vk

Release 3.0

Dmitry Voronin

Jul 04, 2022

CONTENTS

VK library is a vk.com (the largest Russian social network) python API wrapper

Contents:

CHAPTER

ONE

INSTALLATION

Project is available on PyPI and Github and can be easily installed using pip

1.1 Install from PyPI

```
pip install vk
```

1.2 Install from Github

Warning: Although the version on GitHub is the latest, it may not be stable!

```
pip install git+https://github.com/voronind/vk
```

CHAPTER

TWO

VK.COM API

2.1 Getting access

To use vk.com API you need an access token. There are several types of tokens, and the official documentation describes the process of obtaining each of them well. Based on our experience, we can offer you the fastest way to get a token - from the official application

Steps:

1. Sign up in social network
2. Go to https://vkhost.github.io and choose any application (I prefer vk.com)
3. Grant access to your account and copy *access_token* parameter from URL

Pros:

- You can use methods that are prohibited by unofficial applications (*messages* section for example)

Cons:

- The token has a certain lifetime (~12 hours)

2.2 Making API request

To make request to vk.com API we need send GET or POST HTTP request to address https://api.vk.com/method/METHOD with parameters of specific method, access token, version and other parameters (see official documentation for more details). This module is needed in order to protect you from raw HTTP requests and provide a convenient interface for making requests.

CHAPTER

THREE

USAGE

3.1 API method request example

Several types of APIs are implemented in this module. Each of them is needed for certain purposes, but they are all united by the way of accessing the VK API. After initializing the class, you can call any method. Let's try to figure out what's going on here:

```
>>> import vk
>>> api = vk.API(access_token='...', v='5.131')
>>> print(api.users.get(user_ids=1))
[{'id': 1, 'first_name': '', 'last_name': '', ... }]
```

It gets user info with **user id** equal to **1**. `vk.api.APINamespace` object is used to create API request and send it via original *vk.session.API* class object (or another), which in turn, manages access token, sends API request, gets JSON response, parses and returns it.

More formally, this forms the following POST request to the VK API:
https://api.vk.com/method/users.get?user_ids=1&access_token=...&v=5.131

3.2 vk.API

class vk.session.**API**(**args*, ***kwargs*)

The simplest VK API implementation. Can process any API method that can be called from the server

Parameters

- **access_token** (*Optional[str]*) – Access token for API requests obtained by any means (see *documentation*). Optional when using *InteractiveMixin*
- ****kwargs** (*any*) – Additional parameters, which will be passed to each request. The most useful is *v* - API version and *lang* - language of responses (see *documentation*)

Example

```
>>> import vk
>>> api = vk.API(access_token='...', v='5.131')
>>> print(api.users.get(user_ids=1))
[{'id': 1, 'first_name': '', 'last_name': '', ... }]
```

3.3 vk.UserAPI

class vk.session.**UserAPI**(**args*, ***kwargs*)

Subclass of *vk.session.API*. It differs only in that it can get access token using user credentials (Implicit flow authorization).

> **Warning:** This implementation uses the web version of VK to log in and receive cookies, and then obtains an access token through Implicit flow authorization. In the future, VK may change the approach to authorization (for example, replace it with VK ID) and maintaining operability will become quite a difficult task, and most likely it will be **deprecated**. Use *vk.session.DirectUserAPI* instead

Parameters

- **user_login** (*Optional[str]*) – User login, optional when using *InteractiveMixin*
- **user_password** (*Optional[str]*) – User password, optional when using *InteractiveMixin*
- **client_id** (*Optional[int]*) – ID of the application to authorize with, defaults to "VK Admin" app ID
- **scope** (*Optional[Union[str, int]]*) – Access rights you need. Can be passed comma-separated list of scopes, or bitmask sum all of them (see official documentation). Defaults to 'offline'
- ****kwargs** (*any*) – Additional parameters, which will be passed to each request. The most useful is *v* - API version and *lang* - language of responses (see *documentation*)

Example

```
>>> import vk
>>> api = vk.UserAPI(
...     user_login='...',
...     user_password='...',
...     scope='offline,wall',
...     v='5.131'
... )
>>> print(api.users.get(user_ids=1))
[{'id': 1, 'first_name': '', 'last_name': '', ... }]
```

get_auth_check_code()

Callback to retrieve authentication check code (if account supports 2FA). Default behavior is to raise exception, redefine in a subclass

Returns

The authentication check code can be obtained in the sent SMS, using Google Authenticator (or another authenticator), or it can be one of ten backup codes

3.4 vk.DirectUserAPI

class vk.session.**DirectUserAPI**(**args*, ***kwargs*)

Subclass of *vk.session.UserAPI*. Can get access token using user credentials (through Direct authorization).

See also:

Necessary data (**client_id** and **client_secret**) from other official applications

Parameters

- **user_login** (*Optional[str]*) – User login, optional when using *InteractiveMixin*
- **user_password** (*Optional[str]*) – User password, optional when using *InteractiveMixin*
- **client_id** (*Optional[int]*) – ID of the official application, defaults to *"VK for Android"* app ID
- **client_secret** (*Optional[str]*) – Client secret of the official application, defaults to client secret of *"VK for Android"* app
- **scope** (*Optional[Union[str, int]]*) – Access rights you need. Can be passed comma-separated list of scopes, or bitmask sum all of them (see official documentation). Defaults to 'offline'
- ****kwargs** (*any*) – Additional parameters, which will be passed to each request. The most useful is *v* - API version and *lang* - language of responses (see *documentation*)

Example

```
>>> import vk
>>> api = vk.DirectUserAPI(
...     user_login='...',
...     user_password='...',
...     scope='offline,wall',
...     v='5.131'
... )
>>> print(api.users.get(user_ids=1))
[{'id': 1, 'first_name': '', 'last_name': '', ... }]
```

3.5 vk.CommunityAPI

class vk.session.**CommunityAPI**(**args*, ***kwargs*)

Subclass of *vk.session.UserAPI*. Can get community access token using user credentials (Implicit flow authorization for communities). To select a community on behalf of which to make request to the API method, you can pass the **group_id** param (defaults to the first community from the passed list)

> **Warning:** This implementation uses the web version of VK to log in and receive cookies, and then obtains an access tokens through Implicit flow authorization for communities. In the future, VK may change the approach to authorization (for example, replace it with VK ID) and maintaining operability will become quite a difficult task, and most likely it will be **deprecated**.
>
> You can create a group token on the management page: Community -> Management -> Working with API -> Access Tokens -> Create a token (bonus - the token has no expiration date)

Parameters

- **user_login** (*Optional[str]*) – User login, optional when using *InteractiveMixin*
- **user_password** (*Optional[str]*) – User password, optional when using *InteractiveMixin*
- **group_ids** (*List[int]*) – List of community IDs to be authorized
- **client_id** (*Optional[int]*) – ID of the application to authorize with, defaults to "VK Admin" app ID
- **scope** (*Optional[Union[str, int]]*) – Access rights you need. Can be passed comma-separated list of scopes, or bitmask sum all of them (see official documentation). Defaults to None. **Be careful**, only *manage*, *messages*, *photos*, *docs*, *wall* and *stories* are available for communities
- ****kwargs** (*any*) – Additional parameters, which will be passed to each request. The most useful is *v* - API version and *lang* - language of responses (see *documentation*)

Example

```
>>> import vk
>>> api = vk.CommunityAPI(
...     user_login='...',
...     user_password='...',
...     group_ids=[123456, 654321],
...     scope='messages',
...     v='5.131'
... )
>>> print(api.users.get(user_ids=1))
[{'id': 1, 'first_name': '', 'last_name': '', ... }]
>>> print(api.users.get(group_id=654321, user_ids=1))
[{'id': 1, 'first_name': '', 'last_name': '', ... }]
```

CHAPTER

FOUR

ADVANCED USAGE

4.1 Errors handling

The easiest way to catch an error is using the *try-except* statement:

```
import vk
from vk.exceptions import VkAPIError

api = vk.API(access_token='Invalid access token')

try:
    user = api.users.get(user_ids=1)
except VkAPIError as e:
    print(e)

# 5. User authorization failed: invalid access_token (4).. request_params = { ... }
```

Class *vk.exceptions.VkAPIError* provides basic functionality for error processing

exception vk.exceptions.**VkAPIError**(*error_data*)

Class to represent a VK API error

Parameters
error_data (*dict*) – Parsed JSON object of error

method

The method whose call resulted in an error. Relevant only for errors that occurred during the *execute* method

Type
Union[str, None]

code

Error code. To conveniently determine the type of error, you can use *vk.exceptions.ErrorCodes* enumeration class

Type
int

message

A message explaining the nature and/or cause of the error

Type
str

request_params

Dictionary (param-value) of request parameters that were passed to the API method

Type
dict

redirect_uri

The link you need to click to pass validation. *None* for all errors except 17

Type
Union[str, None]

captcha_sid

Captcha SID. *None* for all errors except 14

Type
Union[str, None]

captcha_img

Link to the image to be solved. *None* for all errors except 14

Type
Union[str, None]

For a simpler definition of the type of error, you can use the *vk.exceptions.ErrorCodes*

class vk.exceptions.**ErrorCodes**(*value*)

Enumeration object of VK API error codes. See official documentation for more details

AUTHORIZATION_FAILED = 5

Invalid access token

PERMISSION_IS_DENIED = 7

No rights to perform this action

CAPTCHA_NEEDED = 14

Need to enter the code from the image (Captcha)

ACCESS_DENIED = 15

No access to call this method

INVALID_USER_ID = 113

Invalid user ID or user deactivated

```
import vk
from vk.exceptions import ErrorCodes, VkAPIError

api = vk.API(access_token='Invalid access token', v='5.131')

try:
    user = api.users.get(user_ids=1)
except VkAPIError as e:
    print(e.code == ErrorCodes.AUTHORIZATION_FAILED)

# True
```

4.2 Global errors handling

Some errors can occur in any request and handling them every time the method called will be a very difficult task, so you can define a global handler for each error

APIBase.**on_api_error**(*request*)

Default API error handler that handles all errros and raises them. You can add a handler for a specific error by redefining it in your class and appending the error code to the method name. In this case, the redefined method will be called instead of *on_api_error()*. The *vk.exceptions.VkAPIError* object can be obtained via `request.api_error`

Parameters
request (*vk.api.APIRequest*) – API request object

Example

```
import vk

class API(vk.APIBase):
    def on_api_error_1(self, request):
        print('An unknown error has occurred :(')

api = API()
```

For some popular errors, the *vk.session.API* already has its own handlers, for example, for processing captcha:

API.**on_api_error_14**(*request*)

Captcha error handler. Retrieves captcha via *API.get_captcha_key()* and resends request

API.**get_captcha_key**(*api_error*)

Callback to retrieve CAPTCHA key. Default behavior is to raise exception, redefine in a subclass

Parameters
api_error (vk.exceptions.VkAPIError) – Captcha error that occurred

Returns
Captcha solution (a short string consisting of lowercase letters and numbers)

4.3 Connection parameters

You can specify additional connection parameters in each API implementation: *timeout*, which specifies the time to complete the request (default is **10**) and *proxy*, which specifies which proxy to use (default is **None**).

```
import vk

api = vk.API(
    ...
    timeout=5,
    proxy='socks5://127.0.0.1:9050'
)
```

4.4 Interactive

class vk.session.**InteractiveMixin**

Mixin that receives the necessary data from the console

Example

```
import vk
from vk.session import InteractiveMixin

class API(InteractiveMixin, vk.API):
    pass

api = API()

# The input will appear: `VK API access token: `
```

CHAPTER

FIVE

CONTRIBUTION

This section will be useful for first-time contributors

5.1 Setup environment

For setup/build/test we use tox, so you need to install it first

```
git clone https://github.com/voronind/vk
cd vk

pip install virtualenv
virtualenv venv
source venv/bin/activate

pip install tox
```

To prevent unformatted code from commit, we recommend adding a pre-commit hook or execute `tox -e fix` yourself before each commit

```
pip install pre-commit
pre-commit install
```

5.2 Tox targets

```
tox             # To run tests
tox -e docs     # To build documentation
tox -e fix      # To format files
```

5.3 Testing

Since some test suites use real calls to the VK API, you should create the necessary data for this and save it as environment variables. Also don't forget to add them to your repository secrets

Varibale	Description	How to get
VK_ACCESS_TOKEN	Access token for VK API. We reccomend to use community token, because it doesn't have an expiration date	Create your community, go to its settings (API section), create API key with messages scopes
VK_USER_LOGIN	Login from the VK account. For rare tests, you can use your account, otherwise we recommend to register a test one	-
VK_USER_PASSWORD	Password from the VK account. For rare tests, you can use your account, otherwise we recommend to register a test one	-
VK_GROUP_IDS	IDs of communities in which you have admin rights	Create *N* your communities, copy their IDs and pass it to env var as comma-separated list

5.4 Logging

It's very useful for the module's debug to include logs to better understand what is happening

```
import vk
import logging

logging.basicConfig()
logging.getLogger('vk').setLevel(logging.DEBUG)
```

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

INDEX