

# 9. ISP Go API

## Users

[**user\_id**] is the ISP's user ID. It can contain the characters A-Z, a-z, 0-9 underscore and dash.

The maximum length is 32 characters.

### Creating a user

A new user is created when trying to create a property for him: specify categories, assign ip.

```
POST /users/[user_id]/ip/["ip_address"]
```

### Deleting a user

```
DELETE /users/[user_id]
```

### Check a user existence:

```
HEAD /users/[user_id]
```

If a user is not found returns HTTP response code 404.

### Getting a list of active users

```
GET /active_users/
```

Response format:

```
["geek", "bugtest", "hammer"]
```

Active users are users which have at least one IP address.

### Getting a list of all users

```
GET /users
```

Response format:

```
[{
  "name": "geek",
  "safesearch": "off",
  "safeyoutube": "off",
  "status": "enabled",
  "filter": [1, 2],
  "ip": ["192.168.5.6", "2001:0db8:85a3:0000:0000:8a2e:0370:7334"],
  "whitelist": ["facebook.com", "google.com"],
  "blacklist": []
},{
  "name": "bugtest",
  "safesearch": "off",
  "safeyoutube": "on",
  "status": "enabled",
  "filter": [1, 2, 3],
  "ip": ["192.168.5.8"],
  "whitelist": ["reddit.com", "google.com"],
  "blacklist": []
}]
```

By default, returns first 100 users.

```
GET /users?start=100&stop=200
```

Start and stop parameters indicate that should be returned users with `user_id` from 100 to 200.

## Search a user

```
GET /search/user1
```

Response format:

```
[{
  "name": "user1",
  "safesearch": "off",
  "safeyoutube": "off",
  "status": "enabled",
  "filter": [],
  "ip": [],
  "whitelist": [],
```

```
"blacklist": []  
}]
```

The search is possible by IP or username. Wildcards or masks can be used in the search.

```
GET /search/192.168.0.*
```

Returns all users which IP address starts with `192.168.0.`

```
GET /search/user*
```

Returns all users which name starts with `user`

## GET user's information

```
GET /users/[user_id]
```

Response format:

```
{  
  "name": "[user_id]",  
  "safesearch": "off",  
  "safeyoutube": "off",  
  "status": "enabled",  
  "filter": [3, 4, 5, 6],  
  "ip": ["192.168.5.6", "2001:0db8:85a3:0000:0000:8a2e:0370:7334"],  
  "whitelist": ["reddit.com", "google.com"],  
  "blacklist": []  
}
```

## Update user's information

```
PUT /users/[user_id]
```

Content-type: application/json

```
{  
  "name": "[user_id]"  
  "safesearch": "on",  
  "safeyoutube": "off",
```

```
"status": "enabled",
"filter": [2, 3, 4, 5],
"ip": ["192.168.5.8", "2001:0db8:85a3:0000:0000:8a2e:0370:7334"],
"whitelist": ["docs.google.com", "atlassian.net"],
"blacklist" ["youtube.com", "pornhub.com"]
}
```

## Disable filtering

This feature allows users to suspend filtering without settings reset.

```
POST /users/[user_id]/status/disabled
```

```
Content-type: application/json
```

## Enable filtering

```
POST /users/[user_id]/status/enabled
```

```
Content-type: application/json
```

## SafeSearch

### Enabling safe search by default for a user

```
PUT /config/
```

```
Content-type: application/json
```

```
{ "safesearch": true }
```

### Disabling safe search by default for a user

```
PUT /config/
```

```
Content-type: application/json
```

```
{ "safesearch": false }
```

## Enabling safe search by default for users

```
PUT /userconfig/
```

```
Content-type: application/json
```

```
{ "safesearch": true }
```

## Disabling safe search by default for users

```
PUT /userconfig/
```

```
Content-type: application/json
```

```
{ "safesearch": false }
```

## Enabling safe Search for [user\_id]

```
POST /users/[user_id]/safesearch/on
```

```
Content-type: application/json
```

## Disabling safe Search for [user\_id]

```
POST /users/[user_id]/safesearch/off
```

```
Content-type: application/json
```

```
PUT /userconfig/
```

```
Content-type: application/json
```

```
{ "safeyoutube": true }
```

### Safe Youtube

If this option is enabled, the user will be automatically redirected to the safe version of YouTube.

## Enabling safe YouTube by default

```
PUT /config/
```

```
Content-type: application/json
```

```
{ "safeyoutube": true }
```

## Disabling safe YouTube by default

```
PUT /config/
```

```
Content-type: application/json
```

```
{ "safeyoutube": false }
```

## Enabling safe YouTube by default for users

```
PUT /userconfig/
```

```
Content-type: application/json
```

```
{ "safeyoutube": true }
```

## Disabling safe YouTube by default for users

```
PUT /userconfig/
```

```
Content-type: application/json
```

```
{ "safeyoutube": false }
```

## User addresses

### Get the user's IP address

```
GET /users/[user_id]/ip/
```

Response format:

```
[ "192.168.5.6", "2001:0db8:85a3:0000:0000:8a2e:0370:7334" ]
```

## Add user's IP address (will be added to existing)

```
POST /users/[user_id]/ip/
```

```
Content-type: application/json
```

```
[ "127.0.0.1" ]
```

or a list of IP addresses:

```
POST /users/[user_id]/ip/
```

```
Content-type: application/json
```

```
[ "127.0.0.1", "2001:0db8:85a3:0000:0000:8a2e:0370:7334" ]
```

## Update user's IP address (existing IP address will be replaced)

```
PUT /users/[user_id]/ip/
```

```
Content-type: application/json
```

```
[ "127.0.0.1" ]
```

or a list of IP addresses:

```
PUT /users/[user_id]/ip/
```

```
Content-type: application/json
```

```
[ "127.0.0.1", "2001:0db8:85a3:0000:0000:8a2e:0370:7334" ]
```

## Delete all user's IP addresses (disables filtering for the user)

```
DELETE /users/[user_id]/ip/
```

## Deleting one of the user's IP addresses

```
DELETE /users/[user_id]/ip/[127.0.0.1]
```

### General information

## Get the grouped list of categories with the names and identifiers of the categories

In order to get categories in the language other than the default one, you should add the HTTP header Accept-Language to the request with language indication (for example pt\_BR).

```
GET /categorygroups/
```

Response format:

```
[{
  "group": "Security",
  "categories": {
    "3": "Virus Propagation",
    "4": "Phishing",
    ...
  }
}, {
  "group": "Illegal Activity",
  "categories": {
    "6": "Drugs",
    ...
  }
},
...
]
```

## Get a list of categories with names and identifiers of categories

```
GET /categories/
```



Response format:

```
{
  "3": "Virus Propagation",
  "4": "Phishing",
  ...
}
```

## Check a website

```
GET /site/facebook.com
```

Response format:

```
{
  "domain": "facebook.com",
  "categories": [29],
}
```

## Categories for blocking

### Adding a category to the list of blocked by default

```
PUT /config/
```

```
Content-type: application/json
```

```
{ "filter": [1, 2] }
```

### Removing a category from the blocked by default

```
PUT /config/
```

```
Content-type: application/json
```

```
{ "filter": [] }
```

## Adding a category to the list of blocked by default for users

```
PUT /userconfig/
```

```
Content-type: application/json
```

```
{ "filter": [1, 2] }
```

## Removing a category to the list of blocked by default for users

```
PUT /userconfig/
```

```
Content-type: application/json
```

```
{ "filter": [] }
```

## Get a list of user's categories

```
GET /users/[user_id]/filter/
```

Response format:

```
[1, 2, 3]
```

## Add category to user's list

```
POST /users/[user_id]/filter/
```

```
Content-type: application/json
```

```
[1, 2]
```

## Save new user's list of categories

```
PUT /users/[user_id]/filter/
```

```
Content-type: application/json
```

```
[1, 2, 3]
```

## Disable one category for a user

```
DELETE /users/[user_id]/filter/2
```

## Disable all categories for a user

```
DELETE /users/[user_id]/filter/
```

### Black list

## Get the user's black list

```
GET /users/[user_id]/blacklist/
```

Response format:

```
[ "google.com", "facebook.com" ]
```

## Add a domain to the user's black list

```
POST /users/[user_id]/blacklist/
```

Content-type: application/json

```
[ "www.facebook.com" ]
```

## Replace to new black list

```
PUT /users/[user_id]/blacklist/
```

Content-type: application/json

```
[ "www.facebook.com" ]
```

## Remove a domain from the user's black list

```
DELETE /users/[user_id]/blacklist/facebook.com
```

## Delete the user's black list

```
DELETE /users/[user_id]/blacklist/
```

## Setting the "White list only" option

For the user to work in the "everything is prohibited except what is explicitly whitelisted" mode, you need to blacklist the root domain:

```
POST /users/[user_id]/blacklist/
```

```
Content-type: application/json
```

```
[ "- " ]
```

To delete the root domain from the blacklist:

```
DELETE /users/[user_id]/blacklist/-
```

## White list

### Get the user's white list

```
GET /user/[user_id]/whitelist/
```

Response format:

```
[ "google.com" ]
```

### Add domain to a user's white list

```
POST /users/[user_id]/whitelist/
```

```
Content-type: application/json
```

```
[ "google.com" ]
```

### Replace the user's white list

```
PUT /users/[user_id]/whitelist/
```

Content-type: application/json

```
[ "google.com" ]
```

## Remove a domain from the user's white list

```
DELETE /users/[user_id]/whitelist/google.com
```

## Delete the user's white list

```
DELETE /users/[user_id]/whitelist
```

### Global blacklist (takes precedence over user lists)

## Get the global black list

```
GET /blacklist
```

Response format:

```
[ "google.com" ]
```

## Add domain to global black list

```
POST /blacklist/
```

Content-type: application/json

```
[ "www.google.com" ]
```

## Replace to new global black list

```
PUT /blacklist/
```

Content-type: application/json

```
[ "www.google.com" ]
```

## Remove a domain from the global black list

```
DELETE /blacklist/google.com
```

## Delete the global black list

```
DELETE /blacklist/
```

Setting the option "The global whitelist only"

For the user to work in the "everything is prohibited except what is explicitly whitelisted" mode, you need to blacklist the root domain:

```
POST /blacklist/
```

```
Content-type: application/json
```

```
[ "-"]
```

To delete the root domain from the blacklist:

```
DELETE /blacklist/-
```

## Global white list

## Get the global white list

```
GET /whitelist/
```

Response format:

```
[ "google.com" ]
```

## Add a domain to the global white list

```
POST /blacklist/
```

```
Content-type: application/json
```

```
[ "www.google.com" ]
```

## Replace to new global white list

```
PUT /blacklist/
```

```
Content-type: application/json
```

```
[ "www.google.com" ]
```

## Remove a domain from the global white list

```
DELETE /whitelist/google.com
```

## Delete the global white list

```
DELETE /whitelist/
```

## Reports

### User activity by hour

Where `[date_from]`, `[date_to]` are dates in **YYYY-MM-DD** format.

```
GET /users/[user_id]/stat/activity/hour?from=[date_from]&to=[date_to]
```

```
{
  "labels": [ "2022-06-29 14:00:00", "2022-06-29 15:00:00" ],
  "datasets": [ {
    "label": "Requests",
    "data": [ 375, 275 ]
  }, {
    "label": "Blocks",
```

```
"data": [13, 0]
}]
}
```

Where:

**labels** - a list of timestamps,

**datasets** - a list of objects containing data for charts and legends,

**data** - a list of values corresponding to timestamps,

**label** - chart name (Requests - number of requests, Blocks - number of blocks).

## User activity by day

Where **[date\_from]**, **[date\_to]** are dates in **YYYY-MM-DD** format.

```
GET /users/[user_id]/stat/activity/day?from=[date_from]&to=[date_to]

{
  "labels": ["2022-06-29", "2022-06-30"],
  "datasets": [{
    "label": "Requests",
    "data": [5770, 3456]
  }, {
    "label": "Blocks",
    "data": [8, 9]
  }]
}
```

Where:

**labels** - a list of timestamps,

**datasets** - a list of objects containing data for charts and legends,

**data** - a list of values corresponding to timestamps,

**label** - chart name (Requests - number of requests, Blocks - number of blocks).

## Domains

```
GET /users/[user_id]/stat/domains/[filter]?from=[date_from]&to=[date_to]

{
  "labels": ["example.com", "google.com", "asdfg.com"],
```



```

"datasets": [{
  "label": "Requests",
  "data": [630, 474, 290]
},{
  "label": "NXdomain",
  "data": [0, 0, 290]
},{
  "label": "Blocks",
  "data": [630, 0, 0]
}]
}

```

Where:

[date\_from], [date\_to] - dates in **YYYY-MM-DD** format,

[filter] - can be set to all, www

all - returns all domains,

www - returns only domains start with www

labels - a list of domains,

datasets - a list of objects containing data for charts and legends,

data - a list of values corresponding to timestamps,

label - chart name (**Requests** - number of requests, **Blocks** - number of blocks, **NXdomain** - number of requests to non-existing domains).

## Detailed

```
GET /users/[user_id]/stat/detail?from=[date_from]&to=[date_to]
```

```

{
  "timestamps": ["2022-06-29 15:00:00", "2022-06-29 16:00:00"],
  "reports": [{
    "labels": ["clients4.google.com", "www.google.ru"],
    "cats": [[48, 251], [48]],
    "datasets": [{
      "label": "Requests",
      "data": [29, 20]
    }, {
      "label": "NXdomain",
      "data": [0, 0]
    }, {
      "label": "Blocks",

```

```

        "data": [0, 0]
    }
  ], {
    "labels": ["live.github.com", "lb._dns-sd._udp.0.0.200.10.in-addr.arpa",
"ssl.gstatic.com"],
    "cats": [[2], [2], [2]],
    "datasets": [{
      "label": "Requests",
      "data": [73, 48, 48]
    }, {
      "label": "NXdomain",
      "data": [0, 48, 0]
    }, {
      "label": "Blocks",
      "data": [0, 0, 0]
    }
  ]
}]
}

```

Where:

`[date_from]`, `[date_to]` - dates in **YYYY-MM-DD** format,

`timestamps` - a list of timestamps,

`reports` - a list of appropriate intervals reports,

`labels` - a list of domains,

`cats` - lists of categories for domains,

`datasets` - a list of objects containing data for charts and legends,

`data` - a list of values associated with each domain,

`label` - chart name (**Requests** - number of requests, **Blocks** - number of blocks, **NXdomain** - number of requests to non-existing domains).

## Raw statistics

Full user statistics for the last hour. Updates every 5 minutes.

```
GET /users/[user_id]/stat/raw
```

```

{
  "fields": ["created", "nxdomain", "address", "host", "blockedhost", "reason", "cats",
"blockedcats"],
  "data": [
    ["2016-07-11 17:50:26", "0", "10.200.1.88", "tpc.googlesyndication.com",

```

```
"tpc.googlesyndication.com", "4", [2], [0]],  
  ["2016-07-11 17:50:26", "0", "10.200.1.88", "tpc.googlesyndication.com",  
"tpc.googlesyndication.com", "4", [2], [0]]  
]  
}
```

## Response codes

**200** OK - Successful request

**201** Created - Successful element creation request

**204** No Content - Successful request with an empty response

**400** Bad Request - Invalid request

**404** Resource is not found - The resource does not exist

**422** Unprocessable Entity - The request does not contain the

**500** Internal server error

---

Revision #7

Created 15 September 2022 06:57:07

Updated 16 September 2022 12:42:25