# API

Our API is running at https://api.vpsfree.cz. The API can be used to perform most actions, just like you can do from the web interface. The only things that the API currently doesn't include are the management of user profiles (address, email, etc.) and data transfers.

In reality, the web interface running at https://vpsadmin.vpsfree.cz uses the API and calls it for every action.

## API Documentation

The API documentation, i.e. a list of objects, possible actions as well as input and output parameters can be found at https://api.vpsfree.cz/v3.0/.

Even without logging in, you can see a list of all objects, i.e. even those that only administrators can work with. On the top right, you can log in using the same credentials as in vpsAdmin. Afterwards, you will only see the objects, actions and parameters that the user who is currently logged in can work with.

## Working with the API

The API is built on the HaveAPI framework which we have developed. The framework creates a self-describing API, which means that premade generic clients can be used:

- Ruby - https://github.com/vpsfreecz/vpsfree-client
- PHP - https://github.com/vpsfreecz/haveapi-client-php
- JavaScript - https://github.com/vpsfreecz/haveapi-client-js
- The HaveAPI web interface - https://github.com/vpsfreecz/haveapi-webui
- A file system based on FUSE - https://github.com/vpsfreecz/haveapi-fs

Usage examples can always be found in the `README.md` of each client. In general, the client receives the API URL, then the client downloads the documentation and uses it for setup.

The API is RESTful, so any REST client can be used for simple actions. A description of the custom data transfer protocol that HaveAPI clients abstract can be found in the documentation.

## Authentication

There are two authentication methods available. The first and simpler one is HTTP basic. The name and password must be sent along with every API request. This is a good choice for one-off actions. However, if you need to call the API several times or automatically, storing the password on the disk or copying it constantly is not a good idea.

Another method is authentication using tokens. The way this method works is that the client first

requests a token to be created, one for which the client needs the name and password. As soon as the client receives the token, the name and password can be forgotten since it is the token that is then used for further authentication.

There can be several types of tokens with different lifetimes:

- fixed - token validity is fixed
- renewable_manual - token validity can be manually extended
- renewable_auto - token validity is extended after every request
- permament - the token is valid permanently, or until it is deleted

The type of token and time period by which its validity is extended is chosen by the client.

# CLI

The Ruby client also includes a CLI. In order for it to work properly, you need Ruby >= 2.0 and Ruby header files, OpenSSL and ncurses (mostly packages with the `-dev` or `-devel` suffix)

If you're using OS X, you first need to install OpenSSL using Homebrew and only then can you install EventMachine (a gem required by the client).

```
$ brew install openssl
$ sudo gem install eventmachine -- --with-opt-
include="/usr/local/opt/openssl/"
```

It can be installed using ruby gems:

```
$ gem install vpsfree-client
```

## Installing in Windows 10 Using a Ubuntu Linux Subsystem

### Installing Ubuntu (Windows 10 Subsystem for Linux)

1. in Windows 10, enable Developer Mode, let it install
2. in the Programs and Features menu, open Turn Windows Features On or Off, scroll to the very bottom, choose Windows Subsystem for Linux, run the installation and let the computer restart
3. after the restart, run bash in the Start menu as admin
4. create a Unix username and password
5. press the Y key and let the Ubuntu base be installed

### Installing Dependencies

```
sudo apt-get install ruby2.0 ruby2.0-dev libssl-dev make g++
```

### A Quick & Dirty Fix to Set Ruby2.0 as Default Instead of 1.9

```
sudo rm /usr/bin/ruby /usr/bin/gem /usr/bin/irb /usr/bin/rdoc /usr/bin/erb
sudo ln -s /usr/bin/ruby2.0 /usr/bin/ruby
sudo ln -s /usr/bin/gem2.0 /usr/bin/gem
sudo ln -s /usr/bin/irb2.0 /usr/bin/irb
sudo ln -s /usr/bin/rdoc2.0 /usr/bin/rdoc
sudo ln -s /usr/bin/erb2.0 /usr/bin/erb
sudo gem update --system
sudo gem pristine --all
```

Source: http://blog.costan.us/2014/04/restoring-ruby-20-on-ubuntu-1404.html

### Installing vpsfree-client

```
sudo gem install vpsfree-client
```

---

After the installation, `vpsfreectl` should be available in $PATH. If not, this can be fixed easily:

```
$ gem env | grep "EXECUTABLE DIRECTORY"
```

This command will display the location of all executables installed using `gem`. All you need to do is add that folder to $PATH, e.g.:

```
$ PATH="$PATH:/home/user/.gem/ruby/2.0.0/bin"
```

## Usage

```
$ vpsfreectl --help
Usage: vpsfreectl [options] <resource> <action> [objects ids] [--
[parameters]]
    -u, --api URL                    API URL
    -a, --auth METHOD                Authentication method
        --list-versions              List all available API versions
        --list-auth-methods [VERSION]
                                     List available authentication methods
        --list-resources [VERSION]   List all resource in API version
        --list-actions [VERSION]     List all resources and actions in API
version
        --version VERSION            Use specified API version
    -c, --columns                    Print output in columns
    -H, --no-header                  Hide header row
    -L, --list-parameters            List output parameters
    -o, --output PARAMETERS          Parameters to display, separated by a
comma
```

```
    -r, --rows                      Print output in rows
    -s, --sort PARAMETER            Sort output by parameter
        --save                      Save credentials to config file for
later use
        --raw                       Print raw response as is
        --timestamp                 Display Datetime parameters as
timestamp
        --utc                       Display Datetime parameters in UTC
        --localtime                 Display Datetime parameters in local
timezone
        --date-format FORMAT        Display Datetime in custom format
    -v, --[no-]verbose              Run verbosely
        --client-version            Show client version
        --protocol-version          Show protocol version
        --check-compatibility       Check compatibility with API server
    -h, --help                      Show this message

Commands:
vps remote_console VPS_ID           Open VPS remote console

Available resources:
auth_token
cluster
cluster_resource
dataset
dataset.snapshot
dataset.plan
dataset_plan
dns_resolver
environment
environment.config_chain
environment.dataset_plan
integrity_check
integrity_fact
integrity_object
ip_address
location
mail_log
mail_recipient
mail_template
mail_template.recipient
node
os_template
pool
snapshot_download
transaction_chain
transaction_chain.transaction
user
user.environment_config
user.cluster_resource
user.state_log
```

```
user_session
vps
vps.state_log
vps.config
vps.feature
vps.ip_address
vps.mount
vps.console_token
vps_config
```

By choosing an authentication method, object or action, help will have new options available:

```
$ vpsfreectl --auth basic --help
$ vpsfreectl vps --help
$ vpsfreectl vps list --help
```

## Authentication

### HTTP Basic

You can either pass the username and password using the `--username` and `--password` parameters or you can skip the parameters and the program will prompt the user to input the credentials.

```
$ vpsfreectl --auth basic user current
```

### Tokens

Authentication using tokens creates a few new options:

```
$ vpsfreectl --auth token --help
    ...
    -a, --auth METHOD               Authentication method
    -s, --save                      Save credentials to config file for
later
        --username USER             User name
        --password PASSWORD         Password
        --token TOKEN               Token
        --token-lifetime LIFETIME   Token lifetime, defaults to
renewable_auto
        --token-interval SECONDS    How long will token be valid in seconds
        --new-token                 Request new token
        --token-via VIA             Send token as a query parameter or in
HTTP header (query_param, header)
    ...
```

Again, you don’t have to send the username and password as parameters, the program will request

them.

One-time token:

```
$ vpsfreectl --auth token user current
```

After finishing, the token is forgotten and next time you will need to request a new one. A token for a use like this makes no sense.

Saving the token:

```
$ vpsfreectl --auth token --save user current
```

The token is saved in the `~/.haveapi-client.yml` file and automatically loaded upon the next startup.

```
$ vpsfreectl user current
```

However, a token like this expires within 20 minutes and it is necessary to request a new one. It is possible to create a token with a longer validity right away:

```
$ vpsfreectl --auth token --save --token-interval $((24*60*60)) user current
# 1 day
```

Or a token with a permanent lifetime straight away:

```
$ vpsfreectl --auth token --save --token-lifetime permanent user current
```

Be careful if you're using tokens with an unlimited validity since anybody with access to `~/.haveapi-client.yml` can access and use the token.

## Actions and Parameters

Objects and actions have the same names as they do in the documentation – they're just written in lowercase letters and underscores are used instead of spaces. Every action has its input and output parameters. These can be displayed using the `--help` parameter if you provide the names of both the object and the action.

```
$ vpsfreectl vps list --help
...
Action description:
List VPS

Input parameters:
        --offset OFFSET             The offset of the first object
        --limit LIMIT               The number of objects to retrieve
        --node NODE                 Filter by node
        --location LOCATION         Filter by location
```

```
        --environment ENVIRONMENT    Filter by environment
        --os-template OS_TEMPLATE    OS template
        --object-state OBJECT_STATE  Object state
    -h, --help                       Show this message

Output parameters:
    id
    user                             VPS owner
    hostname                         VPS hostname
    os_template
    dns_resolver                     DNS resolver the VPS will use
    node                             Node VPS will run on
    dataset                          Dataset the VPS resides in
    created_at
    memory                           Minimally 1024, maximally 12288, step
size is 128
    swap                             Minimally 0, maximally 12288, step size
is 128
    cpu                              Minimally 1, maximally 8, step size is
1
    maintenance_lock
    maintenance_lock_reason
    object_state
    expiration_date                  A date after which the state will
progress
    running
    process_count
    used_memory                      in MB
    used_disk                        in MB
...
```

You can filter VPSs by server, location, environment, distro, state and limit the number of displayed items or pages.

Action parameters are separated from client parameters by two dashes `--`. The following command displays the first three VPSs:

```
$ vpsfreectl vps list -- --limit 3
```

If, for example, you want to filter by location, you first have to display a list of locations:

```
$ vpsfreectl location list
 ID  Label
  3  Praha
  4  Brno
  5  Playground
```

Now let’s display the VPSs found in the “Praha” (Prague) location:

```
$ vpsfreectl vps list -- --location 3
```

## Output Formatting

The output can be either formatted into columns or rows. Under default settings, columns are used to display a group of objects and lines are used for a single object. You can choose the format using the `-c` option, `--columns` for columns and `--rows` for rows.

An example of the output formatted in columns (in this case the `--columns` option is unnecessary – it will be chosen automatically since the output contains several objects):

```
$ vpsfreectl vps list --columns
 ID  Label
  3  Praha
  4  Brno
  5  Playground
```

An example of the output formatted into rows:

```
$ vpsfreectl location list --rows
    ID:  3
 Label:  Praha

    ID:  4
 Label:  Brno

    ID:  5
 Label:  Playground
```

When used in scripts, the column formatting could be problematic due to the header with parameter names. You can disable it using the `-H`, `--no-header` option so that it won't be displayed.

## Choosing the Parameters to Display

The `-o`, `--output` option is used to set what output parameters of the action and in which order they will be displayed. Parameter names are separated by a comma.

```
$ vpsfreectl vps list -o id,hostname,node,os_template
      VPS id:  4710
    Hostname:  vps
        Node:  node7.prg (#108)
 OS template:  CentOS 7 (#43)
```

## Sorting

The `-s`, `--sort` option can be used to sort the output in ascending order according to a specific parameter (on the client's side).

```
$ vpsfreectl os_template list --sort label
```

```
 ID  Label                  Info   Supported
 42  Arch Linux [TEST]      -      0
 24  CentOS 6               -      1
 43  CentOS 7               -      1
 20  Debian 6               -      1
 31  Debian 7               -      1
 38  Debian 7 [TEST]        -      0
 46  Debian 8               -      1
 33  Fedora 20              -      1
 40  Fedora 22              -      1
 14  Gentoo 13.0            -      1
 37  Gentoo [TEST]          -      0
 32  OpenSUSE 12.3          -      1
 26  Scientific Linux 6.6   -      1
 45  Scientific Linux 7     -      0
 30  Ubuntu 12.04           -      1
 35  Ubuntu 14.04           -      1
 39  Ubuntu 14.04 [TEST]    -      0
 47  openSUSE 13.2 [TEST]   -      0
```

### Date and Time Format

The client includes several options that are used to choose the date and time format if an action returns a parameter of the type `Datetime`.

- `--utc`
- `--localtime`
- `--timestamp`
- `--date-format FORMAT` - format according to Time#strftime

## Updates

In order to make use of new features, the client needs to be updated from time to time. This is done using the following command:

```
$ gem update vpsfree-client
```

Then you can remove the old version:

```
$ gem cleanup vpsfree-client
```

This command only removes the old versions of the `vpsfree-client` gem, but its dependencies remain. In order to remove all old gems, use the following command:

```
$ gem cleanup -n  # prints which gems would be deleted
$ gem cleanup  # deletes the gems
```

# File system

haveapi-fs is a file system based on FUSE, i.e. it makes it possible to mount the API as a file system from userspace. Objects, actions and their parameters can be browsed like folders and files in any program. The file system is installed the same way as a CLI, it just has a different name:

```
$ gem install haveapi-fs
```

## Usage

The usage of haveapi-fs is described in detail in README.md. This is just a brief description.

```
$ haveapi-fs https://api.vpsfree.cz /mnt/api.vpsfree.cz
Username: mylogin
Password:

$ cd /mnt/api.vpsfree.cz
$ ls -1
auth_token
cluster
cluster_resource
dataset
dataset_plan
dns_resolver
environment
help.html
help.man
help.md
help.txt
ip_address
language
location
mail_template
migration_plan
node
object_history
os_template
snapshot_download
transaction
transaction_chain
user
user_session
vps
vps_config
```

Each folder contains the files help.{html,txt,md,man}, which describe what the current folder contains.

```
$ cat vps/5685/hostname
my-vps

$ echo better-hostname > vps/5685/hostname
$ echo 1 > vps/5685/save
$ cat vps/5685/actions/update/status
1
$ cat vps/5685/hostname
better-hostname
```

The example demonstrates that individual object attributes can simply be changed by writing into a file. The save action can be called by writing a one, or it can be run as an executable file.

In order to get clearer creation/modification of objects, you can use the YAML files `create.yml` or `update.yml`. These files contain parameters in the form of a hash and after saving and closing the file, the corresponding action is performed.

This way, you can perform any operation in the API.

# Usage Examples

The following scripts demonstrate ways you can use the API. These scripts can be run from anywhere – manually, from cron, etc.

Although these examples are still valid, the CLI client already contains commands which make downloading backups even easier.

## Daily Downloads of VPS Backups

```
#!/usr/bin/env ruby
# Downloading the last (most recent) backup of all VPSs.
require 'vpsfree/client'

api = VpsFree::Client::Client.new
api.authenticate(:token, token: 'PUT YOUR TOKEN HERE')

downloads = []

api.dataset.list(role: :hypervisor).each do |ds|
  last_snapshot = ds.snapshot.list.last

  dl = api.snapshot_download.create(snapshot: last_snapshot.id)
  unless dl.api_response.ok?
    warn "#{ds.name}@#{last_snapshot.created_at}:
#{dl.api_response.message}"
    next
  end
```

```
  downloads << dl.id
end

# vpsAdmin is now creating a tar.gz with the backup and it will take a while
until it can be
# downloaded. There are regular checks for the “ready” parameter for all
archives.
puts "Waiting for the downloads to be ready..."
loop do
  sleep(5*60)

  downloads.delete_if do |id|
    dl = api.snapshot_download.find(id)
    unless dl.api_response.ok?
      warn "Download #{id} has failed"
      next(true)  # remove from the list
    end
    if dl.ready
      puts "Downloading #{id} to #{dl.file_name} (#{dl.size / 1024} GB)"
      `wget "#{dl.url}"`
      dl.delete
      true
    else
      false
    end
  end
  break if downloads.empty?
end

puts "Done"
```

## Daily NAS Backups with a Weekly History

```
#!/usr/bin/env ruby
# Daily NAS snapshotting with a 7-day history
require 'vpsfree/client'

api = VpsFree::Client::Client.new
api.authenticate(:token, token: 'PUT YOUR TOKEN HERE')

api.dataset.list(role: :primary).each do |ds|
  # Creating a new snapshot
  ds.snapshot.create

  # Naive waiting until the snapshot is created. A different option would be
checking
  # the state of the corresponding transaction chain.
  sleep(10)

  # Deleting older snapshots
```

```
  t = Time.now - 7 * 24 * 60 * 60
  snapshots = ds.snapshot.list(meta: {count: true})
  deleted = 0
  snapshots.each do |snap|
    # Keep at least 7 snapshots
    break if snapshots.total_count - deleted < 7
    # Deleting redundant snapshots older than 7 days
    if snap.created_at < t
      puts "Delete #{ds.name}@#{snap.created_at}"
      snap.delete
      deleted += 1

      # Once again, naive waiting until the action finishes.
      sleep(30)
    end
  end
end
```