Package ‘vkR’

October 12, 2022

Type Package
Title Access to VK API via R
Description Provides an interface to the VK API [https://vk.com/dev/methods]. VK [https://vk.com/] is the largest European online social networking service, based in Russia.

Version 0.2
Date 2020-09-25
Maintainer Dmitriy Sorokin <dementiy@yandex.ru>

URL https://github.com/Dementiy/vkR

BugReports https://github.com/Dementiy/vkR/issues

License GPL-3

Encoding UTF-8

Depends R (>= 3.0.0)

Imports graphics, httr, purrr, jsonlite, stats, utils, XML

Suggests stringr, tm, plyr, dplyr, reshape2, jpeg, igraph, rgexf, httpuv, mongolite

RoxygenNote 7.1.1


NeedsCompilation no

Author Dmitriy Sorokin [aut, cre],
        Anton Antonov [ctb]

Repository CRAN

Date/Publication 2020-09-29 05:20:02 UTC
R topics documented:

- age_predict .................................................. 4
- areFriends ..................................................... 4
- boardGetComments ........................................... 5
- boardGetCommentsExecute .................................. 6
- boardGetCommentsList ....................................... 7
- clear_text .................................................. 7
- collection_exists ............................................ 8
- create_empty_collection .................................... 8
- databaseGetChairs .......................................... 9
- databaseGetCities .......................................... 9
- databaseGetCitiesById ...................................... 10
- databaseGetCountries ....................................... 11
- databaseGetCountriesById .................................. 11
- databaseGetFaculties ....................................... 12
- databaseGetRegions ......................................... 13
- databaseGetSchoolClasses .................................. 13
- databaseGetSchools ......................................... 14
- databaseGetSchoolsById ..................................... 15
- databaseGetSchoolsById ..................................... 15
- db_drop .................................................. 16
- db_drop_collection ......................................... 16
- db_getActive ................................................ 17
- db_getName ................................................ 17
- db_get_collection .......................................... 17
- db_get_connection .......................................... 18
- db_init .................................................. 18
- db_insert ................................................ 18
- db_load .................................................. 19
- db_load_collection ........................................ 19
- db_metaConnection ........................................ 20
- db_save .................................................. 20
- db_update ................................................ 21
- execute ................................................ 22
- filterAttachments ........................................... 22
- getAccessToken ............................................. 23
- getArbitraryNetwork ......................................... 23
- getCountryByCityId ......................................... 23
- getEgoNetwork ............................................. 24
- getFriends ................................................ 24
- getFriendsBy25 ............................................ 25
- getFriendsFor ............................................... 26
- getGroups ................................................ 26
- getGroupsById ............................................. 27
- getGroupsForUsers ......................................... 28
- getGroupsMembers ........................................... 28
- getGroupsMembersExecute ................................... 29
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>getMutual</td>
<td>30</td>
</tr>
<tr>
<td>getMutualExecute</td>
<td>31</td>
</tr>
<tr>
<td>getPaths</td>
<td>32</td>
</tr>
<tr>
<td>getStatus</td>
<td>32</td>
</tr>
<tr>
<td>getTopics</td>
<td>33</td>
</tr>
<tr>
<td>getTopicsExecute</td>
<td>34</td>
</tr>
<tr>
<td>getURLs</td>
<td>35</td>
</tr>
<tr>
<td>getUsers</td>
<td>36</td>
</tr>
<tr>
<td>getUsersExecute</td>
<td>41</td>
</tr>
<tr>
<td>getWall</td>
<td>46</td>
</tr>
<tr>
<td>getWallExecute</td>
<td>47</td>
</tr>
<tr>
<td>get_stop_words</td>
<td>49</td>
</tr>
<tr>
<td>groupsSearch</td>
<td>49</td>
</tr>
<tr>
<td>handle_captcha</td>
<td>50</td>
</tr>
<tr>
<td>handle_validation</td>
<td>51</td>
</tr>
<tr>
<td>has_error</td>
<td>51</td>
</tr>
<tr>
<td>likesGetList</td>
<td>51</td>
</tr>
<tr>
<td>likesGetListForObjects</td>
<td>52</td>
</tr>
<tr>
<td>me</td>
<td>53</td>
</tr>
<tr>
<td>messagesGet</td>
<td>54</td>
</tr>
<tr>
<td>messagesGetHistory</td>
<td>55</td>
</tr>
<tr>
<td>messagesGetHistoryAll</td>
<td>55</td>
</tr>
<tr>
<td>messagesGetHistoryExecute</td>
<td>56</td>
</tr>
<tr>
<td>messagesSend</td>
<td>57</td>
</tr>
<tr>
<td>messagesSplitByDate</td>
<td>58</td>
</tr>
<tr>
<td>newsfeedSearch</td>
<td>58</td>
</tr>
<tr>
<td>or</td>
<td>59</td>
</tr>
<tr>
<td>postGetComments</td>
<td>60</td>
</tr>
<tr>
<td>profile_fields</td>
<td>61</td>
</tr>
<tr>
<td>queryBuilder</td>
<td>61</td>
</tr>
<tr>
<td>repeat_last_query</td>
<td>62</td>
</tr>
<tr>
<td>request_delay</td>
<td>62</td>
</tr>
<tr>
<td>saveAsGEXF</td>
<td>62</td>
</tr>
<tr>
<td>search.getHints</td>
<td>63</td>
</tr>
<tr>
<td>setAccessToken</td>
<td>63</td>
</tr>
<tr>
<td>setAPIVersion</td>
<td>64</td>
</tr>
<tr>
<td>setRepeats</td>
<td>64</td>
</tr>
<tr>
<td>setTimeout</td>
<td>64</td>
</tr>
<tr>
<td>show_collections</td>
<td>64</td>
</tr>
<tr>
<td>show_dbs</td>
<td>65</td>
</tr>
<tr>
<td>tag2Id</td>
<td>65</td>
</tr>
<tr>
<td>try_handle_error</td>
<td>66</td>
</tr>
<tr>
<td>try_handle_network_error</td>
<td>66</td>
</tr>
<tr>
<td>usersGetFollowers</td>
<td>66</td>
</tr>
<tr>
<td>usersGetSubscriptions</td>
<td>67</td>
</tr>
<tr>
<td>usersSearch</td>
<td>68</td>
</tr>
<tr>
<td>use_db</td>
<td>70</td>
</tr>
<tr>
<td>vkApply</td>
<td>71</td>
</tr>
</tbody>
</table>
areFriends

Description
Checks the friendship status between two users

Usage
areFriends(source_id, target_id)

Arguments
source_id Source user ID
target_id Target user ID
Examples

```r
## Not run:
areFriends(me(), 123456)

## End(Not run)
```

---

**boardGetComments**

*Returns a list of comments on a topic on a community’s discussion board*

---

**Description**

Returns a list of comments on a topic on a community’s discussion board

**Usage**

```r
boardGetComments(
    group_id = "",
    topic_id = "",
    need_likes = 0,
    start_comment_id = "",
    offset = 0,
    count = 20,
    sort = "",
    extended = 0,
    v = getAPIVersion()
)
```

**Arguments**

- `group_id` ID of the community that owns the discussion board.
- `topic_id` Topic ID.
- `need_likes` 1 - to return the likes field, 0 - not to return the likes field (default).
- `start_comment_id` Positive number.
- `offset` Offset needed to return a specific subset of comments.
- `count` Number of comments to return (default 20, maximum 100).
- `sort` Sort order: asc - chronological, desc - reverse chronological.
- `extended` 1 — to return information about users who posted comments; 0 — to return no additional fields (default).
- `v` Version of API
boardGetCommentsExecute

*Returns a list of comments on a topic on a community’s discussion board*

**Description**

Returns a list of comments on a topic on a community’s discussion board.

**Usage**

```r
boardGetCommentsExecute(
    group_id = "",
    topic_id = "",
    need_likes = 0,
    start_comment_id = "",
    offset = 0,
    count = 20,
    sort = "",
    progress_bar = FALSE,
    v = getAPIVersion()
)
```

**Arguments**

- `group_id`: ID of the community that owns the discussion board.
- `topic_id`: Topic ID.
- `need_likes`: 1 - to return the likes field, 0 - not to return the likes field (default).
- `start_comment_id`: Positive number.
- `offset`: Offset needed to return a specific subset of comments.
- `count`: Number of comments to return (default 20, 0 - for all comments).
- `progress_bar`: Display progress bar.
- `v`: Version of API
**boardGetCommentsList**  
*Returns a list of comments on a community’s discussion board*

### Description

Returns a list of comments on a community’s discussion board

### Usage

```r
boardGetCommentsList(topics, progress_bar = FALSE, v = getAPIVersion())
```

### Arguments

- **topics**  
  A list of topics (from getTopicsExecute())
- **progress_bar**  
  Display progress bar
- **v**  
  Version of API

---

**clear_text**  
*Clear text*

### Description

Clear text

### Usage

```r
clear_text(lines, patterns = list())
```

### Arguments

- **lines**  
  List of lines
- **patterns**  
  List of user defined patterns
**collection_exists**  
*Check if collection exists*

**Description**
Check if collection exists

**Usage**

```python
    collection_exists(
        collection_name,
        collection_suffix = '',
        db_name = db_getActive()
    )
```

**Arguments**
- **collection_name**  
  Collection name
- **collection_suffix**  
  Collection suffix
- **db_name**  
  Database name

---

**create_empty_collection**  
*Create empty collection*

**Description**
Create empty collection

**Usage**

```python
    create_empty_collection(collection, suffix, db_name = db_getActive())
```

**Arguments**
- **collection**  
  Collection name
- **suffix**  
  Collection suffix
- **db_name**  
  Database name
databaseGetChairs

Returns list of chairs on a specified faculty

Description

Returns list of chairs on a specified faculty

Usage

databaseGetChairs(
    faculty_id = "",
    offset = "",
    count = "100",
    v = getAPIVersion()
)

Arguments

faculty_id       ID of the faculty to get chairs from
offset           Offset required to get a certain subset of chairs
count            Amount of chairs to get
v                Version of API

Examples

## Not run:
databaseGetChairs(206)

## End(Not run)

databaseGetCities

Returns a list of cities

Description

Returns a list of cities

Usage

databaseGetCities(
    country_id = "",
    region_id = "",
    q = "",
    need_all = "1",
    offset = "",
    count = "100",
    v = getAPIVersion()
)
count = "100",
    v = getAPIVersion()
)

Arguments

country_id  Country ID
region_id   Region ID
q           Search query
need_all    1 - to return all cities in the country; 0 - to return major cities in the country (default)
offset      Offset needed to return a specific subset of cities
count       Number of cities to return
v            Version of API

Examples

## Not run:
databaseGetCities(country_id=1, need_all=0)

## End(Not run)

---

databaseGetCitiesById  Returns information about cities by their IDs

Description

Returns information about cities by their IDs

Usage

databaseGetCitiesById(city_ids = "", v = getAPIVersion())

Arguments

city_ids     City IDs
v            Version of API

Examples

## Not run:
databaseGetCitiesById('1,2')

## End(Not run)
databaseGetCountries

Returns a list of countries

Description

Returns a list of countries

Usage

databaseGetCountries(
    need_all = "1",
    code = "",
    offset = "",
    count = "100",
    v = getAPIVersion()
)

Arguments

need_all 1 - to return a full list of all countries; 0 - to return a list of countries near the current user’s country

code Country codes in ISO 3166-1 alpha-2 standard

offset Offset needed to return a specific subset of countries

count Number of countries to return

v Version of API

Examples

## Not run:
databaseGetCountries(count=234)

## End(Not run)

databaseGetCountriesById

Returns information about countries by their IDs

Description

Returns information about countries by their IDs

Usage

databaseGetCountriesById(country_ids, v = getAPIVersion())
databaseGetFaculties

Arguments

country_ids   Country IDs
v             Version of API

Examples

## Not run:
databaseGetCountriesById('1,2,3,4')

## End(Not run)

databaseGetFaculties    Returns a list of faculties (i.e., university departments)

Description

Returns a list of faculties (i.e., university departments)

Usage

databaseGetFaculties(
    university_id = "",
    offset = "",
    count = "100",
    v = getAPIVersion()
)

Arguments

university_id   University ID
offset          Offset needed to return a specific subset of faculties
count           Number of faculties to return
v                Version of API

Examples

## Not run:
databaseGetFaculties(53)

## End(Not run)
**databaseGetRegions**

*Returns a list of regions*

**Description**

Returns a list of regions

**Usage**

```r
databaseGetRegions(
  country_id = "", 
  q = "", 
  offset = "", 
  count = "100", 
  v = getAPIVersion()
)
```

**Arguments**

- **country_id** Country ID, received in database.getCountries method
- **q** Search query
- **offset** Offset needed to return specific subset of regions
- **count** Number of regions to return
- **v** Version of API

**Examples**

```r
## Not run:
databaseGetRegions(229)
## End(Not run)
```

---

**databaseGetSchoolClasses**

*Returns a list of available classes*

**Description**

Returns a list of available classes

**Usage**

```r
databaseGetSchoolClasses(country_id = "", v = getAPIVersion())
```
databaseGetSchools

**Arguments**

- `country_id`  
  Country ID
- `v`  
  Version of API

**Examples**

```r
## Not run:
databaseGetSchoolClasses(1)

## End(Not run)
```

databaseGetSchools  
*Returns a list of schools*

**Description**

Returns a list of schools

**Usage**

```r
databaseGetSchools(
  q = "", 
  city_id = "", 
  offset = "", 
  count = "100", 
  v = getAPIVersion()
)
```

**Arguments**

- `q`  
  Search query
- `city_id`  
  City ID
- `offset`  
  Offset needed to return a specific subset of schools
- `count`  
  Number of schools to return
- `v`  
  Version of API

**Examples**

```r
## Not run:
databaseGetSchools(city_id = 2)

## End(Not run)
```
**databaseGetStreetsById**

*Returns information about streets by their IDs*

**Description**

Returns information about streets by their IDs

**Usage**

```r
databaseGetStreetsById(street_ids = "", v = getAPIVersion())
```

**Arguments**

- `street_ids` Street IDs
- `v` Version of API

**Examples**

```r
## Not run:
databaseGetStreetsById(1)

## End(Not run)
```

**databaseGetUniversities**

*Returns a list of higher education institutions*

**Description**

Returns a list of higher education institutions

**Usage**

```r
databaseGetUniversities(
  q = "",
  country_id = "",
  city_id = "",
  offset = "",
  count = "100",
  v = getAPIVersion()
)
```
### db_drop_collection

**Arguments**

- `q` (Search query)
- `country_id` (Country ID)
- `city_id` (City ID)
- `offset` (Offset needed to return a specific subset of universities)
- `count` (Number of universities to return)
- `v` (Version of API)

**Examples**

```r
## Not run:
databaseGetUniversities(city_id = '2')
## End(Not run)
```

---

### db_drop

**Drop database**

**Description**

Drop database

**Usage**

```r
db_drop(db_name)
```

**Arguments**

- `db_name` (Database name)

---

### db_drop_collection

**Drop collection**

**Description**

Drop collection

**Usage**

```r
db_drop_collection(
  collection_name,
  collection_suffix = "",
  db_name = db_getActive()
)
```
**db_getActive**

**Arguments**
- `collection_name`  
  Collection name
- `collection_suffix`  
  Collection suffix
- `db_name`  
  Database name

**Description**

The current database name

**Usage**

```python
db_getActive()
```

---

**db_getActive**  
*The current database name*

---

**db_getName**  
*The current database name*

**Description**

The current database name

**Usage**

```python
db_getName()
```

---

**db_get_collection**  
*Get collection*

**Description**

Get collection

**Usage**

```python
db_get_collection(
    collection_name,
    collection_suffix = "",
    db_name = db_getActive()
)
```
### Arguments

- **collection_name**: Collection name
- **collection_suffix**: Collection suffix
- **db_name**: Database name

### Description

Get a mongo connection object

### Usage

```r
db_get_connection(
  collection_name,
  collection_suffix = "",
  db_name = db_getActive()
)
```

### Arguments

- **collection_name**: Collection name
- **collection_suffix**: Collection suffix
- **db_name**: Database name

### Description

Initialize database

### Usage

```r
db_init(db_name = "temp", verbose = FALSE)
```

### Arguments

- **db_name**: Database name (‘temp’ by default)
- **verbose**: Emit some more output
db_insert

**Insert object into existing collection**

**Description**
Insert object into existing collection

**Usage**

```python
db_insert(object, collection, suffix, db_name = db_getActive())
```

**Arguments**

- **object**: Object to insert
- **collection**: Collection name
- **suffix**: Collection suffix
- **db_name**: Database name

---

db_load

**Load all collections from db for specified data base**

**Description**
Load all collections from db for specified data base

**Usage**

```python
db_load(db_name = db_getActive())
```

**Arguments**

- **db_name**: Database name
db_load_collection  

*Load collection from db*

**Description**

Load collection from db

**Usage**

```cpp
db_load_collection(
    collection_name,
    collection_suffix = "",
    db_name = db_getActive()
)
```

**Arguments**

- `collection_name`  
  Collection name
- `collection_suffix`  
  Collection suffix
- `db_name`  
  Database name

---

db_metaConnection  

*Get meta connection*

**Description**

Get meta connection

**Usage**

```cpp
db_metaConnection()
```
db_save

Save object to db

Description
Save object to db

Usage
db_save(object, collection, suffix = "", db_name = db_getActive())

Arguments
- object: Object to save
- collection: Collection name
- suffix: Collection suffix
- db_name: Database name

db_update

Update existing records

Description
Update existing records

Usage
db_update(
    object,
    key,
    collection,
    suffix = "",
    db_name = db_getActive(),
    upsert = FALSE
)

Arguments
- object: Object to insert
- key: Key
- collection: Collection name
- suffix: Collection suffix
- db_name: Database name
- upsert: Insert a new document if no matching document exists
### execute

**Description**

A universal method for calling a sequence of other methods while saving and filtering interim results

**Usage**

```python
execute(code, params = list())
```

**Arguments**

- **code**  
  Algorithm code in VKScript
- **params**  
  Parameters list

### filterAttachments

**Description**

Filtering attachments by type

**Usage**

```python
filterAttachments(attachments, type)
```

**Arguments**

- **attachments**  
  List of attachments
- **type**  
  type field may have the following values:
  - **photo** - photo from an album;
  - **posted_photo** - photo uploaded directly from user’s computer;
  - **video** - video;
  - **audio** - audio;
  - **doc** - document;
  - **graffiti** - graffiti;
  - **url** - web page URL;
  - **note** - note;
  - **app** - image uploaded with a third party application;
  - **poll** - poll;
  - **page** - wiki page.
getAccessToken

Get access token

Description
Get access token

Usage
getAccessToken()

getArbitraryNetwork
Building a friend graph for an arbitrary list of users

Description
Building a friend graph for an arbitrary list of users

Usage
getArbitraryNetwork(users_ids, format = "edgelist")

Arguments
users_ids User IDs
format Either "edgelist" for a list of edges or "adjmatrix" for an adjacency matrix

getCountryByCityId
Get country ID and title by given city ID

Description
Get country ID and title by given city ID

Usage
getCountryByCityId(city_id)

Arguments
city_id City ID
**getEgoNetwork**

*Building a friend graph*

**Description**

Building a friend graph

**Usage**

```r
getEgoNetwork(users_ids = "")
```

**Arguments**

- `users_ids`: User IDs

**getFriends**

*Returns a list of user IDs or detailed information about a user’s friends*

**Description**

Returns a list of user IDs or detailed information about a user’s friends

**Usage**

```r
getFriends(
  user_id = "",
  order = "",
  list_id = "",
  count = "",
  offset = "",
  fields = "",
  name_case = "",
  flatten = FALSE,
  v = getAPIVersion()
)
```

**Arguments**

- `user_id`: User ID. By default, the current user ID
- `order`: Sort order (name - by name, hints - by rating)
- `list_id`: ID of the friend list returned by the friends.getLists method to be used as the source. This parameter is taken into account only when the uid parameter is set to the current user ID
- `count`: Number of friends to return
getFriendsBy25

Offset needed to return a specific subset of friends
fields Profile fields to return
name_case Case for declension of user name and surname
flatten Automatically flatten nested data frames into a single non-nested data frame
v Version of API

Examples

## Not run:
friends_list <- getFriends(user_id=1, order='name', fields='bdate')
friends <- friends_list$items

## End(Not run)

getFriendsBy25 Returns a list of friends IDs for the specified users

Description

Returns a list of friends IDs for the specified users

Usage

getFriendsBy25(user_ids, v = getAPIVersion())

Arguments

user_ids User IDs (maximum 25)
v Version of API

Examples

## Not run:
my_friends <- getFriends()
friends_of_friends <- getFriendsBy25(my_friends$items[1:25])

## End(Not run)
getFriendsFor  
*Returns a list of friends IDs for the specified users*

**Description**

Returns a list of friends IDs for the specified users

**Usage**

getFriendsFor(users_ids, v = getAPIVersion())

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>users_ids</td>
<td>User IDs</td>
</tr>
<tr>
<td>v</td>
<td>Version of API</td>
</tr>
</tbody>
</table>

**Examples**

```r
## Not run:
friends <- getFriendsFor(sample(x=seq(1:1000000), size=100, replace=FALSE))
users <- getUsersExecute(friends, fields = 'sex')
## End(Not run)
```

getGroups  
*Returns a list of the communities to which a user belongs*

**Description**

Returns a list of the communities to which a user belongs

**Usage**

getGroups(
  user_id = "",
  extended = "",
  filter = "",
  fields = "",
  offset = "",
  count = "",
  v = getAPIVersion()
)
getGroupsById

Arguments

- **user_id**: User ID
- **extended**: 1 - to return complete information about a user’s communities; 0 - to return a list of community IDs without any additional fields (default)
- **filter**: Types of communities to return: admin, editor, moder, groups, publics, events
- **fields**: List of additional fields to be returned
- **offset**: Offset needed to return a specific subset of communities
- **count**: Number of communities to return (maximum value 1000)
- **v**: Version of API

Examples

```r
## Not run:
groups <- getGroups(me(), extended = 1, fields = 'city')
## End(Not run)
```

getGroupsById

*Returns information about communities by their IDs*

Description

Returns information about communities by their IDs

Usage

```r
getGroupsById(group_ids = '', group_id = '', fields = '', v = getAPIVersion())
```

Arguments

- **group_ids**: IDs or screen names of communities
- **group_id**: ID or screen name of the community
- **fields**: Group fields to return
- **v**: Version of API
getGroupsForUsers  Returns a list of the communities for the specified users  

Description

Returns a list of the communities for the specified users

Usage

getGroupsForUsers(
  users,
  extended = "",
  filter = "",
  fields = "",
  progress_bar = FALSE,
  v = getAPIVersion()
)

Arguments

users A list of users
extended 1 - to return complete information about a user’s communities; 0 - to return a list of community IDs without any additional fields (default)
filter Types of communities to return: admin, editor, moder, groups, publics, events
fields List of additional fields to be returned
progress_bar Display progress bar
v Version of API

Examples

## Not run:
members <- getGroupsForUsers(c(me(), 123456), extended = 1, fields='city', progress_bar = TRUE)

## End(Not run)

groupsMembers  Returns a list of community members

Description

Returns a list of community members
Usage

getGroupsMembersExecute(
    group_id = "", 
    sort = "", 
    offset = "", 
    count = "", 
    fields = "", 
    filter = "", 
    v = getAPIVersion()
)

Arguments

- **group_id**: ID or screen name of the community
- **sort**: Sort order
- **offset**: Offset needed to return a specific subset of community members
- **count**: Number of community members to return (maximum value 1000)
- **fields**: List of additional fields to be returned
- **filter**: friends - only friends in this community will be returned; unsure - only those who pressed 'I may attend' will be returned (if it's an event)
- **v**: Version of API

Examples

```r
## Not run:
members <- getGroupsMembers(1, fields='sex,bdate,city')
## End(Not run)
```

getGroupsMembersExecute

*Returns a list of community members*

Description

Returns a list of community members

Usage

getGroupsMembersExecute(
    group_id = "", 
    sort = "", 
    offset = 0, 
    count = 0, 
    fields = "", 
    filter = "", 
    v = getAPIVersion()
)
getMutual

Returns a list of user IDs of the mutual friends of two users

Description

Returns a list of user IDs of the mutual friends of two users

Usage

getMutual(
  source_id = "",
  target_uid = "",
  target_uids = "",
  order = "",
  count = "",
  offset = "",
  v = getAPIVersion()
)
getMutualExecute

Arguments

source_id: ID of the user whose friends will be checked against the friends of the user specified in target_uid

target_uid: ID of the user whose friends will be checked against the friends of the user specified in source_uid

target_uids: List of target uids (list of comma-separated positive numbers, the maximum number of elements allowed is 100)

order: Sort order

count: Number of mutual friends to return

offset: Offset needed to return a specific subset of mutual friends

v: Version of API

Examples

## Not run:
mutual_friends <- getMutual(target_uid=1)

## End(Not run)

---

getMutualExecute: Returns a list of user IDs of the mutual friends of two users

Description

Returns a list of user IDs of the mutual friends of two users

Usage

getMutualExecute(
  source_id = "",  
  target_uid = "",  
  target_uids = "",  
  order = "",  
  count = "",  
  offset = "",  
  progress_bar = FALSE,  
  v = getAPIVersion()
)

Arguments

source_id: ID of the user whose friends will be checked against the friends of the user specified in target_uid

target_uid: ID of the user whose friends will be checked against the friends of the user specified in source_uid
getStatus

**target_uids**  List of target uids
**order**  Sort order
**count**  Number of mutual friends to return
**offset**  Offset needed to return a specific subset of mutual friends
**progress_bar**  Display progress bar
**v**  Version of API

**Examples**

```
## Not run:
mutual_friends <- getMutualExecute(target_uid=1)

## End(Not run)
```

---

getPaths  *Returns a list of paths between two users*

**Description**

Returns a list of paths between two users

**Usage**

```
getPaths(source_id, target_id, are_friends = FALSE, max_depth = 5)
```

**Arguments**

- **source_id**: Source ID
- **target_id**: Target ID
- **are_friends**: By default is FALSE
- **max_depth**: Maximum depth

---

getStatus  *Returns data required to show the status of a users and/or communities*

**Description**

Returns data required to show the status of a users and/or communities
getTopics

Returns a list of topics on a community’s discussion board

Usage

getTopics(
  group_id = "",  
  topics_ids = "",  
  order = "",  
  offset = 0,  
  count = 40,  
  extended = 0,  
  preview = 0,  
  preview_length = 90,  
  v = getAPIVersion()
)

Description

Returns a list of topics on a community’s discussion board

Usage

getStatus(
  users_ids = c(),  
  groups_ids = c(),  
  progress_bar = FALSE,  
  v = getAPIVersion()
)

Arguments

users_ids User IDs  
groups_ids Community IDs  
progress_bar Display progress bar  
v Version of API

Examples

## Not run:
status.me <- getStatus()  
status.friends <- getStatus(users_ids = getFriends()$items)  
status.groups <- getStatus(groups_ids = getGroups()$items)  
status.friends_and_groups <- getStatus(users_ids = getFriends()$items,  
groups_ids = getGroups()$items, progress_bar = T)
## End(Not run)
Arguments

- **group_id**: ID of the community that owns the discussion board.
- **topics_ids**: IDs of topics to be returned (100 maximum). By default, all topics are returned. If this parameter is set, the order, offset, and count parameters are ignored.
- **order**: Sort order:
  - 1 - by date updated in reverse chronological order;
  - 2 - by date created in reverse chronological order;
  - -1 - by date updated in chronological order;
  - -2 - by date created in chronological order.
  If no sort order is specified, topics are returned in the order specified by the group administrator. Pinned topics are returned first, regardless of the sorting.
- **offset**: Offset needed to return a specific subset of topics.
- **count**: Number of topics to return (default 40, maximum value 100).
- **extended**: 1 — to return information about users who created topics or who posted there last; 0 — to return no additional fields (default).
- **preview**: 1 — to return the first comment in each topic; 2 — to return the last comment in each topic; 0 — to return no comments.
- **preview_length**: Number of characters after which to truncate the previewed comment. To preview the full comment, specify 0.
- **v**: Version of API

---

**getTopicsExecute**

*Returns a list of topics on a community’s discussion board*

---

Description

Returns a list of topics on a community’s discussion board

Usage

```r
getTopicsExecute(
  group_id = "",
  order = "",
  offset = 0,
  count = 40,
  preview = 0,
  preview_length = 90,
  use_db = FALSE,
  db_params = list(),
  progress_bar = FALSE,
  v = getAPIVersion()
)
```
Arguments

- **group_id**: ID of the community that owns the discussion board.
- **order**: Sort order:
  - 1 - by date updated in reverse chronological order;
  - 2 - by date created in reverse chronological order;
  - -1 - by date updated in chronological order;
  - -2 - by date created in chronological order.
  If no sort order is specified, topics are returned in the order specified by the group administrator. Pinned topics are returned first, regardless of the sorting.
- **offset**: Offset needed to return a specific subset of topics.
- **count**: Number of topics to return (default 40, 0 - for all topics).
- **preview**: 1 — to return the first comment in each topic; 2 — to return the last comment in each topic; 0 — to return no comments.
- **preview_length**: Number of characters after which to truncate the previewed comment. To preview the full comment, specify 0.
- **use_db**: Use database
- **db_params**: Collection name and suffix
- **progress_bar**: Display progress bar
- **v**: Version of API

---

**getURLs**  
*Extract URLs from messages*

Description

Extract URLs from messages

Usage

```r
getURLs(messages, message_body = FALSE)
```

Arguments

- **messages**: Array of messages
- **message_body**: Add message body to URLs
getUsers returns detailed information on users

**Description**

Returns detailed information on users

**Usage**

```r
getUsers(
  user_ids = "",
  fields = "",
  name_case = "nom",
  flatten = FALSE,
  v = getAPIVersion()
)
```

**Arguments**

- **user_ids** User IDs or screen names (screen_name). By default, current user ID (the maximum number of elements allowed is 1000)
- **fields** Profile fields to return (see details for more information about fields)
- **name_case** Case for declension of user name and surname
- **flatten** Automatically flatten nested data frames into a single non-nested data frame
- **v** Version of API

**Details**

User object describes a user profile, contains the following fields:

- **uid** User ID
- **first_name** First name
- **last_name** Last name
- **deactivated** Returns if a profile is deleted or blocked. Gets the value deleted or banned. Keep in mind that in this case no additional fields are returned
- **hidden**: 1 Returns while operating without access_token if a user has set the "Who can see my profile on the Internet" -> "Only VK users" privacy setting. Keep in mind that in this case no additional fields are returned
- **verified** Returns 1 if the profile is verified, 0 if not
- **blacklisted** Returns 1 if a current user is in the requested user's blacklist
- **sex** User sex (1 - female; 2 - male; 0 - not specified)
- **bdate** User's date of birth. Returned as DD.MM.YYYY or DD.MM (if birth year is hidden). If the whole date is hidden, no field is returned
• **city** ID of the city specified on user’s page in "Contacts" section. Returns city ID that can be used to get its name using places.getCityById method. If no city is specified or main information on the page is hidden for in privacy settings, then it returns 0

• **country** ID of the country specified on user’s page in "Contacts" section. Returns country ID that can be used to get its name using places.getCountryById method. If no country is specified or main information on the page is hidden in privacy settings, then it returns 0

• **home_town** User’s home town

• **photo_50** Returns URL of square photo of the user with 50 pixels in width. In case user does not have a photo, http://vk.com/images/camera_c.gif is returned

• **photo_100** Returns URL of square photo of the user with 100 pixels in width. In case user does not have a photo, http://vk.com/images/camera_b.gif is returned

• **photo_200_orig** Returns URL of user’s photo with 200 pixels in width. In case user does not have a photo, http://vk.com/images/camera_a.gif is returned

• **photo_200** Returns URL of square photo of the user with 200 pixels in width. If the photo was uploaded long time ago, there can be no image of such size and in this case the reply will not include this field

• **photo_400_orig** Returns URL of user’s photo with 400 pixels in width. If user does not have a photo of such size, reply will not include this field

• **photo_max** Returns URL of square photo of the user with maximum width. Can be returned a photo both 200 and 100 pixels in width. In case user does not have a photo, http://vk.com/images/camera_b.gif is returned

• **photo_max_orig** Returns URL of user’s photo of maximum size. Can be returned a photo both 400 and 200 pixels in width. In case user does not have a photo, http://vk.com/images/camera_a.gif is returned

• **online** Information whether the user is online. Returned values: 1 - online, 0 - offline. If user utilizes a mobile application or site mobile version, it returns online_mobile additional field that includes 1. With that, in case of application, online_app additional field is returned with application ID.

• **lists** Information about friend lists. Returns IDs of friend lists the user is member of, separated with a comma. The field is available for friends.get method only. To get information about ID and names of friend lists use friends.getLists method. If user is not a member of any friend list, then when accepting data in XML format the respective <user> node does not contain <lists> tag

• **domain** Page screen name. Returns a string with a page screen name (only subdomain is returned, like andrew). If not set, "id'+uid is returned, e.g. id35828305

• **has_mobile** Information whether the user’s mobile phone number is available. Returned values: 1 - available, 0 - not available. We recommend you to use it prior to call of secure.sendSMSNotification method

• **contacts** Information about user’s phone numbers. If data are available and not hidden in privacy settings, the following fields are returned (mobile_phone - user’s mobile phone number (only for standalone applications); home_phone - user’s additional phone number)

• **site** Returns a website address from a user profile

• **education** Information about user’s higher education institution. The following fields are returned:
- **university** University ID
- **university_name** University name
- **faculty** Faculty ID
- **faculty_name** Faculty name
- **graduation** Graduation year

- **universities** List of higher education institutions where user studied. Returns universities array with university objects with the following fields:
  - **id** University ID
  - **country** ID of the country the university is located in
  - **city** ID of the city the university is located in
  - **name** University name
  - **faculty** Faculty ID
  - **faculty_name** Faculty name
  - **chair** University chair ID
  - **chair_name** Chair name
  - **graduation** Graduation year

- **schools** List of schools where user studied in. Returns schools array with school objects with the following fields:
  - **id** School ID
  - **country** ID of the country the school is located in
  - **city** ID of the city the school is located in
  - **name** School name
  - **year_from** Year the user started to study
  - **year_to** Year the user finished to study
  - **year_graduated** Graduation year
  - **class** School class letter
  - **speciality** Speciality
  - **type** Type ID
  - **type_str** Type name

- **status** User status. Returns a string with status text that is in the profile below user’s name

- **last_seen** Last visit date. Returns last_seen object with the following fields:
  - **time** Last visit date (in Unix time)
  - **platform** Type of the platform that used for the last authorization. See more at Using LongPoll server

- **followers_count** Number of user’s followers

- **common_count** Number of common friends with a current user

- **counters** Number of various objects the user has. Can be used in users.get method only when requesting information about a user. Returns an object with fields:
  - **albums** Number of photo albums
  - **videos** Number of videos
  - **audios** Number of audios
- **notes** Number of notes
- **friends** Number of friends
- **groups** Number of communities
- **online_friends** Number of online friends
- **mutual_friends** Number of mutual friends
- **user_videos** Number of videos the user is tagged on
- **followers** Number of followers
- **user_photos** Number of photos the user is tagged on
- **subscriptions** Number of subscriptions

**occupation** Current user’s occupation. Returns following fields:
- **type** Can take the values: work, school, university
- **id** ID of school, university, company group (the one a user works in)
- **name** Name of school, university or work place

**nickname** User nickname

**relatives** Current user’s relatives list. Returns a list of objects with id and type fields (name instead of id if a relative is not a VK user). type - relationship type. Possible values:
- **sibling**
- **parent**
- **child**
- **grandparent**
- **grandchild**

**relation** User relationship status. Returned values:
- **1** - Single
- **2** - In a relationship
- **3** - Engaged
- **4** - Married
- **5** - It’s complicated
- **6** - Actively searching
- **7** - In love

**personal** Information from the "Personal views" section
- **political** Political views:
  - **1** - Communist
  - **2** - Socialist
  - **3** - Moderate
  - **4** - Liberal
  - **5** - Conservative
  - **6** - Monarchist
  - **7** - Ultraconservative
  - **8** - Apathetic
  - **9** - Libertarian
- **langs** Languages
- **religion** World view
- **inspired_by** Inspired by
- **people_main** Important in others:
  * 1 - Intellect and creativity
  * 2 - Kindness and honesty
  * 3 - Health and beauty
  * 4 - Wealth and power
  * 5 - Courage and persistence
  * 6 - Humor and love for life
- **life_main** Personal priority:
  * 1 - Family and children
  * 2 - Career and money
  * 3 - Entertainment and leisure
  * 4 - Science and research
  * 5 - Improving the world
  * 6 - Personal development
  * 7 - Beauty and art
  * 8 - Fame and influence
- **smoking** Views on smoking (1 - very negative; 2 - negative; 3 - neutral; 4 - compromisable; 5 - positive)
- **alcohol** Views on alcohol (1 - very negative; 2 - negative; 3 - neutral; 4 - compromisable; 5 - positive)

- **connections** Returns specified services such as: skype, facebook, twitter, livejournal, instagram
- **exports** External services with export configured (twitter, facebook, livejournal, instagram)
- **wall_comments** Wall comments allowed (1 - allowed, 0 - not allowed)
- **activities** Activities
- **interests** Interests
- **music** Favorite music
- **movies** Favorite movies
- **tv** Favorite TV shows
- **books** Favorite books
- **games** Favorite games
- **about** "About me"
- **quotes** Favorite quotes
- **can_post** Can post on the wall: 1 - allowed, 0 - not allowed
- **can_see_all_posts** Can see other users’ posts on the wall: 1 - allowed, 0 - not allowed
- **can_see_audio** Can see other users’ audio on the wall: 1 - allowed, 0 - not allowed
- **can_write_private_message** Can write private messages to a current user: 1 - allowed, 0 - not allowed
- **timezone** user time zone. Returns only while requesting current user info
- **screen_name** User page’s screen name (subdomain)
getUsersExecute

Examples

```r
## Not run:
user <- getUsers('1', fields='sex,bdate,city')
## End(Not run)
```

getUsersExecute (Returns detailed information on arbitrary number of users)

Description

Returns detailed information on arbitrary number of users

Usage

```r
getUsersExecute(
  users_ids,
  fields = "",
  name_case = "nom",
  drop = FALSE,
  flatten = FALSE,
  use_db = FALSE,
  db_params = list(),
  progress_bar = FALSE,
  v = getAPIVersion()
)
```

Arguments

- `users_ids`: User IDs or screen names (screen_name). By default, current user ID
- `fields`: Profile fields to return
- `name_case`: Case for declension of user name and surname
- `drop`: Drop deleted or banned users
- `flatten`: Automatically flatten nested data frames into a single non-nested data frame
- `use_db`: Use database
- `db_params`: Collection name and suffix
- `progress_bar`: Display progress bar
- `v`: Version of API
Details

User object describes a user profile, contains the following fields:

- **uid** User ID
- **first_name** First name
- **last_name** Last name
- **deactivated** Returns if a profile is deleted or blocked. Gets the value deleted or banned. Keep in mind that in this case no additional fields are returned
- **hidden** Returns 1 if a user has set the "Who can see my profile on the Internet" -> "Only VK users" privacy setting. Keep in mind that in this case no additional fields are returned
- **verified** Returns 1 if the profile is verified, 0 if not
- **blacklisted** Returns 1 if a current user is in the requested user’s blacklist
- **sex** User sex (1 - female; 2 - male; 0 - not specified)
- **bdate** User’s date of birth. Returned as DD.MM.YYYY or DD.MM (if birth year is hidden). If the whole date is hidden, no field is returned
- **city** ID of the city specified on user’s page in "Contacts" section. Returns city ID that can be used to get its name using places.getCityById method. If no city is specified or main information on the page is hidden in privacy settings, then it returns 0
- **country** ID of the country specified on user’s page in "Contacts" section. Returns country ID that can be used to get its name using places.getCountryById method. If no country is specified or main information on the page is hidden in privacy settings, then it returns 0
- **home_town** User’s home town
- **photo_50** Returns URL of square photo of the user with 50 pixels in width. In case user does not have a photo, http://vk.com/images/camera_c.gif is returned
- **photo_100** Returns URL of square photo of the user with 100 pixels in width. In case user does not have a photo, http://vk.com/images/camera_b.gif is returned
- **photo_200_orig** Returns URL of user’s photo with 200 pixels in width. In case user does not have a photo, http://vk.com/images/camera_a.gif is returned
- **photo_200** Returns URL of square photo of the user with 200 pixels in width. If the photo was uploaded long time ago, there can be no image of such size and in this case the reply will not include this field
- **photo_400_orig** Returns URL of user’s photo with 400 pixels in width. If user does not have a photo of such size, reply will not include this field
- **photo_max** Returns URL of square photo of the user with maximum width. Can be returned a photo both 200 and 100 pixels in width. In case user does not have a photo, http://vk.com/images/camera_b.gif is returned
- **photo_max_orig** Returns URL of user’s photo of maximum size. Can be returned a photo both 400 and 200 pixels in width. In case user does not have a photo, http://vk.com/images/camera_a.gif is returned
- **online** Information whether the user is online. Returned values: 1 - online, 0 - offline. If user utilizes a mobile application or site mobile version, it returns online_mobile additional field that includes 1. With that, in case of application, online_app additional field is returned with application ID.
- **lists** Information about friend lists. Returns IDs of friend lists the user is member of, separated with a comma. The field is available for friends.get method only. To get information about ID and names of friend lists use friends.getLists method. If user is not a member of any friend list, then when accepting data in XML format the respective `<lists>` tag does not contain `<lists>` tag.

- **domain** Page screen name. Returns a string with a page screen name (only subdomain is returned, like andrew). If not set, "id'+uid is returned, e.g. id35828305

- **has_mobile** Information whether the user’s mobile phone number is available. Returned values: 1 - available, 0 - not available. We recommend you to use it prior to call of secure.sendSMSNotification method.

- **contacts** Information about user’s phone numbers. If data are available and not hidden in privacy settings, the following fields are returned (mobile_phone - user’s mobile phone number (only for standalone applications); home_phone - user’s additional phone number)

- **site** Returns a website address from a user profile.

- **education** Information about user’s higher education institution. The following fields are returned:
  - **university** University ID
  - **university_name** University name
  - **faculty** Faculty ID
  - **faculty_name** Faculty name
  - **graduation** Graduation year

- **universities** List of higher education institutions where user studied. Returns universities array with university objects with the following fields:
  - **id** University ID
  - **country** ID of the country the university is located in
  - **city** ID of the city the university is located in
  - **name** University name
  - **faculty** Faculty ID
  - **faculty_name** Faculty name
  - **chair** University chair ID
  - **chair_name** Chair name
  - **graduation** Graduation year

- **schools** List of schools where user studied in. Returns schools array with school objects with the following fields:
  - **id** School ID
  - **country** ID of the country the school is located in
  - **city** ID of the city the school is located in
  - **name** School name
  - **year_from** Year the user started to study
  - **year_to** Year the user finished to study
  - **year_graduated** Graduation year
  - **class** School class letter
- **speciality** Speciality
- **type** Type ID
- **type_str** Type name

- **status** User status. Returns a string with status text that is in the profile below user’s name

- **last_seen** Last visit date. Returns last_seen object with the following fields:
  - **time** Last visit date (in Unix time)
  - **platform** Type of the platform that used for the last authorization. See more at Using LongPoll server

- **followers_count** Number of user’s followers

- **common_count** Number of common friends with a current user

- **counters** Number of various objects the user has. Can be used in users.get method only when requesting information about a user. Returns an object with fields:
  - **albums** Number of photo albums
  - **videos** Number of videos
  - **audios** Number of audios
  - **notes** Number of notes
  - **friends** Number of friends
  - **groups** Number of communities
  - **online_friends** Number of online friends
  - **mutual_friends** Number of mutual friends
  - **user_videos** Number of videos the user is tagged on
  - **followers** Number of followers
  - **user_photos** Number of photos the user is tagged on
  - **subscriptions** Number of subscriptions

- **occupation** Current user’s occupation. Returns following fields:
  - **type** Can take the values: work, school, university
  - **id** ID of school, university, company group (the one a user works in)
  - **name** Name of school, university or work place

- **nickname** User nickname

- **relatives** Current user’s relatives list. Returns a list of objects with id and type fields (name instead of id if a relative is not a VK user). type - relationship type. Possible values:
  - **sibling**
  - **parent**
  - **child**
  - **grandparent**
  - **grandchild**

- **relation** User relationship status. Returned values:
  - **1** - Single
  - **2** - In a relationship
  - **3** - Engaged
  - **4** - Married
• **personal** Information from the "Personal views" section
  
  – **political** Political views:
    * 1 - Communist
    * 2 - Socialist
    * 3 - Moderate
    * 4 - Liberal
    * 5 - Conservative
    * 6 - Monarchist
    * 7 - Ultraconservative
    * 8 - Apathetic
    * 9 - Libertarian
  
  – **langs** Languages
  
  – **religion** World view
  
  – **inspired_by** Inspired by
  
  – **people_main** Important in others:
    * 1 - Intellect and creativity
    * 2 - Kindness and honesty
    * 3 - Health and beauty
    * 4 - Wealth and power
    * 5 - Courage and persistence
    * 6 - Humor and love for life
  
  – **life_main** Personal priority:
    * 1 - Family and children
    * 2 - Career and money
    * 3 - Entertainment and leisure
    * 4 - Science and research
    * 5 - Improving the world
    * 6 - Personal development
    * 7 - Beauty and art
    * 8 - Fame and influence
  
  – **smoking** Views on smoking (1 - very negative; 2 - negative; 3 - neutral; 4 - compromisable; 5 - positive)
  
  – **alcohol** Views on alcohol (1 - very negative; 2 - negative; 3 - neutral; 4 - compromisable; 5 - positive)

• **connections** Returns specified services such as: skype, facebook, twitter, livejournal, instagram

• **exports** External services with export configured (twitter, facebook, livejournal, instagram)

• **wall_comments** Wall comments allowed(1 - allowed, 0 - not allowed)

• **activities** Activities
• **interests** Interests
• **music** Favorite music
• **movies** Favorite movies
• **tv** Favorite TV shows
• **books** Favorite books
• **games** Favorite games
• **about** "About me"
• **quotes** Favorite quotes
• **can_post** Can post on the wall: 1 - allowed, 0 - not allowed
• **can_see_all_posts** Can see other users' posts on the wall: 1 - allowed, 0 - not allowed
• **can_see_audio** Can see other users' audio on the wall: 1 - allowed, 0 - not allowed
• **can_write_private_message** Can write private messages to a current user: 1 - allowed, 0 - not allowed
• **timezone** user time zone. Returns only while requesting current user info
• **screen_name** User page’s screen name (subdomain)

### Examples
```r
## Not run:
random_ids <- sample(x=seq(1:10000000), size=10000, replace=FALSE)
users <- getUsersExecute(random_ids, fields='sex,bdate,city')
## End(Not run)
```

---

**getWall**  
*Returns a list of posts on a user wall or community wall*

**Description**

Returns a list of posts on a user wall or community wall

**Usage**

```r
getWall(
  owner_id = "",
  domain = "",
  offset = "",
  count = "",
  filter = "owner",
  extended = "",
  fields = "",
  v = getAPIVersion()
)
```
getWallExecute

Arguments

owner_id   ID of the user or community that owns the wall. By default, current user ID. Use a negative value to designate a community ID.
domain   User or community short address.
offset   Offset needed to return a specific subset of posts.
count   Number of posts to return (maximum 100).
filter   Filter to apply:
  • owner - posts by the wall owner;
  • others - posts by someone else;
  • all - posts by the wall owner and others (default);
  • postponed - timed posts (only available for calls with an access_token);
  • suggests - suggested posts on a community wall.
extended   1 - to return wall, profiles, and groups fields, 0 - to return no additional fields (default).
fields   List of comma-separated words
v   Version of API

Value

Returns a list of post objects. If extended is set to 1, also returns the following:
  • wall - Contains a list of post objects.
  • profiles - Contains user objects with additional fields photo and online.
  • groups - Contains community objects.

Examples

```r
## Not run:
wall <- getWall(domain='spbrug', count=10, progress_bar=TRUE)
## End(Not run)
```

getWallExecute   Returns a list of posts on a user wall or community wall

Description

Returns a list of posts on a user wall or community wall
getWallExecute

Usage

```r
getWallExecute(
  owner_id = "",
  domain = "",
  offset = 0,
  count = 10,
  filter = "owner",
  extended = "",
  fields = "",
  use_db = FALSE,
  db_params = list(),
  progress_bar = FALSE,
  v = getAPIVersion()
)
```

Arguments

- **owner_id**: ID of the user or community that owns the wall. By default, current user ID. Use a negative value to designate a community ID.
- **domain**: User or community short address.
- **offset**: Offset needed to return a specific subset of posts.
- **count**: Number of posts to return (0 for all posts).
- **filter**: Filter to apply:
  - **owner**: posts by the wall owner;
  - **others**: posts by someone else;
  - **all**: posts by the wall owner and others (default);
  - **postponed**: timed posts (only available for calls with an access_token);
  - **suggests**: suggested posts on a community wall.
- **extended**: 1 - to return wall, profiles, and groups fields, 0 - to return no additional fields (default).
- **fields**: List of comma-separated words
- **use_db**: Use database
- **db_params**: Collection name and suffix
- **progress_bar**: Display progress bar
- **v**: Version of API

Value

Returns a list of post objects. If extended is set to 1, also returns the following:

- **wall**: Contains a list of post objects.
- **profiles**: Contains user objects with additional fields photo and online.
- **groups**: Contains community objects.
get_stop_words

Examples

```r
## Not run:
# get all posts from wall
wall <- getWallExecute(domain='spbrug', count=0, progress_bar=TRUE)

## End(Not run)
```

```r
get_stop_words
Get stop words list for russian language
```

Description

Get stop words list for russian language

Usage

```r
get_stop_words(stop_words = c())
```

Arguments

- `stop_words`: User defined stop words

```r
groupsSearch
Returns a list of communities matching the search criteria
```

Description

Returns a list of communities matching the search criteria

Usage

```r
groupsSearch(
    q = "",
    type = "",
    country_id = "",
    city_id = "",
    future = 0,
    market = 0,
    sort = 0,
    offset = 0,
    count = 20,
    v = getAPIVersion()
)
```
**Arguments**

- **q**: Search query string
- **type**: Community type. Possible values: group, page, event
- **country_id**: Country ID
- **city_id**: City ID. If this parameter is transmitted, country_id is ignored
- **future**: 1 — to return only upcoming events. Works with the type = event only
- **market**: 1 — to return communities with enabled market only
- **sort**: Sort order. Possible values:
  - 0 — default sorting (similar the full version of the site);
  - 1 — by growth speed;
  - 2 — by the "day attendance/members number" ratio;
  - 3 — by the "Likes number/members number" ratio;
  - 4 — by the "comments number/members number" ratio;
  - 5 — by the "boards entries number/members number" ratio.
- **offset**: Offset needed to return a specific subset of results
- **count**: Number of communities to return (default 20, maximum value 1000)
- **v**: Version of API

---

**handle_captcha**  
*Captcha error handler*

**Description**

Captcha error handler

**Usage**

```python
handle_captcha(error)
```

**Arguments**

- **error**: Error object
**handle_validation**  
*Validation error handler*

**Description**
Validation error handler

**Usage**
`handle_validation(error)`

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>Error object</td>
</tr>
</tbody>
</table>

---

**has_error**  
*Get error code from response*

**Description**
Get error code from response

**Usage**
`has_error(response)`

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>httr response object</td>
</tr>
</tbody>
</table>

---

**likesGetList**  
*Returns a list of IDs of users who added the specified object to their Likes list*

**Description**
Returns a list of IDs of users who added the specified object to their Likes list
Usage

```python
likesGetList(
    type = "",
    owner_id = "",
    item_id = "",
    page_url = "",
    filter = "",
    friends_only = "0",
    extended = "",
    offset = "",
    count = "100",
    skip_own = 0,
    v = getAPIVersion()
)
```

Arguments

type Object type
owner_id ID of the user, community, or application that owns the object
item_id Object ID
page_url URL of the page where the Like widget is installed. Used instead of the item_id parameter
filter Filters to apply: likes - returns information about all users who liked the object (default); copies - returns information only about users who told their friends about the object
friends_only Specifies which users are returned: 1 - to return only the current user's friends; 0 - to return all users (default)
extended Specifies whether extended information will be returned. 1 - to return extended information about users and communities from the Likes list; 0 - to return no additional information (default)
offset Offset needed to select a specific subset of users
count Number of user IDs to return (maximum 1000)
skip_own Flag, either 1 or 0
v Version of API

---

**likesGetListForObjects**

*Returns a list of IDs of users who added the specified objects to their Likes list*

---

**Description**

Returns a list of IDs of users who added the specified objects to their Likes list
Usage

likesGetListForObjects(
  objects,
  type = "post",
  filter = "likes",
  friends_only = 0,
  extended = 0,
  skip_own = 0,
  progress_bar = FALSE,
  v = getAPIVersion()
)

Arguments

objects List of objects (objects must contain fields owner_id and id)
type Object type (post or comment)
filter Filters to apply: likes - returns information about all users who liked the object (default); copies - returns information only about users who told their friends about the object
friends_only Specifies which users are returned: 1 - to return only the current user's friends; 0 - to return all users (default)
extended Specifies whether extended information will be returned. 1 - to return extended information about users and communities from the Likes list; 0 - to return no additional information (default)
skip_own flag, either 1 or 0
progress_bar Display progress bar
v Version of API

Examples

```r
## Not run:
wall <- getWallExecute(domain = 'privivkanet', count = 10, progress_bar = TRUE)
post_likers <- likesGetListForObjects(wall, type = 'post', progress_bar = TRUE)
post_likers_extended <- likesGetListForObjects(wall, type = 'post', extended = 1, progress_bar = TRUE)

## End(Not run)
```

me

Returns current user ID

Description

Returns current user ID
messagesGet

Returns a list of the current user’s incoming or outgoing private messages

Usage

messagesGet(
    out = "", 
    offset = "", 
    count = "", 
    time_offset = "", 
    filters = "", 
    preview_length = "", 
    last_message_id = "", 
    v = getAPIVersion()
)

Arguments

out  1 - to return outgoing messages; 0 - to return incoming messages (default)
offset  Offset needed to return a specific subset of messages
count  Number of messages to return
time_offset  Maximum time since a message was sent, in seconds. To return messages without a time limitation, set as 0
filters  Filter to apply: 1 - unread only; 2 - not from the chat; 4 - messages from friends
preview_length  Number of characters after which to truncate a previewed message. To preview the full message, specify 0
last_message_id  ID of the message received before the message that will be returned last
v  Version of API
messagesGetHistory

Returns message history for the specified user or group chat

Description

Returns message history for the specified user or group chat

Usage

messagesGetHistory(
    offset = "",
    count = "",
    user_id = "",
    peer_id = "",
    start_message_id = "",
    rev = "",
    v = getAPIVersion()
)

Arguments

offset Offset needed to return a specific subset of messages
count Number of messages to return (maximum value 200)
user_id ID of the user whose message history you want to return
peer_id Destination ID (user ID, group ID or chat ID)
start_message_id Starting message ID from which to return history
rev Sort order: 1 - return messages in chronological order; 0 - return messages in reverse chronological order
v Version of API

messagesGetHistoryAll

Returns all message history for the specified user or group chat

Description

Returns all message history for the specified user or group chat

Usage

messagesGetHistoryAll(user_id = "", peer_id = "", rev = 0, v = getAPIVersion())
messagesGetHistoryExecute

*Returns message history for the specified user or group chat*

**Arguments**

- **user_id**: ID of the user whose message history you want to return
- **peer_id**: Destination ID (user ID, group ID or chat ID)
- **rev**: Sort order: 1 - return messages in chronological order; 0 - return messages in reverse chronological order
- **v**: Version of API

```python
messagesGetHistoryExecute(
    offset = 0,
    count = 0,
    user_id = "",
    peer_id = "",
    start_message_id = "",
    rev = 0,
    progress_bar = FALSE,
    v = getAPIVersion()
)
```

**Description**

Returns message history for the specified user or group chat

**Usage**

```python
messagesGetHistoryExecute(
    offset = 0,
    count = 0,
    user_id = "",
    peer_id = "",
    start_message_id = "",
    rev = 0,
    progress_bar = FALSE,
    v = getAPIVersion()
)
```
**messagesSend**  
Sends a message.

**Description**
Sends a message.

**Usage**
```python
messagesSend(
    user_id,
    random_id = "",
    peer_id = "",
    domain = "",
    chat_id = "",
    user_ids = "",
    message = "",
    lat = "",
    long = "",
    attachment = "",
    forward_messages = "",
    sticker_id = "",
    v = getAPIVersion()
)
```

**Arguments**
- **user_id**: User ID (by default — current user).
- **random_id**: Unique identifier to avoid resending the message.
- **peer_id**: Destination ID.
- **domain**: User's short address (for example, illarionov).
- **chat_id**: ID of conversation the message will relate to.
- **user_ids**: IDs of message recipients (if new conversation shall be started).
- **message**: Text of the message (required if attachments is not set).
- **lat**: Geographical latitude of a check-in, in degrees (from -90 to 90).
- **long**: Geographical longitude of a check-in, in degrees (from -180 to 180).
- **attachment**: List of objects attached to the message, separated by commas, in the following format: `<type><owner_id>_<media_id>`.
  - **type** — Type of media attachment:
    - photo - photo
    - video - video
    - audio - audio
    - doc - document
newsfeedSearch

- wall - wall post
  <owner_id> — ID of the media attachment owner.
  <media_id> — media attachment ID.

forward_messages
  ID of forwarded messages, separated with a comma. Listed messages of the
  sender will be shown in the message body at the recipient’s.

sticker_id
  Sticker id.

v
  Version of API.

messagesSplitByDate  Split messages by days, weeks, months

Description

Split messages by days, weeks, months

Usage

messagesSplitByDate(messages, format = "%y-%m-%d")

Arguments

messages  List of messages from messagesGet()
format  Character string giving a date-time format as used by strptime

newsfeedSearch  Returns search results by statuses

Description

Returns search results by statuses

Usage

newsfeedSearch(
  q = "",
  extended = "",
  count = "",
  latitude = "",
  longitude = "",
  start_time = "",
  end_time = "",
  start_from = "",
  fields = "",
  v = getAPIVersion()
)

Arguments

 q  Search query string (e.g., New Year).
 extended  1 — to return additional information about the user or community that placed the post
 count  Number of posts to return
 latitude  Geographical latitude point (in degrees, -90 to 90) within which to search
 longitude  Geographical longitude point (in degrees, -180 to 180) within which to search
 start_time  Earliest timestamp (in Unix time) of a news item to return. By default, 24 hours ago
 end_time  Latest timestamp (in Unix time) of a news item to return. By default, the current time
 start_from  String, accessible for versions from 5.13
 fields  Additional fields of profiles and communities to return
 v  Version of API

or  Logical or operator

Description

Logical or operator

Usage

or(expr1, expr2)

Arguments

 expr1  Expression 1
 expr2  Expression 2
postGetComments

Returns a list of comments on a post on a user wall or community wall

Description

Returns a list of comments on a post on a user wall or community wall

Usage

postGetComments(
    owner_id = "", 
    post_id = "", 
    need_likes = 1, 
    start_comment_id = "", 
    offset = 0, 
    count = 10, 
    sort = "", 
    preview_length = 0, 
    extended = "", 
    progress_bar = FALSE, 
    v = getAPIVersion()
)

Arguments

owner_id User ID or community ID. Use a negative value to designate a community ID.
post_id Post ID.
need_likes 1 - to return the likes field (default), 0 - not to return the likes field.
start_comment_id Positive number
offset Offset needed to return a specific subset of comments.
count Number of comments to return.
sort Sort order: asc - chronological, desc - reverse chronological.
preview_length Number of characters at which to truncate comments when previewed. Specify 0 (default) if you do not want to truncate comments.
extended Flag, either 1 or 0.
progress_bar Display progress bar
v Version of API
**profile_fields**

*Helper function for working with profile fields*

### Description

Helper function for working with profile fields

### Usage

```r
profile_fields(fields = "")
```

### Arguments

- **fields**
  Profile fields to return

### Examples

```r
## Not run:
# get list of all fields
fields <- profile_fields('all')

# get list of all fields except specified
fields <- profile_fields('all - photo_50,photo_100,photo_200')

# get only specified fields
fields <- profile_fields('sex,bdate')

## End(Not run)
```

---

**queryBuilder**

*Returns a query string*

### Description

Returns a query string

### Usage

```r
queryBuilder(method_name, ...)
```

### Arguments

- **method_name**
  Method name

- **...**
  Method arguments
repeat_last_query  Repeat last function call

Description
Repeat last function call

Usage
repeat_last_query(params = list(), n = 1)

Arguments
- params: Query params
- n: The number of generations to go back

request_delay  Delaying a request if necessary

Description
VK can accept maximum 3 requests to API methods per second from a client.

Usage
request_delay()

saveAsGEXF  Converts the given igraph object to GEXF format and saves it at the given filepath location

Description
Converts the given igraph object to GEXF format and saves it at the given filepath location

Usage
saveAsGEXF(g, filepath = "converted_graph.gexf")

Arguments
- g: Input igraph object to be converted to gexf format
- filepath: File location where the output gexf file should be saved

Author(s)
Gopalakrishna Palem, <Gopalakrishna.Palem@Yahoo.com>
**search.getHints**  
*Allows the programmer to do a quick search for any substring*

**Description**

Allows the programmer to do a quick search for any substring

**Usage**

```python
search.getHints(
    q = "",  
    limit = "",  
    filters = "",  
    search_global = "",  
    v = getAPIVersion()
)
```

**Arguments**

- `q`  
  Search query string
- `limit`  
  Maximum number of results to return
- `filters`  
  List of comma-separated words
- `search_global`  
  Flag, either 1 or 0, default 1
- `v`  
  Version of API

---

**setAccessToken**  
*Set access token*

**Description**

Set access token

**Usage**

```python
setAccessToken(access_token = "")
```

**Arguments**

- `access_token`  
  Access token
### setAPIVersion

**Description**

Set API version

**Usage**

`setAPIVersion(v)`

**Arguments**

- `v`  
  API version

### setRepeats

**Description**

Set maximum number of repeats

**Usage**

`setRepeats(n)`

**Arguments**

- `n`  
  Repeats number

### setTimeout

**Description**

Set timeout

**Usage**

`setTimeout(secs)`

**Arguments**

- `secs`  
  Seconds
show\_collections

\begin{tabular}{ll}
show\_collections & Show collections \\
\end{tabular}

\textbf{Description}

Show collections

\textbf{Usage}

\begin{verbatim}
show\_collections()
\end{verbatim}

show\_dbs

\begin{tabular}{ll}
show\_dbs & Show databases \\
\end{tabular}

\textbf{Description}

Show databases

\textbf{Usage}

\begin{verbatim}
show\_dbs()
\end{verbatim}

tag2Id

\begin{tabular}{ll}
tag2Id & Returns user id by tag \\
\end{tabular}

\textbf{Description}

Returns user id by tag

\textbf{Usage}

\begin{verbatim}
tag2Id(tag)
\end{verbatim}

\textbf{Arguments}

\begin{tabular}{ll}
tag & Tag \\
\end{tabular}
try_handle_error  
*Check response for errors*

**Description**

Check response for errors

**Usage**

```
try_handle_error(response)
```

**Arguments**

- `response`: httr response object

---

try_handle_network_error  
*Try to handle network error*

**Description**

Try to handle network error

**Usage**

```
try_handle_network_error(error)
```

**Arguments**

- `error`: Error

---

**usersGetFollowers**  
*Returns a list of IDs of followers of the user in question, sorted by date added, most recent first*

**Description**

Returns a list of IDs of followers of the user in question, sorted by date added, most recent first
**Usage**

```r
usersGetFollowers(
  user_id = "",
  offset = 0,
  count = 0,
  fields = "",
  name_case = "",
  drop = FALSE,
  flatten = FALSE,
  progress_bar = FALSE,
  v = getAPIVersion()
)
```

**Arguments**

- **user_id**: User ID
- **offset**: Offset needed to return a specific subset of followers
- **count**: Number of followers to return
- **fields**: Profile fields to return
- **name_case**: Case for declension of user name and surname
- **drop**: Drop deleted or banned followers
- **flatten**: Automatically flatten nested data frames into a single non-nested data frame
- **progress_bar**: Display progress bar
- **v**: Version of API

**Examples**

```r
## Not run:
my_followers <- usersGetFollowers(me())
## End(Not run)
```

---

**usersGetSubscriptions**  
Returns a list of IDs of users and communities followed by the user

**Description**

Returns a list of IDs of users and communities followed by the user
Usage

\begin{verbatim}
usersGetSubscriptions(
    user_id = "",
    extended = "1",
    offset = 0,
    count = 0,
    fields = "",
    flatten = FALSE,
    progress_bar = FALSE,
    v = getAPIVersion()
)
\end{verbatim}

Arguments

- **user_id**: User ID
- **extended**: 1 - to return a combined list of users and communities, 0 - to return separate lists of users and communities
- **offset**: Offset needed to return a specific subset of subscriptions
- **count**: Number of users and communities to return
- **fields**: Profile fields to return
- **flatten**: Automatically flatten nested data frames into a single non-nested data frame
- **progress_bar**: Display progress bar
- **v**: Version of API

Examples

```r
## Not run:
my_subscriptions <- usersGetSubscriptions(me())
## End(Not run)
```

---

**usersSearch**

*Returns a list of users matching the search criteria*

Description

Returns a list of users matching the search criteria

Usage

\begin{verbatim}
usersSearch(
    q = "",
    sort = "",
    offset = "",
    count = "20",
)
usersSearch

```r
fields = "", 
city = "", 
country = "", 
hometown = "", 
university_country = "", 
university = "", 
university_year = "", 
university_faculty = "", 
university_chair = "", 
sex = "", 
status = "", 
age_from = "", 
age_to = "", 
birth_day = "", 
birth_month = "", 
birth_year = "", 
online = "", 
has_photo = "", 
school_country = "", 
school_city = "", 
school_class = "", 
school = "", 
school_year = "", 
religion = "", 
interests = "", 
company = "", 
position = "", 
group_id = "", 
from_list = "", 
flatten = FALSE, 
v = getAPIVersion()
)
```

**Arguments**

- **q**
  - Search query string (e.g., Vasya Babich)

- **sort**
  - Sort order: 1 - by date registered; 0 - by rating

- **offset**
  - Offset needed to return a specific subset of users

- **count**
  - Number of users to return

- **fields**
  - Profile fields to return

- **city**
  - City ID

- **country**
  - Country ID

- **hometown**
  - City name in a string

- **university_country**
  - ID of the country where the user graduated

- **university**
  - ID of the institution of higher education
university_year
    Year of graduation from an institution of higher education
university_faculty
    Faculty ID
university_chair
    Chair ID
sex
    1 - female; 2 - male; 0 - any (default)
status
    Relationship status: 1 - Not married; 2 - In a relationship; 3 - Engaged; 4 - Married; 5 - It's complicated; 6 - Actively searching; 7 - In love
age_from
    Minimum age
age_to
    Maximum age
birth_day
    Day of birth
birth_month
    Month of birth
birth_year
    Year of birth
online
    1 - online only; 0 - all users
has_photo
    1 - with photo only; 0 - all users
school_country
    ID of the country where users finished school
school_city
    ID of the city where users finished school
school_class
    Positive number
school
    ID of the school
school_year
    School graduation year
religion
    Users’ religious affiliation
interests
    Users’ interests
company
    Name of the company where users work
position
    Job position
group_id
    ID of a community to search in communities
from_list
    List of comma-separated words
flatten
    Automatically flatten nested data frames into a single non-nested data frame
v
    Version of API

use_db (db_name)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>db_name</td>
<td>Database name</td>
</tr>
</tbody>
</table>
vkApply

Apply a method over a vector of objects

Description

Returns a data frame of the same number of rows as length of ‘objs’, each element of which is the result of applying ‘method’ to the corresponding element of ‘objs’

Usage

vkApply(objs, method)

Arguments

- objs: A vector of objects
- method: The function to be applied to each element of ‘objs’

Examples

```r
## Not run:
users <- vkApply(c("",1234567), function(user) getUsers(user, fields="sex"))
countries <- vkApply(c(2,5122182,1906578), getCountryByCityId)
## End(Not run)
```

vkOAuth

Client authorization

Description

Client authorization

Usage

vkOAuth(client_id, scope = "friends", email, password)

Arguments

- client_id: Application ID
- scope: Requested application access permissions (see below)
- email: Email or phone number
- password: Password
Details

List of Available Settings of Access Permissions:

- **friends** Access to friends.
- **photos** Access to photos.
- **audio** Access to audios.
- **video** Access to videos.
- **docs** Access to documents.
- **notes** Access to user notes.
- **pages** Access to wiki pages.
- **status** Access to user status.
- **wall** Access to standard and advanced methods for the wall.
- **groups** Access to user groups.
- **messages** Access to advanced methods for messaging.
- **notifications** Access to notifications about answers to the user.

Examples

```r
## Not run:
# an example of an authenticated request
vkOAuth(client_id = 123456,
    scope = "friends,groups,messages",
    email = "your_email@example.com",
    password = "your_secret_password")

# save access token for future sessions
at <- getAccessToken()

# an example of request
me()

# an example of an authenticated request without specifying email and password
vkOAuth(client_id = 123456, scope = "friends,groups,messages")

# copy access token from browser address bar
setAccessToken("your_secret_access_token")

## End(Not run)
```
Description

Client authorization (for web application)

Usage

vkOAuthWeb(app_name, client_id, client_secret)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>app_name</td>
<td>Application name</td>
</tr>
<tr>
<td>client_id</td>
<td>Application ID</td>
</tr>
<tr>
<td>client_secret</td>
<td>Application secret key</td>
</tr>
</tbody>
</table>

Description

Create post object

Usage

vkPost(...)  

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>List of attributes</td>
</tr>
</tbody>
</table>
vkR

Access to VK API via R

Description

This package provides a series of functions that allow R users to access VK's API ([https://vk.com/dev/methods](https://vk.com/dev/methods)) to get information about users, messages, groups, posts and likes.

Details

VK ([https://vk.com/](https://vk.com/)) is the largest European online social networking service, based in Russia. It is available in several languages, and is especially popular among Russian-speaking users. VK allows users to message each other publicly or privately, to create groups, public pages and events, share and tag images, audio and video, and to play browser-based games [1].

Author(s)

Dmitriy Sorokin <dementiy@yandex.ru>

References


See Also

vkOAuth, getUsersExecute, getWallExecute, getFriends, getFriendsFor, getGroupsForUsers, getGroupsMembersExecute, likesGetListForObjects, messagesGetHistoryExecute, getArbitraryNetwork, getStatus

vk_stop

Custom error

Description

Custom error

Usage

vk_stop(message = "", call = sys.call(), error_code = ")

Arguments

message Error message
call Call expression
error_code Error code
wallGetById  

**Description**

Returns a list of posts from user or community walls by their IDs

**Usage**

```python
wallGetById(
    posts = "",
    extended = "",
    copy_history_depth = "",
    fields = "",
    v = getAPIVersion()
)
```

**Arguments**

- **posts**  
  User or community IDs and post IDs, separated by underscores. Use a negative value to designate a community ID.

- **extended**  
  1 - to return user and community objects needed to display posts, 0 - no additional fields are returned (default).

- **copy_history_depth**  
  Sets the number of parent elements to include in the array copy_history that is returned if the post is a repost from another wall.

- **fields**  
  List of comma-separated words

- **v**  
  Version of API

**Value**

Returns a list of post objects. If extended is set to 1, returns the following:

- **wall** - Contains post objects.

- **profiles** - Contains user objects with additional fields sex, photo, photo_medium_rec, and online.

- **groups** - Contains community objects.

If the post is a copy of another post, returns an additional array copy_history with information about original posts.
wallGetComments

Returns a list of comments on a post on a user wall or community wall

Description

Returns a list of comments on a post on a user wall or community wall

Usage

wallGetComments(
    owner_id = "", 
    post_id = "", 
    need_likes = "", 
    start_comment_id = "", 
    offset = "", 
    count = "10", 
    sort = "", 
    preview_length = "0", 
    extended = "", 
    v = getAPIVersion() 
)

Arguments

owner_id User ID or community ID. Use a negative value to designate a community ID.
post_id Post ID.
need_likes 1 - to return the likes field, 0 - not to return the likes field (default).
start_comment_id Positive number.
offset Offset needed to return a specific subset of comments.
count Number of comments to return (maximum 100).
sort Sort order: asc - chronological, desc - reverse chronological.
preview_length Number of characters at which to truncate comments when previewed. By default, 90. Specify 0 if you do not want to truncate comments.
extended Flag, either 1 or 0.
v Version of API
wallGetCommentsList  Returns a list of comments on a user wall or community wall

Description
Returns a list of comments on a user wall or community wall

Usage
wallGetCommentsList(posts, progress_bar = FALSE, v = getAPIVersion())

Arguments
- **posts**: A list of posts or wall object (from getWallExecute())
- **progress_bar**: Display progress bar
- **v**: Version of API

wallGetReposts  Returns information about reposts of a post on user wall or community wall

Description
Returns information about reposts of a post on user wall or community wall

Usage
wallGetReposts(
    owner_id = "",  
    post_id = "",  
    offset = "",  
    count = "20",  
    v = getAPIVersion()
)

Arguments
- **owner_id**: User ID or community ID. By default, current user ID. Use a negative value to designate a community ID.
- **post_id**: Post ID.
- **offset**: Offset needed to return a specific subset of reposts.
- **count**: Number of reposts to return.
- **v**: Version of API
Value

Returns an object containing the following fields:

- **items** - An array of wall reposts.
- **profiles** - Information about users with additional fields sex, online, photo, photo_medium_rec,
  and screen_name.
- **groups** - Information about communities.

---

**wallSearch** 

*Allows to search posts on user or community walls*

**Description**

Allows to search posts on user or community walls

**Usage**

```javascript
wallSearch(
    owner_id = "",  
    domain = "",    
    query = "",     
    owners_only = "", 
    count = "20",   
    offset = "0",   
    extended = "",  
    fields = "",    
    v = getAPIVersion()
)  
```

**Arguments**

- **owner_id**  User or community id. Remember that for a community owner_id must be negative.
- **domain**    User or community screen name.
- **query**     Search query string.
- **owners_only** 1 - returns only page owner’s posts.
- **count**     Count of posts to return.
- **offset**    Results offset.
- **extended** Show extended post info.
- **fields**  List of comma-separated words
- **v**        Version of API

**Value**

If executed successfully, returns a list of post objects.
Index

age_predict, 4
areFriends, 4

boardGetComments, 5
boardGetCommentsExecute, 6
boardGetCommentsList, 7

clear_text, 7
collection_exists, 8
create_empty_collection, 8

databaseGetChairs, 9
databaseGetCities, 9
databaseGetCitiesById, 10
databaseGetCountries, 11
databaseGetCountriesById, 11
databaseGetFaculties, 12
databaseGetRegions, 13
databaseGetSchoolClasses, 13
databaseGetSchools, 14
databaseGetStreetsById, 15
databaseGetUniversities, 15
db_drop, 16
db_drop_collection, 16
db_get_collection, 17
db_get_connection, 18
db_getActive, 17
db_getName, 17
db_init, 18
db_insert, 19
db_load, 19
db_load_collection, 20
db_metaConnection, 20
db_save, 21
db_update, 21
execute, 22
filterAttachments, 22
get_stop_words, 49
getAccessToken, 23
getArbitraryNetwork, 23, 74
getCountryByCityId, 23
getEgoNetwork, 24
getFriends, 24, 74
getFriendsBy25, 25
getFriendsFor, 26, 74
groups, 26
groupsById, 27
groupsForUsers, 28, 74
groupsMembers, 28
groupsMembersExecute, 29, 74
getMutual, 30
groupsMutualExecute, 31
groupsPaths, 32
groupsStatus, 32, 74
groupsTopics, 33
groupsTopicsExecute, 34
groupsURLs, 35
groupsUsers, 36
groupsUsersExecute, 41, 74
groupsWall, 46
groupsWallExecute, 47, 74
groupsSearch, 49
handle_captcha, 50
handle_validation, 51
has_error, 51
likesGetList, 51
likesGetListForObjects, 52, 74
me, 53
messagesGet, 54
messagesGetHistory, 55
messagesGetHistoryAll, 55
messagesGetHistoryExecute, 56, 74
messagesSend, 57
messagesSplitByDate, 58
newsfeedSearch, 58
or, 59

getPostComments, 60
profile_fields, 61

queryBuilder, 61

repeat_last_query, 62
request_delay, 62

saveAsGEXF, 62
search.getHints, 63
setAccessToken, 63
setAPIVersion, 64
setRepeats, 64
setTimeout, 64
show_collections, 65
show_dbs, 65

tag2Id, 65
try_handle_error, 66
try_handle_network_error, 66

use_db, 70
usersGetFollowers, 66
usersGetSubscriptions, 67
usersSearch, 68

vk_stop, 74
vkApply, 71
vkOAuth, 71, 74
vkOAuthWeb, 73
vkPost, 73
vkR, 74

wallGetById, 75
wallGetComments, 76
wallGetCommentsList, 77
wallGetReposts, 77
wallSearch, 78