on GPS antenna disconnect.

parameters

name	description	type
type	Default `type`: "antenna_disconnect".	string enum

Check engine (MIL)

A rule that triggers on check engine (MIL) events.

parameters

name	description	type
type	Default `type`: "check_engine_light".	string enum

Inputs triggering.

A rule on inputs triggering.

parameters

name	description	type
type	Default `type`: "input_change".	string enum
secondary_text	Secondary text of rule notification.	string
param	Input number.	int

Outputs triggering

A rule on outputs triggering.

parameters

name	description	type
type	Default `type`: "output_change".	string enum
secondary_text	Secondary text of rule notification.	string
param	Output number.	int

Parameter in range

A rule that triggers on a parameter in range.

parameters

name	description	type
type	Default `type`: "sensor_range".	string enum
secondary_text	Secondary text of rule notification.	string

extended parameters

name	description	type
threshold	A threshold for a sensor.	int
sensor_id	Id of a tracked sensor.	int
min	A minimum range value.	int
max	A maximum range value.	int

Last update: November 25, 2020

Sensor actions

API base path: `/tracker/sensor`

sensor

Data types

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,
    "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
    }
}
```

- `id` - int. Sensor's id.
- `sensor_type` - string enum.
- `name` - string. A name of sensor.
- `input_name` - string.
- `divider` - double.
- `accuracy` - int.
- `units` - string.
- `units_type` - string 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. Array of int. 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.

Discrete input

```
{
  "type": "discrete",
  "id": 888951,
  "sensor_type": "ignition",
  "name": "Ignition",
  "input_number": 4
}
```

- `id` - int. An id of a sensor.
- `sensor_type` - string enum. Type of the sensor.
- `name` - string.
- `input_number` - int. Assigned input number.

batch_list

List tracker sensors bound to trackers with specified identificators (parameter `trackers`).

There exist a similar method for working with a single tracker - list.

parameters

Name	Description	Type
trackers	Set of tracker identificators. 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).	array of ints

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,
        "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	Sensor object.	JSON object

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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, "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/fsm/tracker/sensor/delete'
\
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
"123456", "sensor_id": "23456"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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/fsm/tracker/sensor/list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
"123456"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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,
    "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

name	description	type
sensor	Sensor object.	JSON object

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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, "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.

Last update: October 23, 2020

Sensor calibration data

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/fsm/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/fsm/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/fsm/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 LLSmonitor'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 LLSmonitor 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 LLSmonitor XML file).
- 235 – Missing calibration data (if there is no calibration data for the specified sensor number).

Last update: October 23, 2020

Input name

API base path: `/tracker/sensor/input_name`

list

Returns descriptions of all sensor inputs existing in the system.

examples

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/tracker/sensor/input_name?
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).

```
{
  "success": true,
  "list": [{
    "input_name": "analog_1",
    "description": "Sensor analógico #1"
  },
  {
    "input_name": "can_coolant_t",
    "description": "CAN: Temperatura del refrigerante"
  }]
}
```

errors

No specific errors.

Last update: October 5, 2020

Settings 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/fsm/tracker/settings' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
"123456"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/tracker/settings?
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 `<` and `>` symbols.	string	"Courier"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/tracker/update' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
"123456", "group_id": "1", "label": "Courier"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/tracker/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: October 23, 2020

LBS settings

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/fsm/tracker/settings/lbs/
read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
"123456"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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/fsm/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/fsm/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: October 23, 2020

Tracking mode

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/fsm/tracker/settings/
tracking/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
"123456"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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 tracking profiles for more information.	JSON object

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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: October 23, 2020

Tracking profiles

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 string 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` - string 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` - string 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 Castel 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 Castel MPIP-620, Castel PT-690, Castel 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` - string 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` - string enum. Can be "SCHEDULED" | "PERIODICAL". How device wakes up in `ASSET` mode. default="PERIODICAL".
- `wakeup_day` - string 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` - string 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` - string 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` - string 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` - string 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` - string 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_ps

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` - string enum. Can be "always_on" | "auto".
- `gsm_working_mode` - string 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 Castel 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` - string 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` string 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` - string 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` - string 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` - string 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` - string 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` - string 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` - string 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` - string 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` - string 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` - string 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=655350.

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` - string enum. Can be "disabled" | "engine" | "asset" | "hybrid".
- `track_by` - optional string 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_awax12

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` - string 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` – string enum. Can be "disabled" | "soft_sleep". * `stop_detection` – string 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` – string 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` - string 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` - string 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` - string 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 23, 2020

Trip detection

API base path: `/tracker/settings/trip_detection`

read

Gets trip 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/fsm/tracker/settings/
trip_detection/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
"123456"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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 trip 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
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/fsm/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/fsm/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: October 23, 2020

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	string enum	"electronic_lock_password"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/tracker/settings/
special/read' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "tracker_id":
"123456"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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` - string enum. Can be "enable" | "disable".
- `ext_input_mode` - string enum. Can be "enable" | "disable".
- `sos_button_mode` - string enum. Can be "enable" | "disable".

auto_geofence_telfm

```
{
    "type": "auto_geofence_telfm",
    "mode": "enable",
    "activation_timeout": 300,
    "radius": 50
}
```

- `mode` - string 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` - string 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` - string 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` - array of strings. 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` - string 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` - string 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` - string 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` - string 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_ql

```
{
    "type": "hbm_ql",
    "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` - string 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_ql

```
{
    "type": "hbm_ms_ql",
    "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` - string 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` - string 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` - string 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` - string enum. Can be "easy" | "normal" | "hard" | "hardest".

ign_src_suntech

```
{
    "type": "ign_src_suntech",
    "mode": "power_voltage",
    "power_voltage_low_level": 12000,
    "power_voltage_high_level": 19000
}
```

- `mode` - string 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` - string 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` - string enum. Can be "enable" | "disable".

scat_mayak_bt_control

```
{
    "type": "scat_mayak_bt_control",
    "function": "bt_disable",
```

```
    "bt_state": true
}
```

- `function` - string enum. Can be "bt_disable" | "bt_enable" | "bt_clear" | "bt_write".
- `bt_state` - boolean.

sos_key

```
{
    "type": "sos_key",
    "mode": "report",
    "phone": "55548875236"
}
```

- `mode` - string 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` - string 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` - string 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` - string 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` - string 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` - string 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.

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/fsm/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: October 23, 2020

Portlets

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` - string 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` - string 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 23, 2020

Engine hours

API base path: `/tracker/stats/engine_hours`

read

Returns engine hours (time when engine is on) count in 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 time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone).	string date/ time	"2020-09-24 03:24:00"
to	To time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone). Specified date must be after "from" date.	string date/ time	"2020-09-24 06:24:00"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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: October 23, 2020

Mileage

API base path: `/tracker/stats/mileage`

read

Returns mileage in kilometers in specified period grouped by trackers and day.

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 time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone).	string date/ time	"2020-09-24 03:24:00"
to	To time in `yyyy-MM-dd HH:mm:ss` format (in user's timezone). Specified date must be after "from" date.	string date/ time	"2020-09-24 06:24:00"

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/tracker/stats/mileage/
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,
  "result": {
    "<tracker_id>": {
      "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: October 23, 2020

Working with zones

Zones used in rules to limit rule area of activity. Also, zone names shown in reports after the address, if an event happened inside the zone.

This document describes CRUD actions for zones. Note that zone points handled separately because they are represented by big arrays of data.

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. Zone ID.
- `label` - string. Zone label.
- `address` - string. Zone address.
- `color` - string. Zone color in 3-byte RGB hex format.
- `radius` - int. Circle radius in meters.
- `center` - location object. Location of circle center.
- `tags` - Array of int. Array of tag IDs.

polygon:

```
{
    "id": 124597,
    "type": "polygon",
    "label": "Zone name",
    "address": "Karlsplatz, 2",
    "color": "27A9E3",
```

```
    "tags": [1,236]
}
```

- `id` - int. Zone ID.
- `label` - string. Zone label.
- `address` - string. Zone address.
- `color` - string. Zone color in 3-byte RGB hex format.
- `tags` - Array of int. Array of tag IDs.

sausage:

Represents all points within certain distance to the specified polyline.

```
{
    "id": 12345,
    "type": "sausage",
    "label": "Zone name",
    "address": "Karlsplatz, 2",
    "color": "27A9E3",
    "radius": 150,
    "tags": [289]
}
```

- `id` - int. Zone ID.
- `label` - string. Zone label.
- `address` - string. Zone address.
- `color` - string. Zone color in 3-byte RGB hex format.
- `radius` - int. Polyline radius in meters.
- `tags` - Array of int. Array of tag IDs.

API actions

API base path: `/zone`

batch_convert

Convert batch of tab-delimited circle zones and return list of checked zones 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 /data/spreadsheet/parse method.	string
fields	Optional, array of field names, default is `["label", "address", "lat", "lng", "radius", "tags"]`.	array of string enum
geocoder	Optional. Geocoder type.	string 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.

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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": "Zone name",
      "address": "Karlsplatz, 2",
      "color": "27A9E3",
      "radius": 100,
      "center": {
        "lat": 48.2009935,
        "lng": 16.3699642
      },
      "tags": []
    }],
```

```
    "limit_exceeded": false
}
```

- `id` - int. Zone ID.
- `label` - string. Zone label.
- `address` - string. Zone address.
- `color` - string. Zone color in 3-byte RGB hex format.
- `radius` - int. Circle radius in meters.
- `center` - location object. Location of circle center.
- `tags` - Array of int. Array of tag IDs.
- `limit_exceeded` - boolean, true if given batch constrained by limit

response with errors object

```
{
    "success": true,
    "list": [{
      "id": null,
      "label": "Zone 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 zone.

required sub-user rights: `zone_update`

parameters

name	description	type
zone	zone JSON-object without "id" and "color" fields.	JSON object
points	Array of new points for this zone. Must contain at least 3 elements. MUST be omitted if zone does not support points (e.g. circle)	array of `zone point` objects
zone.color	Optional. Zone color in 3-byte RGB hex format. Default is "27A9E3".	string

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/zone/create' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "zone":
{"label": "Zone name", "address": "zone address", "radius": 100,
"center": {"lat": 56.827001, "lng": 60.594296}}}'
```

response

```
{
    "success": true,
    "id": 1234567
}
```

- `id` - int. An id of the created zone.

errors

- 202 (Too many points in a zone) – max allowed points count for a zone is 100 for a polygon or 1024 for sausage.
- 230 (Not supported for this entity type) – if "points" were specified, but zone cannot have any points associated with it (e.g. if zone is circle).
- 268 (Over quota) – if the user's quota for zones exceeded.

delete

Deletes user's zone by `zone_id` or array of `zone_ids`.

required sub-user rights: `zone_update`

parameters

name	description	type	format
zone_id	Id of a zone.	int	1234567
zone_ids	Array of zone ids.	array of int	`[1234567, 2345678]`

- Use only one parameter `zone_id` or `zone_ids`.

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/zone/delete' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "zone_id":
"1234567"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/zone/delete?
hash=a6aa75587e5c59c32d347da438505fc3&zone_id=1234567
```

response

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

errors

- 201 (Not found in the database).
- 203 (Delete entity associated with).

response

```
{
    "success": false,
    "status": {
        "code": 203,
        "description": "Delete entity associated with"
    },
    "entities": [
        {
```

```
            "type": "rules",
            "ids": [12345, 23456]
        }
    ]
}
```

- `ids` - array of int. List IDs of the rules which uses the specified zone.

list

Gets all user zones.

examples

cURL

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

HTTP GET

```
https://api.navixy.com/v2/fsm/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. Zone objects without points field.

update

Update zone parameters for the specified zone. Note that zone must exist, must belong to the current user, and its type cannot be changed, e.g. if you already have a zone with ID=1 which type is "circle", you cannot submit a zone which type is "polygon".

required sub-user rights: `zone_update`

parameters

name	description	type
zone	zone JSON-object without "id" and "color" fields.	JSON object
zone.color	Optional. Zone color in 3-byte RGB hex format. Default is "27A9E3".	string

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/zone/update' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "zone":
{"label": "Zone name", "address": "zone address", "radius": 100,
"center": {"lat": 56.827001, "lng": 60.594296}}}'
```

response

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

errors

- 201 (Not found in the database) – if zone with the specified ID cannot be found or belongs to another user.
- 231 (Entity type mismatch) – if type of the submitted zone differs from type of the zone 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 `true` returns ready to create geofences or creates it and returns list of IDs otherwise. Default `true`.	boolean
redirect_target	Optional. URL to redirect. If **redirect_target** 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 zone) – max allowed points count for a zone 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 zones 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>
```

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: November 2, 2020

Zone point

API base path: `/zone/point`.

All actions to retrieve and manipulate points of the zone. Note that "circle" zone type cannot 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.

list

Get points of user's zone with `zone_id`.

parameters

name	description	type	format
zone_id	Id of a zone.	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/fsm/zone/point/list' \
    -H 'Content-Type: application/json' \
    -d '{"hash": "22eac1c27af4be7b9d04da2ce1af111b", "zone_id":
"1234567"}'
```

HTTP GET

```
https://api.navixy.com/v2/fsm/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 zone with the specified ID cannot be found or belongs to another user.
- 230 (Not supported for this entity type) – if zone cannot have any points associated with it (e.g. if zone is circle).

update

Update points for user's zone with `zone_id`.

required sub-user rights: `zone_update`

parameters

name	description	type
zone_id	Id of a zone.	int
points		

name	description	type
	Array of new points for this zone. Must contain at least 3 elements. Maximum number of points depends on zone type.	array of JSON objects

examples

cURL

```
curl -X POST 'https://api.navixy.com/v2/fsm/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 zone with the specified ID cannot be found or belongs to another user.
- 202 (Too many points in a zone) – if "points" array size exceeds limit for this zone type. Max allowed points count for a zone is 100 for a polygon or 1024 for sausage.
- 230 (Not supported for this entity type) – if zone cannot have any points associated with it (e.g. if zone is circle).

Last update: November 25, 2020

WebSocket API

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. Authorize with user/auth. This API method will return the hash you should use for all your next API calls.
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 what: * The subscription requests must contain the `hash` parameter obtained through user/auth action. * Responses and errors for the 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 a websocket using atmosphere-javascript looks like this:

```
    var request = {
        url: 'https://domain.com/event/subscription',
        contentType: "application/json",
        transport: 'websocket'
```

```
    };
    atmosphere.subscribe(request);
```

Executing this code will lead to send a request

```
ws://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 websocket. After what will be sent a first frame through 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: August 21, 2020

WebSocket Subscription

The *subscribe* and *unsubscribe* actions are used by client-side to subscribe on server events and unsubscribe from them. This actions are similar with any other API REST actions, but must be sending inside an open *WebSocket* channel and use only JSON format for the messages between client and server.

Subscribe Action

Request

Request parameters:

- **action** (text: *"subscribe"*).
- **hash** (required, string, length=32): session hash code gotten by user/auth action.
- **trackers** (required, int[], without nulls) - list of tracker ids for the events that require a subscription.
- **events** (required, enum[], without nulls) - list of events to subscribe. Event can be one of: `state`.

Request sample:

```
{
  "action": "subscribe",
  "hash": "f4bf1b75403d851653dad99c78c4b237",
  "events": ["state"],
  "trackers": [15564, 15565, 15568]
}
```

Response

Response parameters:

- **type** (required, text: *"response"*).
- **action** (required, text: *"subscription/subscribe"*).
- **events** (required, enum[], without nulls) - list of the subscribed events. Event can be one of: `state`.
- **data** (required, map, without nulls) - map with the events subscription result. One key on each subscribed event.

- **state** (presents if the "state" subscription requested, map) - the current status of requested trackers.

Keys is a tracker ids, values - one of the item:

- **normal** - non-blocked, normal status. The state events for this tracker will be delivered to client.
- **blocked** - tracker is blocked. The state events for this tracker will *not* be delivered to client.

The lifecycle events will be delivered. After unblocking, current tracker state will be sent automatically.

- **unknown** - tracker id is missed in database or not belong to current user.
- **disallowed** - subscription for this tracker is not allowed by current session.

Response sample:

```
{
  "type": "response",
  "action": "subscription/subscribe",
  "events": ["state"],
  "data": {
    "state": {
      "15564": "normal",
      "15565": "blocked",
      "15568": "unknown"
    }
  }
}
```

The "state" event subscription

After subscribe on the "state", server will send the current states of all non-blocked trackers to which the subscription was made. When changing the state of any tracker to which a subscription is made, the server will send a new state in event message.

Automatic subscriptions

- Subscribing to a *state* automatically creates a subscription to the lifecycle events.
- Subscribing to any event automatically creates a subscription to the logout events.

Unsubscribe Action

Request

Request parameters:

- **action** (text: *"unsubscribe"*).
- **hash** (required, string, length=32): session hash code gotten by user/auth action.
- **trackers** (required, int[], without nulls) - list of tracker ids for the events that require an unsubscription.
- **events** (required, enum[], without nulls) - list of events to unsubscribe. Event can be one of: `state`.

Request sample:

```
{
  "action": "unsubscribe",
  "hash": "f4bf1b75403d851653dad99c78c4b237",
  "events": ["state"],
  "trackers": [15568]
}
```

Response

Response parameters:

- **type** (required, text: *"response"*).
- **action** (required, text: *"subscription/unsubscribe"*).
- **events** (required, enum[], without nulls) - list of the unsubscribed events. Event can be one of: `state`.
- **data** (required, int[], without nulls) - list of the tracker ids from request.

Response sample:

```
{
  "type": "response",
  "action": "subscription/unsubscribe",
  "events": ["state"],
  "data": [15568]
}
```

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 user hash"
    },
    "data": {
        "events": ["state"],
        "trackers": [15564]
    }
}
```

Last update: August 20, 2020

WebSocket Events

The server sends an *event message* through the WebSocket channel when an event occurs and client has subscription on this. All of the event messages contains the fields:

- `type` - "event".
- `event` - one of the items: "state", "lifecycle", "logout"
- `data` (optional, object) - specific event payload.

State event

This messages are coming from server if client is subscribed. to the `state` events of the specific tracker and this tracker is not blocked. It's occur in the cases:

- Tracker stated is changed.
- Immediately after subscription.
- Immediately after unblocking.

Message fields:

- `type` - "event".
- `event` - "state".
- `data` - 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],
    "inputs_update": "2018-10-17 12:51:43",
    "outputs": [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"
  }
}
```


Note: `source_id` is not a `tracker_id`.

Lifecycle event

This messages are coming from server if client is subscribed. to the `state` events of the specific tracker. It's occur in the cases:

- Tracker is blocked.
- Tracker is unblocked.
- Tracker is corrupted (removed).

Message fields:

- `type` - "event".
- `event` - "lifecycle".
- `data` (required, object):
  - `source_id` - source id.
  - `lifecycle_event` - lifecycle event type. One of the items: "block", "unblock", "corrupt".

Message sample:


```
{
  "type": "event",
  "event": "lifecycle",
  "data": {
    "source_id": 123456,
    "lifecycle_event": "block"
```

```
    }
}
```

Logout event

These messages are coming from server if client is subscribed. to the any event. It's occur in 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: August 20, 2020

App: Delivery

Delivery is a special plugin which can be embedded to any other application or website and allow to 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 the 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, FFFFFF`.

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 to specify map scale by default. Parameter will be ignored with switched on autoscaling.

`map`:

- `roadmap` – Google
- `satellite` – Google satellite
- `osm` – Open Street map
- `doublegis` – 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: October 23, 2020

App: Web Locator

"Web Locator" is a special plugin which can be embedded to any other application or website and allow to 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 cartographical 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: December 17, 2020

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: July 30, 2020

Mobile tracker

Plugins are ready-to-use software extensions which can be embedded into 3-rd parties software or web-projects.

If you have more questions please contact our support team.

Last update: July 14, 2020

User aplications

Plugins are ready-to-use software extensions which can be embedded into 3-rd parties software or web-projects.

If you have more questions please contact our support team.

Last update: July 14, 2020