Elogram

Release

November 08, 2016

Contents

Elogram is an “eloquent” way of accessing Instagram’s API, for PHP 5.5+. It offers:

- Simple interface for interacting with the web API. Use methods and arguments instead of cURL.
- Provides an easy-to-use CSRF-protected solution for retrieving access tokens.
- Well-tested
- Extensible
- Makes pagination a breeze.

```php
use Larabros\Elogram\Client;

$client = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);

header('Content-Type: application/json');
$response = $client->media()->search(51.503349, -0.252271);
echo json_encode($response->get());

$response = $client->users()->find('skrawg');
echo json_encode($response->get());

$response  = $client->users()->follows();
echo json_encode($response->get());

$response  = $client->paginate($response, 2);
echo json_encode($response->get());
```

CHAPTER 1

User Guide

1.1 Overview

1.1.1 Requirements

1. PHP 5.5.9+
2. An Instagram client ID and secret

Note: You can create an Instagram application at https://www.instagram.com/developer/clients/register/

1.1.2 Installation

The recommended way to install Elogram is with Composer. Composer is a dependency management tool for PHP that allows you to declare the dependencies your project needs and installs them into your project.

```
# Install Composer
curl -sS https://getcomposer.org/installer | php
```

You can add Elogram as a dependency using the `composer.phar` CLI:

```
php composer.phar require larabros/elogram:~1.0
```

Alternatively, you can specify it as a dependency in your project's existing `composer.json` file:

```
 {
   "require": {
      "larabros/elogram": "~1.0"
   }
}
```

After installing, you need to require Composer's autoloader:

```
require 'vendor/autoload.php';
```

You can find out more on how to install Composer, configure autoloading, and other best-practices for defining dependencies at getcomposer.org.

1.1.3 License

Licensed using the MIT license.

> Copyright (c) 2016 Hassan Khan <contact@hassankhan.me>
>
> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
>
> The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
>
> THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.2 Quickstart

This page provides a quick introduction to Elogram and introductory examples. If you have not already installed, head over to the *Installation* page.

1.2.1 Authentication

To start making requests, you need to authenticate and retrieve an access token. To do this, first instantiate the `Client` class:

Important: The `$clientId`, `$clientSecret` and `$redirectUrl` **must** be the same as in the Instagram Developer Panel

---

Important: You should always store your client ID and secret as an environment variable, or otherwise out of source control. An excellent tool to help you do this is the vlucas/phpdotenv package.

Creating a Client

```
use Larabros\Elogram\Client;

$client = new Client($clientId, $clientSecret, null, $redirectUrl);
```

Setting up the authentication flow

After instantiating the client, start a session and retrieve the the authorization page URL (or retrieve an access token if the user is already authorized):

```php
// Start the session
session_start();

// If we don't have an authorization code then get one
if (!isset($_GET['code'])) {
    header('Location: ' . $client->getLoginUrl());
    exit;
} else {
    $token = $client->getAccessToken($_GET['code']);
    echo json_encode($token); // Save this for future use
}

// You can now make requests to the API
$client->users()->search('skrawg');
```

Using an existing token

If you have an access token in the form of a JSON string, then instantiate the *Client* class with it:

```php
use Larabros\Elogram\Client;

$client = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
```

You can also add the access token after instantiation:

```php
use Larabros\Elogram\Client;

$client = new Client($clientId, $clientSecret, null, $redirectUrl);

//
// Retrieve the access token from somewhere
//

$client->setAccessToken($token);
```

1.2.2 Login permissions (Scopes)

You can request additional access scopes for the access token by passing an array to the *Client::getLoginUrl()* method:

```php
$options  = ['scope' => 'basic public_content'];
$loginUrl = $client->getLoginUrl($options);
```

Note that the scopes **must** separated by a space. Available scopes are listed on the Instagram Developer website.

1.2.3 Secure Requests

Important: Secure requests **must** be enabled in the Instagram Developer Panel for your application.

Secure requests can be enabled by calling *Client::secureRequests()*.

```php
// Enables secure requests
$client->secureRequests();

// Disables secure requests
$client->secureRequests(false);
```

1.2.4 Sending Requests

Simple requests

To simplify requests to the API, it is recommended you read Endpoints. However, sometimes you may need to make a call to the API without syntactic sugar; for this you can use *`Client::request()`*:

```php
use Larabros\Elogram\Client;

$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->request('GET', 'users/self');
echo json_encode($response->get());
```

Paginated Requests

The *Response* object that you receive from making requests contains the data from the multiple requests combined, including the first one. You can also pass a `$limit` as an optional parameter to *`Client::paginate()`*, which sets the number of pages to request, assuming they are available. If `$limit` is not provided, as many pages as available will be requested.

Important: Not setting the `$limit` parameter may cause timeout issues. Be careful of how and where you use it.

```php
use Larabros\Elogram\Client;

$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);

// Get initial response
$response = $client->users()->follows();
echo json_encode($response->get());

// Get next two pages of results
$response = $client->paginate($response, 2);
echo json_encode($response->get());

// Get as many pages as available
$response = $client->paginate($response);
echo json_encode($response->get());
```

1.3 Endpoints

This page provides an extensive list of all endpoints supported by Elogram.

Warning: All code examples below assume you have already instantiated the *`Client`* class with a valid access token. If you do not already have one, head over to the *Authentication* page to read about how to obtain one.

1.3.1 Users

get (*$id = 'self'*)
Get information about a user.

Parameters

- **$id** (*string*) – The ID of the user. Default is `self`

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->users()->get(4);
echo json_encode($response->get());
```

Example response:

```
{
    "meta":
    {
        "code": 200
    },
    "data":
    {
        "username": "mikeyk",
        "first_name": "Mike",
        "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_4_75sq_129232474
        "id": "4",
        "last_name": "Krieger!!"
    }
}
```

getMedia (*$id = 'self', $count = null, $minId = null, $maxId = null*)
Get the most recent media published by a user.

Parameters

- **$id** –
- **$count** (*int|null*) – Count of media to return
- **$minId** (*int|null*) – Return media later than this min_id
- **$maxId** (*int|null*) – Return media earlier than this max_id

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->users()->getMedia('268047373');
echo json_encode($response->get());
```

Example response:

```
{
    "pagination":
    {
        "next_url": "http://localhost:8000/publicapi/v1/users/4/media/recent?q=iphone&lat=37.771
        "next_max_id": 3382522
    },
```

```
        "meta":
        {
            "code": 200
        },
        "data": [
            {
                "type": "image",
                "tags": [],
                "location":
                {
                    "latitude": 37.781109999999998,
                    "id": null,
                    "longitude": -122.3947,
                    "name": null
                },
                "comments":
                {
                    "count": 1,
                    "data": [
                        {
                            "created_time": "1296713291",
                            "text": "Earear",
                            "from":
                            {
                                "username": "mikeyk",
                                "first_name": "Mike",
                                "last_name": "Krieger!!",
                                "type": "user",
                                "id": "4"
                            },
                            "id": "2611728"
                        }]
                },
                "caption":
                {
                    "created_time": "1296713291",
                    "text": "Earear",
                    "from":
                    {
                        "username": "mikeyk",
                        "first_name": "Mike",
                        "last_name": "Krieger!!",
                        "type": "user",
                        "id": "4"
                    },
                    "id": "2611728"
                },
                "link": "http://localhost:8000/p/M5z6/",
                "likes":
                {
                    "count": 0
                },
                "created_time": "1296713289",
                "images":
                {
                    "low_resolution":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2011/02/02/1ce5f3f490a6
```

```
                    "width": 480,
                    "height": 480
                },
                "thumbnail":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2011/02/02/1ce5f3f490a6
                    "width": 150,
                    "height": 150
                },
                "standard_resolution":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2011/02/02/1ce5f3f490a6
                    "width": 612,
                    "height": 612
                }
            },
            "user_has_liked": false,
            "id": "3382522",
            "user":
            {
                "username": "mikeyk",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_4_75sq_1
                "id": "4"
            }
        },
        {
            "type": "video",
            "videos":
            {
                "low_resolution":
                {
                    "url": "http://distilleryvesper9-13.ak.instagram.com/090d06dad9cd11e2aa09123
                    "width": 480,
                    "height": 480
                },
                "standard_resolution":
                {
                    "url": "http://distilleryvesper9-13.ak.instagram.com/090d06dad9cd11e2aa09123
                    "width": 640,
                    "height": 640
                }
            },
            "users_in_photo": null,
            "filter": "Vesper",
            "tags": [],
            "comments":
            {
                "data": [
                    {
                        "created_time": "1279332030",
                        "text": "Love the sign here",
                        "from":
                        {
                            "username": "mikeyk",
                            "full_name": "Mikey Krieger",
                            "id": "4",
                            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/prof
                        },
```

```
                    "id": "8"
                },
                {
                    "created_time": "1279341004",
                    "text": "Chilako taco",
                    "from":
                    {
                        "username": "kevin",
                        "full_name": "Kevin S",
                        "id": "3",
                        "profile_picture": "..."
                    },
                    "id": "3"
                }],
            "count": 2
        },
        "caption": null,
        "likes":
        {
            "count": 1,
            "data": [
                {
                    "username": "mikeyk",
                    "full_name": "Mikeyk",
                    "id": "4",
                    "profile_picture": "..."
                }]
        },
        "link": "http://instagr.am/p/D/",
        "user":
        {
            "username": "kevin",
            "full_name": "Kevin S",
            "profile_picture": "...",
            "bio": "...",
            "website": "...",
            "id": "3"
        },
        "created_time": "1279340983",
        "images":
        {
            "low_resolution":
            {
                "url": "http://distilleryimage2.ak.instagram.com/11f75f1cd9cc11e2a0fd22000aa
                "width": 306,
                "height": 306
            },
            "thumbnail":
            {
                "url": "http://distilleryimage2.ak.instagram.com/11f75f1cd9cc11e2a0fd22000aa
                "width": 150,
                "height": 150
            },
            "standard_resolution":
            {
                "url": "http://distilleryimage2.ak.instagram.com/11f75f1cd9cc11e2a0fd22000aa
                "width": 612,
                "height": 612
```

```
                    }
                },
                "id": "3",
                "location": null
            }

        ]
    }
```

getLikedMedia (*$count = null*, *$maxLikeId = null*)

Get the list of recent media liked by the owner of the access token.

Parameters

- **$count** (*int|null*) – Count of media to return
- **$maxLikeId** (*int|null*) – Return media liked before this id

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->users()->getLikedMedia();
echo json_encode($response->get());
```

Example response:

```
{
    "pagination":
    {
        "next_url": "http://localhost:8000/publicapi/v1/users/self/feed?q=iphone&lat=37.771&acce
        "next_max_like_id": 50
    },
    "meta":
    {
        "code": 200
    },
    "data": [
        {
            "type": "image",
            "tags": [],
            "location":
            {
                "latitude": 37.781089999999999,
                "id": null,
                "longitude": -122.3946,
                "name": null
            },
            "comments":
            {
                "count": 1,
                "data": [
                    {
                        "created_time": "1296272101",
                        "text": "O hai.",
                        "from":
                        {
                            "username": "mikeyk",
                            "first_name": "Mike",
                            "last_name": "Krieger!!",
```

```
                                "type": "user",
                                "id": "4"
                            },
                            "id": "2611719"
                        }]
                },
                "caption":
                {
                    "created_time": "1296272101",
                    "text": "O hai.",
                    "from":
                    {
                        "username": "mikeyk",
                        "first_name": "Mike",
                        "last_name": "Krieger!!",
                        "type": "user",
                        "id": "4"
                    },
                    "id": "2611719"
                },
                "link": "http://localhost:8000/p/M5z4/",
                "likes":
                {
                    "count": 0
                },
                "created_time": "1296673135",
                "images":
                {
                    "low_resolution":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2011/02/02/b6073350d896
                        "width": 480,
                        "height": 480
                    },
                    "thumbnail":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2011/02/02/b6073350d896
                        "width": 150,
                        "height": 150
                    },
                    "standard_resolution":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2011/02/02/b6073350d896
                        "width": 612,
                        "height": 612
                    }
                },
                "user_has_liked": false,
                "id": "3382520",
                "user":
                {
                    "username": "mikeyk",
                    "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_4_75sq_1
                    "id": "4"
                }
            }]
    }
```

search (*$query*, *$count = null*)
Get a list of users matching the query.

Parameters

- **$query** –
- **$count** –

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->users()->search('skrawg');
echo json_encode($response->get());
```

Example response:

```
{
    "meta":
    {
        "code": 200
    },
    "data": [
        {
            "username": "mikeyk",
            "first_name": "Mike",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_4_75sq_12923
            "id": "4",
            "last_name": "Krieger!!"
        }]
}
```

find (*$username*)
Searches for and returns a single user's information. If no results are found, `null` is returned.

Parameters

- **$username** (*string*) – A username to search for

Returns Response|null

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->users()->find('mikeyk');
echo json_encode($response->get());
```

Example response:

```
{
    "meta":
    {
        "code": 200
    },
    "data": {
        "username": "mikeyk",
        "first_name": "Mike",
        "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_4_75sq_129232474
        "id": "4",
        "last_name": "Krieger!!"
```

```
        }
    }
```

1.3.2 Relationships

follows()
Get the list of users this user follows.

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->users()->follows();
echo json_encode($response->get());
```

Example response:

```
{
    "pagination":
    {
        "next_url": "https://api.instagram.com/v1/users/self/follows?cursor=1201686",
        "next_cursor": "1201686"
    },
    "meta":
    {
        "code": 200
    },
    "data": [
        {
            "username": "kevin",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_3_75sq_12954
            "id": "3"
        },
        {
            "username": "moop55_ok",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_332654_75sq_
            "id": "332654"
        },
        {
            "username": "ooop",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_580183_75sq_
            "id": "580183"
        },
        {
            "username": "tastyboo",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_580178_75sq_
            "id": "580178"
        },
        {
            "username": "moopafa",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.jpg",
            "id": "580174"
        },
        {
            "username": "jalter",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.png",
            "id": "51"
```

```
        },
        {
            "username": "doug",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_17_75sq_1287
            "id": "17"
        },
        {
            "username": "danman",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_139329_75sq_
            "id": "139329"
        },
        {
            "username": "amine",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_136_75sq_128
            "id": "136"
        },
        {
            "username": "moopevil3",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.jpg",
            "id": "580164"
        },
        {
            "username": "kevnull",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_12169_75sq_1
            "id": "12169"
        },
        {
            "username": "wymp",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.jpg",
            "id": "402370"
        },
        {
            "username": "rad",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_330307_75sq_
            "id": "330307"
        },
        {
            "username": "mpreysman",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_2431_75sq_12
            "id": "2431"
        },
        {
            "username": "joulee",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_354221_75sq_
            "id": "354221"
        },
        {
            "username": "drtyhbo",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.jpg",
            "id": "451846"
        },
        {
            "username": "pm",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.jpg",
            "id": "21017"
        },
        {
            "username": "mike_matas",
```

```
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.jpg",
            "id": "283400"
        },
        {
            "username": "joehewitt",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.jpg",
            "id": "340483"
        },
        {
            "username": "itod",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_73142_75sq_1
            "id": "73142"
        },
        {
            "username": "erickschonfeld",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_412333_75sq_
            "id": "412333"
        },
        {
            "username": "edog1203",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.jpg",
            "id": "334077"
        },
        {
            "username": "addumb",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_156856_75sq_
            "id": "156856"
        },
        {
            "username": "neo121",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_4124_75sq_12
            "id": "4124"
        },
        {
            "username": "heathsplosion",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_4962_75sq_12
            "id": "4962"
        },
        {
            "username": "baristaskills",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_400745_75sq_
            "id": "400745"
        },
        {
            "username": "mephaust",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.jpg",
            "id": "379207"
        },
        {
            "username": "zuck",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.jpg",
            "id": "314216"
        },
        {
            "username": "kikife",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_89613_75sq_1
            "id": "89613"
        },
```

```
        {
            "username": "nathanfolkman",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_10331_75sq_1
            "id": "10331"
        },
        {
            "username": "szetopia",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_324431_75sq_
            "id": "324431"
        },
        {
            "username": "savasavasava",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_230975_75sq_
            "id": "230975"
        },
        {
            "username": "marcoarment",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_319323_75sq_
            "id": "319323"
        },
        {
            "username": "jonathan",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_104_75sq_128
            "id": "104"
        },
        {
            "username": "marcin",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_3436_75sq_12
            "id": "3436"
        },
        {
            "username": "sara",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_3843_75sq_12
            "id": "3843"
        },
        {
            "username": "garrytan",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_118666_75sq_
            "id": "118666"
        },
        {
            "username": "bijan",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_4415_75sq_12
            "id": "4415"
        },
        {
            "username": "kevinrose",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_124163_75sq_
            "id": "124163"
        },
        {
            "username": "joshjohnson",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_98476_75sq_1
            "id": "98476"
        },
        {
            "username": "andreklbacelar",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_9456_75sq_12
```

```
                "id": "9456"
            },
            {
                "username": "sacca",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_221307_75sq_
                "id": "221307"
            },
            {
                "username": "suzykrieger",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_220658_75sq_
                "id": "220658"
            },
            {
                "username": "nandakbp",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_269007_75sq_
                "id": "269007"
            },
            {
                "username": "jpollock",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.jpg",
                "id": "157011"
            },
            {
                "username": "iphonequeen",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_6392_75sq_12
                "id": "6392"
            },
            {
                "username": "igutfreund",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.jpg",
                "id": "248626"
            },
            {
                "username": "onwall",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/anonymousUser.jpg",
                "id": "248299"
            },
            {
                "username": "missmallibu",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_235309_75sq_
                "id": "235309"
            },
            {
                "username": "jimgoldstein",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_171983_75sq_
                "id": "171983"
            }]
    }
```


followedBy()

Get the list of users this user is followed by.

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->users()->followedBy();
echo json_encode($response->get());
```

Example response:

```
{
    "meta":
    {
        "code": 200
    },
    "data": [
        {
            "username": "shayne",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_20_75sq_1295
            "id": "20"
        }]
}
```

requestedBy()

List the users who have requested this user's permission to follow.

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->users()->requestedBy();
echo json_encode($response->get());
```

Example response:

```
{
    "pagination":
    {},
    "meta":
    {
        "code": 200
    },
    "data": [
        {
            "username": "kevin",
            "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_3_75sq_12954
            "id": "3"
        }]
}
```

getRelationship(*$targetUserId*)

Get information about the relationship of the owner of the access token to another user.

Parameters

- **$targetUserId** (*string*) – The ID of the target user

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->users()->getRelationship('268047373');
echo json_encode($response->get());
```

Example response:

```
{
  "data":
```

```
    {
      "outgoing_status": "none",
      "incoming_status": "requested_by"
    }
}
```

setRelationship (*$targetUserId*, *$action*)
Modify the relationship between the owner of the access token and the target user.

Parameters

- **$targetUserId** (*string*) – The ID of the target user
- **$action** (*string*) – Can be one of: `follow | unfollow | approve | ignore`

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->users()->setRelationship('268047373', 'follows');
echo json_encode($response->get());
```

Example response:

```
{
    "meta":
    {
        "code": 200
    },
    "data":
    {
        "outgoing_status": "follows"
    }
}
```

1.3.3 Media

get (*$id*)
Get information about a media object.

Parameters

- **$id** (*string*) – The ID of the media object

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->media()->get('315');
echo json_encode($response->get());
```

Example response:

```
{
    "meta":
    {
        "code": 200
    },
```

```
        "data":
        {
            "type": "image",
            "location":
            {
                "latitude": 37.782179999999997,
                "id": null,
                "longitude": -122.38849999999999,
                "name": null
            },
            "comments":
            {
                "count": 6,
                "data": [
                    {
                        "created_time": "1280379782",
                        "text": "Next muni is in an hour; forget that, walking home instead",
                        "from":
                        {
                            "username": "mikeyk",
                            "first_name": "Mike",
                            "last_name": "Krieger!!",
                            "type": "user",
                            "id": "4"
                        },
                        "id": "367"
                    },
                    {
                        "created_time": "1280417247",
                        "text": "I'm on the new Burbn!!! So beautiful, @kevin and @mikeyk!",
                        "from":
                        {
                            "username": "grex",
                            "first_name": "Gregor",
                            "last_name": "Hochmuth",
                            "type": "user",
                            "id": "25"
                        },
                        "id": "384"
                    },
                    {
                        "created_time": "1296711880",
                        "text": "hawwoo",
                        "from":
                        {
                            "username": "mikeyk",
                            "first_name": "Mike",
                            "last_name": "Krieger!!",
                            "type": "user",
                            "id": "4"
                        },
                        "id": "2611722"
                    },
                    {
                        "created_time": "1296712932",
                        "text": "foo",
                        "from":
                        {
```

```
                        "username": "mikeyk",
                        "first_name": "Mike",
                        "last_name": "Krieger!!",
                        "type": "user",
                        "id": "4"
                    },
                    "id": "2611725"
                },
                {
                    "created_time": "1296712932",
                    "text": "foo",
                    "from":
                    {
                        "username": "mikeyk",
                        "first_name": "Mike",
                        "last_name": "Krieger!!",
                        "type": "user",
                        "id": "4"
                    },
                    "id": "2611726"
                },
                {
                    "created_time": "1296770323",
                    "text": "o_hai",
                    "from":
                    {
                        "username": "mikeyk",
                        "first_name": "Mike",
                        "last_name": "Krieger!!",
                        "type": "user",
                        "id": "4"
                    },
                    "id": "2611747"
                }]
            },
            "caption": null,
            "link": "https://www.instagram.com/p/9mDRRppRE7/",
            "likes":
            {
                "count": 3
            },
            "created_time": "1280379741",
            "images":
            {
                "low_resolution":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/07/28/84967fafe5f44b43
                    "width": 480,
                    "height": 480
                },
                "thumbnail":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/07/28/84967fafe5f44b43
                    "width": 150,
                    "height": 150
                },
                "standard_resolution":
                {
```

```
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/07/28/84967fafe5f44b43
                    "width": 612,
                    "height": 612
                }
            },
            "user_has_liked": true,
            "id": "315",
            "user":
            {
                "username": "mikeyk",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_4_75sq_12923
                "id": "4"
            }
        }
    }
```

getByShortcode (*$shortcode*)

This method returns the same response as `Media::get`

Parameters

- **$shortcode** (*string*) – The shortcode of the media object

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->media()->getByShortcode('9mDRRppRE7');
echo json_encode($response->get());
```

Example response:

```
{
    "meta":
    {
        "code": 200
    },
    "data":
    {
        "type": "image",
        "location":
        {
            "latitude": 37.782179999999997,
            "id": null,
            "longitude": -122.38849999999999,
            "name": null
        },
        "comments":
        {
            "count": 6,
            "data": [
                {
                    "created_time": "1280379782",
                    "text": "Next muni is in an hour; forget that, walking home instead",
                    "from":
                    {
                        "username": "mikeyk",
                        "first_name": "Mike",
                        "last_name": "Krieger!!",
```

```
                "type": "user",
                "id": "4"
            },
            "id": "367"
        },
        {
            "created_time": "1280417247",
            "text": "I'm on the new Burbn!!! So beautiful, @kevin and @mikeyk!",
            "from":
            {
                "username": "grex",
                "first_name": "Gregor",
                "last_name": "Hochmuth",
                "type": "user",
                "id": "25"
            },
            "id": "384"
        },
        {
            "created_time": "1296711880",
            "text": "hawwoo",
            "from":
            {
                "username": "mikeyk",
                "first_name": "Mike",
                "last_name": "Krieger!!",
                "type": "user",
                "id": "4"
            },
            "id": "2611722"
        },
        {
            "created_time": "1296712932",
            "text": "foo",
            "from":
            {
                "username": "mikeyk",
                "first_name": "Mike",
                "last_name": "Krieger!!",
                "type": "user",
                "id": "4"
            },
            "id": "2611725"
        },
        {
            "created_time": "1296712932",
            "text": "foo",
            "from":
            {
                "username": "mikeyk",
                "first_name": "Mike",
                "last_name": "Krieger!!",
                "type": "user",
                "id": "4"
            },
            "id": "2611726"
        },
        {
```

```
                        "created_time": "1296770323",
                        "text": "o_hai",
                        "from":
                        {
                            "username": "mikeyk",
                            "first_name": "Mike",
                            "last_name": "Krieger!!",
                            "type": "user",
                            "id": "4"
                        },
                        "id": "2611747"
                    }]
            },
            "caption": null,
            "link": "https://www.instagram.com/p/9mDRRppRE7/",
            "likes":
            {
                "count": 3
            },
            "created_time": "1280379741",
            "images":
            {
                "low_resolution":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/07/28/84967fafe5f44b43
                    "width": 480,
                    "height": 480
                },
                "thumbnail":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/07/28/84967fafe5f44b43
                    "width": 150,
                    "height": 150
                },
                "standard_resolution":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/07/28/84967fafe5f44b43
                    "width": 612,
                    "height": 612
                }
            },
            "user_has_liked": true,
            "id": "315",
            "user":
            {
                "username": "mikeyk",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_4_75sq_12923
                "id": "4"
            }
        }
    }
```


search (*$latitude*, *$longitude*, *$distance = 1000*)

Search for recent media in a given area.

Parameters

- **$latitude** (*int*) – Latitude of the center search coordinate. If used, `$longitude` is required

- **$longitude** (*int*) – Longitude of the center search coordinate. If used, `$latitude` is required
- **$distance** (*int*) – The distance in metres. Default is `1000` m, max distance is 5km

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->media()->search(37.78, -122.22);
echo json_encode($response->get());
```

Example response:

```
{
    "meta":
    {
        "code": 200
    },
    "data": [
        {
            "type": "image",
            "tags": [],
            "location":
            {
                "latitude": 37.775382999999991,
                "id": "74480",
                "longitude": -122.223941,
                "name": "Fruitvale BART"
            },
            "comments":
            {
                "count": 1,
                "data": [
                    {
                        "created_time": "1288931764",
                        "text": "Emergency",
                        "from":
                        {
                            "username": "catchfoot",
                            "first_name": "Rachel",
                            "last_name": "Lightfoot",
                            "type": "user",
                            "id": "208329"
                        },
                        "id": "1916879"
                    }]
            },
            "caption":
            {
                "created_time": "1288931764",
                "text": "Emergency",
                "from":
                {
                    "username": "catchfoot",
                    "first_name": "Rachel",
                    "last_name": "Lightfoot",
                    "type": "user",
                    "id": "208329"
```

```
                    },
                    "id": "1916879"
                },
                "link": "https://www.instagram.com/p/9mDASDBRpR/",
                "likes":
                {
                    "count": 0
                },
                "created_time": "1288931757",
                "images":
                {
                    "low_resolution":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2010/11/04/486c002f2fd1
                        "width": 480,
                        "height": 480
                    },
                    "thumbnail":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2010/11/04/486c002f2fd1
                        "width": 150,
                        "height": 150
                    },
                    "standard_resolution":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2010/11/04/486c002f2fd1
                        "width": 612,
                        "height": 612
                    }
                },
                "user_has_liked": false,
                "id": "2582155",
                "user":
                {
                    "username": "catchfoot",
                    "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_208329_7
                    "id": "208329"
                }
            },
            {
                "type": "image",
                "tags": [],
                "location":
                {
                    "latitude": 37.775289999999998,
                    "id": "163042",
                    "longitude": -122.224172,
                    "name": "Powderface"
                },
                "comments":
                {
                    "count": 1,
                    "data": [
                        {
                            "created_time": "1288644742",
                            "text": "Caramel Beignets! Yummy :)",
                            "from":
                            {
```

```
                        "username": "mkram0s",
                        "first_name": "Kate",
                        "last_name": "Ramos",
                        "type": "user",
                        "id": "305504"
                    },
                    "id": "1494733"
                }]
            },
            "caption":
            {
                "created_time": "1288644742",
                "text": "Caramel Beignets! Yummy :)",
                "from":
                {
                    "username": "mkram0s",
                    "first_name": "Kate",
                    "last_name": "Ramos",
                    "type": "user",
                    "id": "305504"
                },
                "id": "1494733"
            },
            "link": "https://www.instagram.com/p/9mDRRppH9R/",
            "likes":
            {
                "count": 0
            },
            "created_time": "1288644633",
            "images":
            {
                "low_resolution":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/11/01/7a74951a954a
                    "width": 480,
                    "height": 480
                },
                "thumbnail":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/11/01/7a74951a954a
                    "width": 150,
                    "height": 150
                },
                "standard_resolution":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/11/01/7a74951a954a
                    "width": 612,
                    "height": 612
                }
            },
            "user_has_liked": false,
            "id": "2086015",
            "user":
            {
                "username": "mkram0s",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_305504_7
                "id": "305504"
            }
```

```
        },
        {
            "type": "image",
            "tags": [],
            "location":
            {
                "latitude": 37.775382999999991,
                "id": "74480",
                "longitude": -122.223941,
                "name": "Fruitvale BART"
            },
            "comments":
            {
                "count": 1,
                "data": [
                    {
                        "created_time": "1288232768",
                        "text": "Bored on Bart ",
                        "from":
                        {
                            "username": "hannahc",
                            "first_name": "Hannah",
                            "last_name": "Coughlin ",
                            "type": "user",
                            "id": "304020"
                        },
                        "id": "1030446"
                    }]
            },
            "caption":
            {
                "created_time": "1288232768",
                "text": "Bored on Bart ",
                "from":
                {
                    "username": "hannahc",
                    "first_name": "Hannah",
                    "last_name": "Coughlin ",
                    "type": "user",
                    "id": "304020"
                },
                "id": "1030446"
            },
            "link": "http://localhost:8000/p/Fm-m/",
            "likes":
            {
                "count": 0
            },
            "created_time": "1288232753",
            "images":
            {
                "low_resolution":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/27/1406596f54be
                    "width": 480,
                    "height": 480
                },
                "thumbnail":
```

```
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/27/1406596f54be
                        "width": 150,
                        "height": 150
                    },
                    "standard_resolution":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/27/1406596f54be
                        "width": 612,
                        "height": 612
                    }
                },
                "user_has_liked": false,
                "id": "1470374",
                "user":
                {
                    "username": "hannahc",
                    "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_304020_7
                    "id": "304020"
                }
            },
            {
                "type": "image",
                "tags": [],
                "location":
                {
                    "latitude": 37.775382999999991,
                    "id": "74480",
                    "longitude": -122.223941,
                    "name": "Fruitvale BART"
                },
                "comments":
                {
                    "count": 1,
                    "data": [
                        {
                            "created_time": "1288136425",
                            "text": "Emergency",
                            "from":
                            {
                                "username": "rolliefingaz",
                                "first_name": "Gregory",
                                "last_name": "Hurcomb",
                                "type": "user",
                                "id": "233618"
                            },
                            "id": "951869"
                        }]
                },
                "caption":
                {
                    "created_time": "1288136425",
                    "text": "Emergency",
                    "from":
                    {
                        "username": "rolliefingaz",
                        "first_name": "Gregory",
                        "last_name": "Hurcomb",
```

```
                    "type": "user",
                    "id": "233618"
                },
                "id": "951869"
            },
            "link": "http://localhost:8000/p/FOZf/",
            "likes":
            {
                "count": 0
            },
            "created_time": "1288136246",
            "images":
            {
                "low_resolution":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/26/3459f1db2d6d
                    "width": 480,
                    "height": 480
                },
                "thumbnail":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/26/3459f1db2d6d
                    "width": 150,
                    "height": 150
                },
                "standard_resolution":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/26/3459f1db2d6d
                    "width": 612,
                    "height": 612
                }
            },
            "user_has_liked": false,
            "id": "1369695",
            "user":
            {
                "username": "rolliefingaz",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_233618_7
                "id": "233618"
            }
        },
        {
            "type": "image",
            "tags": [],
            "location":
            {
                "latitude": 37.775194499999998,
                "id": "101132",
                "longitude": -122.227087,
                "name": "Guadalajara Mexican Cuisine and Bar"
            },
            "comments":
            {
                "count": 2,
                "data": [
                    {
                        "created_time": "1287969851",
                        "text": "Well hello there beautiful",
```

```
                        "from":
                        {
                            "username": "jorstaff",
                            "first_name": "Jordan",
                            "last_name": "Walker",
                            "type": "user",
                            "id": "154425"
                        },
                        "id": "818311"
                    },
                    {
                        "created_time": "1288072257",
                        "text": "Text me next time. I live down the st and I'm ways down for tac
                        "from":
                        {
                            "username": "melisaarreguin",
                            "first_name": "Melisa",
                            "last_name": "Arreguin",
                            "type": "user",
                            "id": "122725"
                        },
                        "id": "898954"
                    }]
            },
            "caption":
            {
                "created_time": "1287969851",
                "text": "Well hello there beautiful",
                "from":
                {
                    "username": "jorstaff",
                    "first_name": "Jordan",
                    "last_name": "Walker",
                    "type": "user",
                    "id": "154425"
                },
                "id": "818311"
            },
            "link": "http://localhost:8000/p/Ejh5/",
            "likes":
            {
                "count": 1
            },
            "created_time": "1287969827",
            "images":
            {
                "low_resolution":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/24/8a466d809fe4
                    "width": 480,
                    "height": 480
                },
                "thumbnail":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/24/8a466d809fe4
                    "width": 150,
                    "height": 150
                },
```

```
                    "standard_resolution":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/24/8a466d809fe4
                        "width": 612,
                        "height": 612
                    }
                },
                "user_has_liked": false,
                "id": "1194105",
                "user":
                {
                    "username": "jorstaff",
                    "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_154425_7
                    "id": "154425"
                }
            },
            {
                "type": "image",
                "tags": [],
                "location":
                {
                    "latitude": 37.775382999999991,
                    "id": "74480",
                    "longitude": -122.223941,
                    "name": "Fruitvale BART"
                },
                "comments":
                {
                    "count": 1,
                    "data": [
                        {
                            "created_time": "1287674437",
                            "text": "Morning commute",
                            "from":
                            {
                                "username": "catchfoot",
                                "first_name": "Rachel",
                                "last_name": "Lightfoot",
                                "type": "user",
                                "id": "208329"
                            },
                            "id": "555932"
                        }]
                },
                "caption":
                {
                    "created_time": "1287674437",
                    "text": "Morning commute",
                    "from":
                    {
                        "username": "catchfoot",
                        "first_name": "Rachel",
                        "last_name": "Lightfoot",
                        "type": "user",
                        "id": "208329"
                    },
                    "id": "555932"
                },
```

```
                "link": "http://localhost:8000/p/DMl1/",
                "likes":
                {
                    "count": 0
                },
                "created_time": "1287674360",
                "images":
                {
                    "low_resolution":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/21/d9949ca81cb5
                        "width": 480,
                        "height": 480
                    },
                    "thumbnail":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/21/d9949ca81cb5
                        "width": 150,
                        "height": 150
                    },
                    "standard_resolution":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/21/d9949ca81cb5
                        "width": 612,
                        "height": 612
                    }
                },
                "user_has_liked": false,
                "id": "838005",
                "user":
                {
                    "username": "catchfoot",
                    "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_208329_7
                    "id": "208329"
                }
            },
            {
                "type": "image",
                "tags": [],
                "location":
                {
                    "latitude": 37.775180799999987,
                    "id": "68841",
                    "longitude": -122.2270716,
                    "name": "El Novillo Taco Truck"
                },
                "comments":
                {
                    "count": 4,
                    "data": [
                        {
                            "created_time": "1287585720",
                            "text": "Carne Asasa Torta. No Onions. ",
                            "from":
                            {
                                "username": "darodriguez",
                                "first_name": "David",
                                "last_name": "Rodriguez",
```

```
                            "type": "user",
                            "id": "113603"
                        },
                        "id": "495555"
                    },
                    {
                        "created_time": "1287585854",
                        "text": "Hey, I wanted onions. :(",
                        "from":
                        {
                            "username": "melisaarreguin",
                            "first_name": "Melisa",
                            "last_name": "Arreguin",
                            "type": "user",
                            "id": "122725"
                        },
                        "id": "495675"
                    },
                    {
                        "created_time": "1287603871",
                        "text": "You got onions. You had the al pastor, remember!?",
                        "from":
                        {
                            "username": "darodriguez",
                            "first_name": "David",
                            "last_name": "Rodriguez",
                            "type": "user",
                            "id": "113603"
                        },
                        "id": "507172"
                    },
                    {
                        "created_time": "1287606841",
                        "text": "Oh shiiiit this would have taken everything to the next level",
                        "from":
                        {
                            "username": "jorstaff",
                            "first_name": "Jordan",
                            "last_name": "Walker",
                            "type": "user",
                            "id": "154425"
                        },
                        "id": "508433"
                    }]
            },
            "caption":
            {
                "created_time": "1287585720",
                "text": "Carne Asasa Torta. No Onions. ",
                "from":
                {
                    "username": "darodriguez",
                    "first_name": "David",
                    "last_name": "Rodriguez",
                    "type": "user",
                    "id": "113603"
                },
                "id": "495555"
```

```
            },
            "link": "http://localhost:8000/p/C5cU/",
            "likes":
            {
                "count": 1
            },
            "created_time": "1287585671",
            "images":
            {
                "low_resolution":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/20/ad6e8429c40c
                    "width": 480,
                    "height": 480
                },
                "thumbnail":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/20/ad6e8429c40c
                    "width": 150,
                    "height": 150
                },
                "standard_resolution":
                {
                    "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/20/ad6e8429c40c
                    "width": 612,
                    "height": 612
                }
            },
            "user_has_liked": false,
            "id": "759572",
            "user":
            {
                "username": "darodriguez",
                "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_113603_7
                "id": "113603"
            }
        },
        {
            "type": "image",
            "tags": [],
            "location":
            {
                "latitude": 37.775180799999987,
                "id": "68841",
                "longitude": -122.2270716,
                "name": "El Novillo Taco Truck"
            },
            "comments":
            {
                "count": 1,
                "data": [
                    {
                        "created_time": "1287585453",
                        "text": "Tortas. ",
                        "from":
                        {
                            "username": "darodriguez",
                            "first_name": "David",
```

```
                                "last_name": "Rodriguez",
                                "type": "user",
                                "id": "113603"
                            },
                            "id": "495311"
                        }]
                },
                "caption":
                {
                    "created_time": "1287585453",
                    "text": "Tortas. ",
                    "from":
                    {
                        "username": "darodriguez",
                        "first_name": "David",
                        "last_name": "Rodriguez",
                        "type": "user",
                        "id": "113603"
                    },
                    "id": "495311"
                },
                "link": "http://localhost:8000/p/C5Wr/",
                "likes":
                {
                    "count": 0
                },
                "created_time": "1287585407",
                "images":
                {
                    "low_resolution":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/20/1fde15405b63
                        "width": 480,
                        "height": 480
                    },
                    "thumbnail":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/20/1fde15405b63
                        "width": 150,
                        "height": 150
                    },
                    "standard_resolution":
                    {
                        "url": "http://distillery-dev.s3.amazonaws.com/media/2010/10/20/1fde15405b63
                        "width": 612,
                        "height": 612
                    }
                },
                "user_has_liked": false,
                "id": "759211",
                "user":
                {
                    "username": "darodriguez",
                    "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_113603_7
                    "id": "113603"
                }
            }]
    }
```

1.3.4 Comments

get (*$mediaId*)
Get a list of recent comments on a media object.

Parameters

- **$mediaId** (*int*) – The ID of the media object

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->comments()->get(420);
echo json_encode($response->get());
```

Example response:

```
{
  "data": [
    {
      "created_time": "1280780324",
      "text": "Really amazing photo!",
      "from":
      {
        "username": "snoopdogg",
        "profile_picture": "http://images.instagram.com/profiles/profile_16_75sq_1305612434.jpg"
        "id": "1574083",
        "full_name": "Snoop Dogg"
      },
      "id": "420"
    }]
}
```

create (*$mediaId*, *$text*)
Create a comment on a media object using the following rules:

•The total length of the comment cannot exceed 300 characters.

•The comment cannot contain more than 4 hashtags.

•The comment cannot contain more than 1 URL.

•The comment cannot consist of all capital letters.

Parameters

- **$mediaId** –
- **$text** (*string*) – Text to post as a comment on the media object as specified by *$mediaId*

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->comments()->create(315, 'A comment');
echo json_encode($response->get());
```

Example response:

```
{
  "meta":
  {
    "code": 200
  },
  "data": null
}
```

delete (*$mediaId*, *$commentId*)

Remove a comment either on the owner of the access token's media object or authored by the owner of the access token.

Parameters

- **$mediaId** –
- **$commentId** (*string*) – The ID of the comment

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->comments()->delete(315, 1);
echo json_encode($response->get());
```

Example response:

```
{
  "meta":
  {
    "code": 200
  },
  "data": null
}
```

1.3.5 Likes

get (*$mediaId*)

Get a list of likes on a media object.

Parameters

- **$mediaId** (*int*) – The ID of the media object

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->likes()->get(420);
echo json_encode($response->get());
```

Example response:

```
{
  "meta":
  {
    "code": 200
  },
  "data": [
```

```
        {
          "username": "jec",
          "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_16_75sq_1288978412
          "id": "16"
        },
        {
          "username": "mikeyk",
          "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_4_75sq_1292324747_
          "id": "4"
        },
        {
          "username": "nicole",
          "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_6_75sq_1285365377.
          "id": "6"
        }]
    }
```

like (*$mediaId*)
Set a like on a media object by the currently authenticated user.

Parameters

- **$mediaId** (*int*) – The ID of the media object

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->likes()->like(315);
echo json_encode($response->get());
```

Example response:

```
{
  "meta":
  {
    "code": 200
  },
  "data": null
}
```

unlike (*$mediaId*)
Remove a like on a media object by the currently authenticated user.

Parameters

- **$mediaId** (*int*) – The ID of the media object

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->likes()->unlike(315);
echo json_encode($response->get());
```

Example response:

```
{
  "meta":
  {
```

```
        "code": 200
      },
      "data": null
    }
```

1.3.6 Tags

get (*$tag*)

Get information about a tag object.

Parameters

- **$tag** (*string*) – Name of the tag

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->tags()->get('nofilter');
echo json_encode($response->get());
```

Example response:

```
{
  "data":
  {
    "media_count": 472,
    "name": "nofilter"
  }
}
```

getRecentMedia (*$tag*, *$count = null*, *$minTagId = null*, *$maxTagId = null*)

Get a list of recently tagged media.

Parameters

- **$tag** –
- **$count** –
- **$minTagId** (*string|null*) – Return media before this min_tag_id
- **$maxTagId** (*string|null*) – Return media after this max_tag_id

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->tags()->getRecentMedia('snowy');
echo json_encode($response->get());
```

Example response:

```
{
  "data": [
    {
      "type": "image",
      "users_in_photo": [],
      "filter": "Earlybird",
```

```
            "tags": ["snow"],
            "comments":
            {
              "count": 3
            },
            "caption":
            {
              "created_time": "1296703540",
              "text": "#Snow",
              "from":
              {
                "username": "emohatch",
                "id": "1242695"
              },
              "id": "26589964"
            },
            "likes":
            {
              "count": 1
            },
            "link": "http://instagr.am/p/BWl6P/",
            "user":
            {
              "username": "emohatch",
              "profile_picture": "http://distillery.s3.amazonaws.com/profiles/profile_1242695_75sq_129
              "id": "1242695",
              "full_name": "Dave"
            },
            "created_time": "1296703536",
            "images":
            {
              "low_resolution":
              {
                "url": "http://distillery.s3.amazonaws.com/media/2011/02/02/f9443f3443484c40b4792fa7c7
                "width": 306,
                "height": 306
              },
              "thumbnail":
              {
                "url": "http://distillery.s3.amazonaws.com/media/2011/02/02/f9443f3443484c40b4792fa7c7
                "width": 150,
                "height": 150
              },
              "standard_resolution":
              {
                "url": "http://distillery.s3.amazonaws.com/media/2011/02/02/f9443f3443484c40b4792fa7c7
                "width": 612,
                "height": 612
              }
            },
            "id": "22699663",
            "location": null
          },
          {
            "type": "video",
            "videos":
            {
              "low_resolution":
```

```
            {
              "url": "http://distilleryvesper9-13.ak.instagram.com/090d06dad9cd11e2aa0912313817975d_
              "width": 480,
              "height": 480
            },
            "standard_resolution":
            {
              "url": "http://distilleryvesper9-13.ak.instagram.com/090d06dad9cd11e2aa0912313817975d_
              "width": 640,
              "height": 640
            },
            "users_in_photo": null,
            "filter": "Vesper",
            "tags": ["snow"],
            "comments":
            {
              "count": 2
            },
            "caption":
            {
              "created_time": "1296703540",
              "text": "#Snow",
              "from":
              {
                "username": "emohatch",
                "id": "1242695"
              },
              "id": "26589964"
            },
            "likes":
            {
              "count": 1
            },
            "link": "http://instagr.am/p/D/",
            "user":
            {
              "username": "kevin",
              "full_name": "Kevin S",
              "profile_picture": "...",
              "id": "3"
            },
            "created_time": "1279340983",
            "images":
            {
              "low_resolution":
              {
                "url": "http://distilleryimage2.ak.instagram.com/11f75f1cd9cc11e2a0fd22000aa8039a_6.
                "width": 306,
                "height": 306
              },
              "thumbnail":
              {
                "url": "http://distilleryimage2.ak.instagram.com/11f75f1cd9cc11e2a0fd22000aa8039a_5.
                "width": 150,
                "height": 150
              },
              "standard_resolution":
              {
```

```
                "url": "http://distilleryimage2.ak.instagram.com/11f75f1cd9cc11e2a0fd22000aa8039a_7.
                "width": 612,
                "height": 612
              }
            },
            "id": "3",
            "location": null
          }
        }]
    }
```

search (*$tag*)
Search for tags by name.

Parameters

- **$tag** (*string*) – Name of the tag

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->tags()->search('snowy');
echo json_encode($response->get());
```

Example response:

```
{
  "data": [
    {
      "media_count": 43590,
      "name": "snowy"
    },
    {
      "media_count": 3264,
      "name": "snowyday"
    },
    {
      "media_count": 1880,
      "name": "snowymountains"
    }]
}
```

1.3.7 Locations

get (*$id*)
Get information about a location.

Parameters

- **$id** (*string*) – The ID of the location

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->locations()->get('1');
echo json_encode($response->get());
```

Example response:

```
{
  "data":
  {
    "id": "1",
    "name": "Dogpatch Labs",
    "latitude": 37.782,
    "longitude": -122.387
  }
}
```

getRecentMedia (*$id*, *$minId = null*, *$maxId = null*)

Get a list of recent media objects from a given location.

Parameters

- **$id** –
- **$minId** (*string|null*) – Return media before this min_id
- **$maxId** (*string|null*) – Return media after this max_id

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->locations()->getRecentMedia('514276');
echo json_encode($response->get());
```

Example response:

```
{
  "data": [
    {
      "type": "image",
      "users_in_photo": [],
      "filter": "Earlybird",
      "tags": [
        "expobar"
      ],
      "comments":
      {
        "count": 0
      },
      "caption":
      {
        "created_time": "1296532028",
        "text": "@mikeyk pulls a shot on our #Expobar",
        "from":
        {
          "username": "josh",
          "full_name": "Josh Riedel",
          "type": "user",
          "id": "33"
        },
        "id": "25663923"
      },
      "likes":
      {
        "count": 35
```

```
        },
        "link": "http://instagr.am/p/BUS3X/",
        "user":
        {
          "username": "josh",
          "profile_picture": "...",
          "id": "33"
        },
        "created_time": "1296531955",
        "images":
        {
          "low_resolution":
          {
            "url": "http://distillery.s3.amazonaws.com/media/2011/01/31/32d364527512437a8a17ba308a
            "width": 306,
            "height": 306
          },
          "thumbnail":
          {
            "url": "http://distillery.s3.amazonaws.com/media/2011/01/31/32d364527512437a8a17ba308a
            "width": 150,
            "height": 150
          },
          "standard_resolution":
          {
            "url": "http://distillery.s3.amazonaws.com/media/2011/01/31/32d364527512437a8a17ba308a
            "width": 612,
            "height": 612
          }
        },
        "user_has_liked": false,
        "id": "22097367",
        "location":
        {
          "latitude": 37.780885099999999,
          "id": "514276",
          "longitude": -122.3948632,
          "name": "Instagram"
        }
      },
      {
        "type": "video",
        "videos":
        {
          "low_resolution":
          {
            "url": "http://distilleryvesper9-13.ak.instagram.com/090d06dad9cd11e2aa0912313817975d_
            "width": 480,
            "height": 480
          },
          "standard_resolution":
          {
            "url": "http://distilleryvesper9-13.ak.instagram.com/090d06dad9cd11e2aa0912313817975d_
            "width": 640,
            "height": 640
          },
          "users_in_photo": null,
          "filter": "Vesper",
```

```
            "tags": [],
            "comments":
            {
              "count": 2
            },
            "caption": null,
            "likes":
            {
              "count": 1
            },
            "link": "http://instagr.am/p/D/",
            "user":
            {
              "username": "kevin",
              "full_name": "Kevin S",
              "profile_picture": "...",
              "id": "3"
            },
            "created_time": "1279340983",
            "images":
            {
              "low_resolution":
              {
                "url": "http://distilleryimage2.ak.instagram.com/11f75f1cd9cc11e2a0fd22000aa8039a_6.
                "width": 306,
                "height": 306
              },
              "thumbnail":
              {
                "url": "http://distilleryimage2.ak.instagram.com/11f75f1cd9cc11e2a0fd22000aa8039a_5.
                "width": 150,
                "height": 150
              },
              "standard_resolution":
              {
                "url": "http://distilleryimage2.ak.instagram.com/11f75f1cd9cc11e2a0fd22000aa8039a_7.
                "width": 612,
                "height": 612
              }
            },
            "id": "3",
            "location":
            {
              "latitude": 37.780885099999999,
              "id": "514276",
              "longitude": -122.3948632,
              "name": "Instagram"
            }
          }
        }]
    }
```


search (*$latitude*, *$longitude*, *$distance = 1000*)

Search for a location by geographic coordinate.

Parameters

- **$latitude** (*int*) – Latitude of the center search coordinate. If used, `$longitude` is required

- **$longitude** (*int*) – Longitude of the center search coordinate. If used, `$latitude` is required
- **$distance** (*int*) – The distance in metres. Default is `1000` m, max distance is 5km

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->locations()->search(48.858325999999998, 2.294505);
echo json_encode($response->get());
```

Example response:

```
{
  "data": [
    {
      "id": "788029",
      "latitude": 48.858844300000001,
      "longitude": 2.2943506,
      "name": "Eiffel Tower, Paris"
    },
    {
      "id": "545331",
      "latitude": 48.858334059662262,
      "longitude": 2.2943401336669909,
      "name": "Restaurant 58 Tour Eiffel"
    },
    {
      "id": "421930",
      "latitude": 48.858325999999998,
      "longitude": 2.294505,
      "name": "American Library in Paris"
    }]
}
```

searchByFacebookPlacesId(*$facebookPlacesId*)

Search for a location by Facebook Places ID.

Parameters

- **$facebookPlacesId** (*int*) – A Facebook Places ID

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->locations()->searchByFacebookPlacesId(114226462057675);
echo json_encode($response->get());
```

Example response:

```
{
  "meta":
  {
    "code": 200
  },
  "data": [
    {
      "id": "788029",
```

```
            "latitude": 48.858844300000001,
            "longitude": 2.2943506,
            "name": "Eiffel Tower, Paris"
        }]
    }
```

searchByFoursquareId (*$foursquareId*)
Search for a location by Foursquare location ID.

Parameters

- **$foursquareId** (*string*) – A Foursquare V2 API location ID

Returns Response

Example request:

```
$client   = new Client($clientId, $clientSecret, $accessToken, $redirectUrl);
$response = $client->locations()->searchByFoursquareId('51a2445e5019c80b56934c75');
echo json_encode($response->get());
```

Example response:

```
{
  "meta":
  {
    "code": 200
  },
  "data": [
    {
      "id": "788029",
      "latitude": 48.858844300000001,
      "longitude": 2.2943506,
      "name": "Eiffel Tower, Paris"
    }]
}
```

1.4 FAQ

1.4.1 Why is there no way to access the *users/feed* endpoint?

Since the endpoint has been deprecated, we've chosen not to implement it in this package.

1.4.2 Why can't I use a Foursquare V1 ID to search for locations?

Foursquare has been providing new V2 IDs for places for some time now, we do not plan to support them as a native method. You can still make the request manually:

1.5 Developers

1.5.1 Contributing

Contributions are **welcome** and will be fully **credited**.

We accept contributions via Pull Requests on Github.

Documentation

Documentation for this project is available at the ReadTheDocs page.

The API reference documentation is available here.

Guidelines

- **PSR-2 Coding Standard:** The easiest way to apply the code standard is to install PHP Code Sniffer.
- **PHP 5.5.9:** Elogram has a minimum PHP version requirement of PHP 5.5.9. Pull requests must not require a PHP version greater than PHP 5.5.9 unless the feature is only utilized conditionally.
- **Add tests!:** All pull requests must include unit tests to ensure the change works as expected and to prevent regressions.
- **Documentation format:** All documentation **except** this file, `CHANGELOG.md`, `LICENSE.md` and `README.md` are in reStructuredText - this includes any code docblocks. Using reST allows us to generate a better experience for users browsing the documentation.
- **Document any change in behaviour:** Make sure the `README.md` and any other relevant documentation are kept up-to-date.
- **Consider our release cycle:** We try to follow SemVer v2. Randomly breaking public APIs is not an option.
- **Use Git Flow:** Don't ask us to pull from your master branch. Set up Git Flow and create a new feature branch from `develop`.
- **One pull request per feature:** If you want to do more than one thing, send multiple pull requests.
- **Send coherent history:** Make sure each individual commit in your pull request is meaningful. If you had to make multiple intermediate commits while developing, please squash them before submitting.

Running tests

In order to contribute, you'll need to checkout the source from GitHub and install dependencies using Composer:

```
$ git clone https://github.com/larabros/elogram.git
$ cd elogram && composer install
$ php vendor/bin/phpunit
```

Reporting a security vulnerability

We want to ensure that Elogram is secure for everyone. If you've discovered a security vulnerability, we appreciate your help in disclosing it to us in a responsible manner.

Publicly disclosing a vulnerability can put the entire community at risk. If you've discovered a security concern, please email us at contact@hassankhan.me. We'll work with you to make sure that we understand the scope of the issue, and that we fully address your concern. We consider correspondence sent to this email address our highest priority, and work to address any issues that arise as quickly as possible.

After a security vulnerability has been corrected, a security hotfix release will be deployed as soon as possible.

Happy coding!

1.5.2 Extending core components

NativeSessionStore

When using Elogram in a framework, you may want to replace the built-in session storage handler with a custom class that interacts with the framework. This class **must** implement the *DataStoreInterface*. An example is provided below:

```
use Elogram\Http\Sessions\DataStoreInterface;

class FrameworkSessionStorageHandler implements DataStorageInterface
{
    public function get($key)
    {
        // @TODO: Implement
    }

    public function set($key, $value)
    {
        // @TODO: Implement
    }
}
```

After creating this class, the *Instagram* class must be instantiated with an array of options:

```
use Larabros\Elogram\Client;

$options = ['session_store' => FrameworkSessionStorageHandler::class];
$client  = new Client($clientId, $clientSecret, $accessToken, $redirectUrl, $options);
```

1.6 API Reference

1.6.1 Larabros\Elogram\Client

class `Client`

Elogram client class.

constant `API_VERSION`

The current version of the API.

property `container`

protected ContainerInterface

The application IoC container.

`__construct` (*$clientId*, *$clientSecret*, *$accessToken = null*, *$redirectUrl = ''*, *$options =* [ ])

Create an instance of *`Client`*.

Parameters

- **`$clientId`** –
- **`$clientSecret`** –
- **`$accessToken`** –
- **`$redirectUrl`** –
- **`$options`** –

buildContainer(*$options*)
Takes the constructor parameters and uses them to instantiate and build a `Container` object.

Parameters

- **$options** –

Returns ContainerInterface

users()
Returns the current instance of *UsersRepository*.

Returns UsersRepository

media()
Returns the current instance of *MediaRepository*.

Returns MediaRepository

comments()
Returns the current instance of *CommentsRepository*.

Returns CommentsRepository

likes()
Returns the current instance of *LikesRepository*.

Returns LikesRepository

tags()
Returns the current instance of *TagsRepository*.

Returns TagsRepository

locations()
Returns the current instance of *LocationsRepository*.

Returns LocationsRepository

request(*$method*, *$uri*, *$parameters* =[ ])
Sends a request.

Parameters

- **$method** (*string*) –
- **$uri** (*string*) –
- **$parameters** –

Returns Response

paginate(*Response $response*, *$limit = null*)
Paginates a *Response*.

Parameters

- **$response** (Response) –
- **$limit** –

Returns Response

getLoginUrl(*$options* =[ ])
Gets the login URL.

Parameters

- **$options** (*array*) –

Returns string

getAccessToken (*$code*, *$grant = 'authorization_code'*)
Sets and returns the access token.

Parameters

- **$code** (*string*) –
- **$grant** (*string*) –

Returns AccessToken

setAccessToken (*AccessToken $token*)
Sets an access token and adds it to *AuthMiddleware* so the application can make authenticated requests.

Parameters

- **$token** (*AccessToken*) –

Returns unknown void

secureRequests (*$enable = true*)
Enables or disables secure requests by adding or removing *SecureRequestMiddleware*.

Parameters

- **$enable** (*bool*) –

Returns unknown void

1.6.2 Larabros\Elogram\Config

class Config
Stores the application configuration.

__construct (*$data*)
Constructor method and sets default options, if any

Parameters

- **$data** (*array*) –

getDefaults ()
{@inheritDoc}

1.6.3 Larabros\Elogram\Container

Larabros\Elogram\Container\Builder

class Builder
Builds `Container` objects for use by the application.

property defaultProviders
protected array

Default application service providers.

__construct (*$config*)
Creates a new instance of *Builder*.

Parameters

- **$config** (*array*) –

createConfig (*$config*)
Creates a *Config* object from raw parameters.

Parameters

- **$config** (*array*) –

Returns Config

registerProviders (*$providers* = [ ])
Register default service providers onto the container.

Parameters

- **$providers** (*array*) –

Returns Builder

registerProvider (*$provider*)
Registers a service provider onto the container.

Parameters

- **$provider** (*string|ServiceProviderInterface*) –

Returns Builder

createContainer (*$config*)
Creates and returns a new instance of `Container` after adding `$config` to it.

Parameters

- **$config** (*array*) –

Returns ContainerInterface

1.6.4 Larabros\Elogram\Exceptions

Larabros\Elogram\Exceptions\APIInvalidParametersError

class **APIInvalidParametersError**
Thrown if a request has invalid or missing parameters.

property message
protected

property code
protected

property file
protected

property line
protected

__clone ()

__construct (*$message*, *$code*, *$previous*)

Parameters

- **$message** –
- **$code** –
- **$previous** –

__wakeup()

getMessage()

getCode()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Larabros\Elogram\Exceptions\APINotAllowedError

class APINotAllowedError

Thrown when an API endpoint is not allowed.

property message
protected

property code
protected

property file
protected

property line
protected

__clone()

__construct(*$message*, *$code*, *$previous*)

Parameters

- **$message** –
- **$code** –
- **$previous** –

__wakeup()

getMessage()

getCode()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Larabros\Elogram\Exceptions\APINotFoundError

class APINotFoundError
Thrown when an API endpoint is not found.

property message
protected

property code
protected

property file
protected

property line
protected

__clone()

__construct(*$message*, *$code*, *$previous*)

Parameters

- **$message** –
- **$code** –
- **$previous** –

__wakeup()

getMessage()

getCode()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Larabros\Elogram\Exceptions\CsrfException

class CsrfException
Thrown if the CSRF verification failed.

property message
protected

property code
protected

property file
protected

property `line`
protected

`__clone`()

`__construct`(*$message*, *$code*, *$previous*)

Parameters

- `$message` –
- `$code` –
- `$previous` –

`__wakeup`()

`getMessage`()

`getCode`()

`getFile`()

`getLine`()

`getTrace`()

`getPrevious`()

`getTraceAsString`()

`__toString`()

Larabros\Elogram\Exceptions\Exception

class `Exception`
Base exception class for this package.

property `message`
protected

property `code`
protected

property `file`
protected

property `line`
protected

`__clone`()

`__construct`(*$message*, *$code*, *$previous*)

Parameters

- `$message` –
- `$code` –
- `$previous` –

`__wakeup`()

`getMessage`()

`getCode`()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Larabros\Elogram\Exceptions\IncompatibleResponseException

class IncompatibleResponseException
Thrown if the response contents cannot be merged.

property message
protected

property code
protected

property file
protected

property line
protected

__clone()

__construct(*$message*, *$code*, *$previous*)

Parameters

- **$message** –
- **$code** –
- **$previous** –

__wakeup()

getMessage()

getCode()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Larabros\Elogram\Exceptions\OAuthAccessTokenException

class OAuthAccessTokenException
Thrown when an access token is invalid.

property `message`
protected

property `code`
protected

property `file`
protected

property `line`
protected

`__clone`()

`__construct`(*$message*, *$code*, *$previous*)

Parameters

- `$message` –
- `$code` –
- `$previous` –

`__wakeup`()

`getMessage`()

`getCode`()

`getFile`()

`getLine`()

`getTrace`()

`getPrevious`()

`getTraceAsString`()

`__toString`()

Larabros\Elogram\Exceptions\OAuthException

class `OAuthException`
OAuthException

property `message`
protected

property `code`
protected

property `file`
protected

property `line`
protected

`__clone`()

`__construct`(*$message*, *$code*, *$previous*)

Parameters

- `$message` –

- **$code** –
- **$previous** –

__wakeup()

getMessage()

getCode()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Larabros\Elogram\Exceptions\OAuthForbiddenException

class OAuthForbiddenException

Thrown when a secure request fails.

property message
protected

property code
protected

property file
protected

property line
protected

__clone()

__construct(*$message*, *$code*, *$previous*)

Parameters

- **$message** –
- **$code** –
- **$previous** –

__wakeup()

getMessage()

getCode()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Larabros\Elogram\Exceptions\OAuthParameterException

class `OAuthParameterException`
Thrown if a required OAuth parameter was not provided in a request.

property `message`
protected

property `code`
protected

property `file`
protected

property `line`
protected

`__clone()`

`__construct` (*$message*, *$code*, *$previous*)

Parameters

- **`$message`** –
- **`$code`** –
- **`$previous`** –

`__wakeup()`

`getMessage()`

`getCode()`

`getFile()`

`getLine()`

`getTrace()`

`getPrevious()`

`getTraceAsString()`

`__toString()`

Larabros\Elogram\Exceptions\OAuthPermissionsException

class `OAuthPermissionsException`
Thrown if a request is made with an access token that is not authorized for a scope.

property `message`
protected

property `code`
protected

property `file`
protected

property `line`
protected

`__clone()`

__construct(*$message*, *$code*, *$previous*)

Parameters

- **$message** –
- **$code** –
- **$previous** –

__wakeup()

getMessage()

getCode()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

Larabros\Elogram\Exceptions\OAuthRateLimitException

class OAuthRateLimitException

Thrown when the API rate limit has been exceeded.

property message
protected

property code
protected

property file
protected

property line
protected

__clone()

__construct(*$message*, *$code*, *$previous*)

Parameters

- **$message** –
- **$code** –
- **$previous** –

__wakeup()

getMessage()

getCode()

getFile()

getLine()

getTrace()

getPrevious()

getTraceAsString()

__toString()

1.6.5 Larabros\Elogram\Helpers

Larabros\Elogram\Helpers\RedirectLoginHelper

class RedirectLoginHelper
RedirectLoginHelper

property provider
protected AdapterInterface

property store
protected DataStoreInterface

__construct(*AdapterInterface $provider*, *DataStoreInterface $store*)
Creates an instance of *RedirectLoginHelper*.

Parameters

- **$provider** (*AdapterInterface*) –
- **$store** (DataStoreInterface) –

getLoginUrl(*$options* = [ ])
Sets CSRF value and returns the login URL.

Parameters

- **$options** (*array*) –

Returns string

getAccessToken(*$code*, *$grant* = *'authorization_code'*)
Validates CSRF and returns the access token.

Parameters

- **$code** (*string*) –
- **$grant** (*string*) –

Returns AccessToken

validateCsrf()
Validates any CSRF parameters.

getInput(*$key*)
Retrieves and returns a value from a GET param.

Parameters

- **$key** (*string*) –

Returns string|null

1.6.6 Larabros\Elogram\Http

Larabros\Elogram\Http\Clients

Larabros\Elogram\Http\Clients\AbstractAdapter

class `AbstractAdapter`
An abstract HTTP client adapter.

`request` (*$method*, *$uri*, *$parameters* =[ ])
{@inheritDoc}

Parameters

- **`$method`** –
- **`$uri`** –
- **`$parameters`** –

`paginate` (*Response $response*, *$limit = null*)
{@inheritDoc}

Parameters

- **`$response`** (`Response`) –
- **`$limit`** –

`resolveExceptionClass` (*ClientException $exception*)
Parses a `ClientException` for any specific exceptions thrown by the API in the response body. If the response body is not in JSON format, an `Exception` is returned.

Check a `ClientException` to see if it has an associated `ResponseInterface` object.

Parameters

- **`$exception`** (*`ClientException`*) –

Returns Exception

Larabros\Elogram\Http\Clients\AdapterInterface

interface `AdapterInterface`
An interface for HTTP clients.

`request` (*$method*, *$uri*, *$parameters* =[ ])
Sends a HTTP request. Use this method as a convenient way of making requests with built-in exception-handling.

Parameters

- **`$method`** –
- **`$uri`** –
- **`$parameters`** –

Returns Response

`paginate` (*Response $response*, *$limit = null*)
Paginates a *`Response`*.

Parameters

- **$response** (Response) –
- **$limit** –

Returns Response

Larabros\Elogram\Http\Clients\GuzzleAdapter

class GuzzleAdapter

A HTTP client adapter for Guzzle.

property guzzle

protected ClientInterface

The Guzzle client instance.

__construct (*ClientInterface $guzzle*)

Creates a new instance of *GuzzleAdapter*.

Parameters

- **$guzzle** (*ClientInterface*) –

request (*$method*, *$uri*, *$parameters* = [ ])

{@inheritDoc}

Parameters

- **$method** –
- **$uri** –
- **$parameters** –

paginate (*Response $response*, *$limit = null*)

{@inheritDoc}

Parameters

- **$response** (Response) –
- **$limit** –

resolveExceptionClass (*ClientException $exception*)

Parses a `ClientException` for any specific exceptions thrown by the API in the response body. If the response body is not in JSON format, an `Exception` is returned.

Check a `ClientException` to see if it has an associated `ResponseInterface` object.

Parameters

- **$exception** (*ClientException*) –

Returns Exception

Larabros\Elogram\Http\Clients\MockAdapter

class MockAdapter

A mock HTTP client adapter.

property fixturesPath

protected string

__construct (*$fixturesPath*)
Creates a new instance of *MockAdapter*.

Parameters

- **$fixturesPath** (*string*) –

request (*$method*, *$uri*, *$parameters* =[ ])
{@inheritDoc}

Parameters

- **$method** –
- **$uri** –
- **$parameters** –

mapRequestToFile (*$method*, *$uri*, *$parameters*)
Parse the correct filename from the request.

Parameters

- **$method** (*string*) –
- **$uri** (*string*) –
- **$parameters** –

Returns string

cleanPath (*$uri*)
Removes any unwanted suffixes and values from a URL path.

Parameters

- **$uri** –

Returns string

mapRequestParameters (*$parameters*)
Parses any filename properties from the request parameters.

Parameters

- **$parameters** –

Returns string

paginate (*Response $response*, *$limit = null*)
{@inheritDoc}

Parameters

- **$response** (Response) –
- **$limit** –

resolveExceptionClass (*ClientException $exception*)
Parses a `ClientException` for any specific exceptions thrown by the API in the response body. If the response body is not in JSON format, an `Exception` is returned.

Check a `ClientException` to see if it has an associated `ResponseInterface` object.

Parameters

- **$exception** (*ClientException*) –

Returns Exception

Larabros\Elogram\Http\Middleware

Larabros\Elogram\Http\Middleware\AbstractMiddleware

class `AbstractMiddleware`
An abstract middleware class.

property `nextHandler`
protected callable

The next handler in the stack.

property `config`
protected ConfigInterface

The application configuration.

`__construct` (*$nextHandler*, *ConfigInterface $config*)
Creates an instance of *`AbstractMiddleware`*.

Parameters

- **`$nextHandler`** –
- **`$config`** (*`ConfigInterface`*) –

`__invoke` (*RequestInterface $request*, *$options*)
{@inheritDoc}

Parameters

- **`$request`** (*`RequestInterface`*) –
- **`$options`** –

Larabros\Elogram\Http\Middleware\AuthMiddleware

class `AuthMiddleware`
A middleware class for authenticating requests made to Instagram's API.

property `nextHandler`
protected callable

The next handler in the stack.

property `config`
protected ConfigInterface

The application configuration.

`__invoke` (*RequestInterface $request*, *$options*)
{@inheritDoc}

Parameters

- **`$request`** (*`RequestInterface`*) –
- **`$options`** –

`create` (*ConfigInterface $config*)
Factory method used to register this middleware on a handler stack.

Parameters

- **$config** (*ConfigInterface*) –

Returns Closure

__construct (*$nextHandler*, *ConfigInterface $config*)
Creates an instance of *AbstractMiddleware*.

Parameters

- **$nextHandler** –
- **$config** (*ConfigInterface*) –

Larabros\Elogram\Http\Middleware\CreateMiddlewareTrait

trait CreateMiddlewareTrait
A trait for creating callables for registering middleware on a handler stack.

create (*ConfigInterface $config*)
Factory method used to register this middleware on a handler stack.

Parameters

- **$config** (*ConfigInterface*) –

Returns Closure

Larabros\Elogram\Http\Middleware\MiddlewareInterface

interface MiddlewareInterface
An interface for PSR-7 compatible middleware.

__invoke (*RequestInterface $request*, *$options*)
Execute the middleware.

Parameters

- **$request** (*RequestInterface*) –
- **$options** (*array*) –

Returns mixed

Larabros\Elogram\Http\Middleware\SecureRequestMiddleware

class SecureRequestMiddleware
A middleware class for making secure requests to Instagram's API.

property nextHandler
protected callable

The next handler in the stack.

property config
protected ConfigInterface

The application configuration.

__invoke (*RequestInterface $request*, *$options*)
{@inheritDoc}

Parameters

- **$request** (*RequestInterface*) –
- **$options** –

generateSig(*$endpoint*, *$params*, *$secret*)
Generates a `sig` value for a request.

Parameters

- **$endpoint** –
- **$params** –
- **$secret** –

Returns string

create(*ConfigInterface $config*)
Factory method used to register this middleware on a handler stack.

Parameters

- **$config** (*ConfigInterface*) –

Returns Closure

getPath(*UriInterface $uri*)
Gets the path from a `UriInterface` instance after removing the version prefix.

Parameters

- **$uri** (*UriInterface*) –

Returns string

getQueryParams(*UriInterface $uri*, *$exclude* =[, *'sig'*], *$params* =[ ])
Gets the query parameters as an array from a `UriInterface` instance.

Parameters

- **$uri** (*UriInterface*) –
- **$exclude** (*array*) –
- **$params** (*array*) –

Returns array

__construct(*$nextHandler*, *ConfigInterface $config*)
Creates an instance of *AbstractMiddleware*.

Parameters

- **$nextHandler** –
- **$config** (*ConfigInterface*) –

Larabros\Elogram\Http\Response

class Response
Represents a response returned from the API.

property raw
protected array

property meta
protected array

property data
protected array

property pagination
protected array

__construct (*$meta* =[ ], *$data* =[ ], *$pagination* =[ ])
Creates a new instance of *Response*.

Parameters

- **$meta** (*array*) –
- **$data** (*array*) –
- **$pagination** (*array*) –

createFromJson (*$response*)
Creates a new instance of *Response* from a JSON-decoded response body.

Parameters

- **$response** (*array*) –

Returns static

getRaw (*$key = null*)
Gets the JSON-decoded raw response.

Parameters

- **$key** (*string|null*) –

Returns array

get ()
Gets the response body. If the response contains multiple records, a `Collection` is returned.

Returns array|Collection

merge (*Response $response*)
Merges the contents of this response with `$response` and returns a new *Response* instance.

Parameters

- **$response** (Response) –

Returns Response

isCollection (*$data*)
Tests the current response data to see if one or more records were returned.

Parameters

- **$data** (*array|Collection*) –

Returns bool

isRecord (*$data*)
Tests the current response data to see if a single record was returned.

Parameters

- **$data** (*array|Collection*) –

Returns bool

hasPages()
If the response has a `pagination` field with a `next_url` key, then returns `true`, otherwise `false`.

Returns bool

nextUrl()
Returns the next URL, if available, otherwise `null`.

Returns string|null

__toString()
Returns the JSON-encoded raw response.

Returns string

Larabros\Elogram\Http\Sessions

Larabros\Elogram\Http\Sessions\DataStoreInterface

interface DataStoreInterface
Defines an interface for getting and setting values from a data store.

get(*$key*)
Get a value from a data store.

Parameters

- **$key** (*string*) –

Returns mixed

set(*$key*, *$value*)
Set a value in the data store.

Parameters

- **$key** (*string*) –
- **$value** –

Returns unknown void

Larabros\Elogram\Http\Sessions\NativeSessionStore

class NativeSessionStore
An implementation of *DataStoreInterface* that uses native sessions.

property sessionPrefix
protected string

__construct()
Creates an instance of *NativeSessionStore*.

get(*$key*)
{@inheritDoc}

Parameters

- **$key** –

set (*$key*, *$value*)
{@inheritDoc}

Parameters

- **$key** –
- **$value** –

Larabros\Elogram\Http\UrlParserTrait

trait UrlParserTrait
Adds utility classes for parsing parts of a URL.

getPath (*UriInterface $uri*)
Gets the path from a `UriInterface` instance after removing the version prefix.

Parameters

- **$uri** (*UriInterface*) –

Returns string

getQueryParams (*UriInterface $uri*, *$exclude* = [, *'sig'*], *$params* = [ ])
Gets the query parameters as an array from a `UriInterface` instance.

Parameters

- **$uri** (*UriInterface*) –
- **$exclude** (*array*) –
- **$params** (*array*) –

Returns array

1.6.7 Larabros\Elogram\Providers

Larabros\Elogram\Providers\CoreServiceProvider

class CoreServiceProvider
Adds core classes to container.

property provides
protected array

The provides array is a way to let the container know that a service is provided by this service provider. Every service that is registered via this service provider must have an alias added to this array or it will be ignored.

register()
Use the register method to register items with the container via the protected `$this->container` property or the `getContainer` method from the `ContainerAwareTrait`.

Returns void

Larabros\Elogram\Providers\EntityServiceProvider

class `EntityServiceProvider`
Adds repository classes to the container.

property `provides`
protected array

The provides array is a way to let the container know that a service is provided by this service provider. Every service that is registered via this service provider must have an alias added to this array or it will be ignored.

`register()`
Use the register method to register items with the container via the protected `$this->container` property or the `getContainer` method from the `ContainerAwareTrait`.

Returns void

Larabros\Elogram\Providers\GuzzleServiceProvider

class `GuzzleServiceProvider`
Adds Guzzle to the project.

property `provides`
protected array

The provides array is a way to let the container know that a service is provided by this service provider. Every service that is registered via this service provider must have an alias added to this array or it will be ignored.

`register()`
Use the register method to register items with the container via the protected `$this->container` property or the `getContainer` method from the `ContainerAwareTrait`.

Returns void

Larabros\Elogram\Providers\MiddlewareServiceProvider

class `MiddlewareServiceProvider`
Adds any middleware to the project.

property `provides`
protected array

The provides array is a way to let the container know that a service is provided by this service provider. Every service that is registered via this service provider must have an alias added to this array or it will be ignored.

`register()`
Use the register method to register items with the container via the protected `$this->container` property or the `getContainer` method from the `ContainerAwareTrait`.

Returns void

`addMiddleware()`

1.6.8 Larabros\Elogram\Repositories

Larabros\Elogram\Repositories\AbstractRepository

class `AbstractRepository`
An abstract repository class. Any new endpoints should extend this class.

property `client`
protected AdapterInterface

`__construct` (*AdapterInterface $client*)
Creates a new instance of *AbstractRepository*.

Parameters

- **`$client`** (*AdapterInterface*) –

Larabros\Elogram\Repositories\CommentsRepository

class `CommentsRepository`
CommentsRepository

property `client`
protected AdapterInterface

`get` (*$mediaId*)
Get a list of recent comments on a media object.

Parameters

- **`$mediaId`** (*int*) – The ID of the media object

Returns Response

`create` (*$mediaId*, *$text*)
Create a comment on a media object using the following rules:

•The total length of the comment cannot exceed 300 characters.

•The comment cannot contain more than 4 hashtags.

•The comment cannot contain more than 1 URL.

•The comment cannot consist of all capital letters.

Parameters

- **`$mediaId`** –
- **`$text`** (*string*) – Text to post as a comment on the media object

Returns Response

`delete` (*$mediaId*, *$commentId*)
Remove a comment either on the owner of the access token's media object or authored by the owner of the access token.

Parameters

- **`$mediaId`** –
- **`$commentId`** (*string*) – The ID of the comment

Returns Response

__construct (*AdapterInterface $client*)
Creates a new instance of *AbstractRepository*.

Parameters

- **$client** (*AdapterInterface*) –

Larabros\Elogram\Repositories\LikesRepository

class LikesRepository
LikesRepository class.

property client
protected AdapterInterface

get (*$mediaId*)
Get a list of likes on a media object.

Parameters

- **$mediaId** (*int*) – The ID of the media object

Returns Response

like (*$mediaId*)
Set a like on a media object by the currently authenticated user.

Parameters

- **$mediaId** (*int*) – The ID of the media object

Returns Response

unlike (*$mediaId*)
Remove a like on a media object by the currently authenticated user.

Parameters

- **$mediaId** (*int*) – The ID of the media object

Returns Response

__construct (*AdapterInterface $client*)
Creates a new instance of *AbstractRepository*.

Parameters

- **$client** (*AdapterInterface*) –

Larabros\Elogram\Repositories\LocationsRepository

class LocationsRepository
LocationsRepository

property client
protected AdapterInterface

get (*$id*)
Get information about a location.

Parameters

- **$id** (*string*) – The ID of the location

Returns Response

getRecentMedia (*$id*, *$minId = null*, *$maxId = null*)
Get a list of recent media objects from a given location.

Parameters

- **$id** –
- **$minId** (*string|null*) – Return media before this min_id
- **$maxId** (*string|null*) – Return media after this max_id

Returns Response

search (*$latitude*, *$longitude*, *$distance = 1000*)
Search for a location by geographic coordinate.

Parameters

- **$latitude** (*int*) – Latitude of the center search coordinate. If used, `$longitude` is required
- **$longitude** (*int*) – Longitude of the center search coordinate. If used, `$latitude` is required
- **$distance** (*int*) – The distance in metres. Default is ‘‘1000‘‘m, max distance is 5km

Returns Response

searchByFacebookPlacesId (*$facebookPlacesId*)
Search for a location by Facebook Places ID.

Parameters

- **$facebookPlacesId** (*int*) – A Facebook Places ID

Returns Response

searchByFoursquareId (*$foursquareId*)
Search for a location by Foursquare location ID.

Parameters

- **$foursquareId** (*string*) – A Foursquare V2 API location ID

Returns Response

__construct (*AdapterInterface $client*)
Creates a new instance of *AbstractRepository*.

Parameters

- **$client** (*AdapterInterface*) –

Larabros\Elogram\Repositories\MediaRepository

class MediaRepository
MediaRepository

property client
protected AdapterInterface

get (*$id*)
Get information about a media object.

Parameters

- **$id** (*string*) – The ID of the media object

Returns Response

getByShortcode (*$shortcode*)
This method returns the same response as *MediaRepository::get*

Parameters

- **$shortcode** (*string*) – The shortcode of the media object

Returns Response

search (*$latitude*, *$longitude*, *$distance = 1000*)
Search for recent media in a given area.

Parameters

- **$latitude** (*int*) – Latitude of the center search coordinate. If used, `$longitude` is required
- **$longitude** (*int*) – Longitude of the center search coordinate. If used, `$latitude` is required
- **$distance** (*int*) – The distance in metres. Default is ‘‘1000‘‘m, max distance is 5km.

Returns Response

__construct (*AdapterInterface $client*)
Creates a new instance of *AbstractRepository*.

Parameters

- **$client** (*AdapterInterface*) –

Larabros\Elogram\Repositories\TagsRepository

class TagsRepository
TagsRepository

property client
protected AdapterInterface

get (*$tag*)
Get information about a tag object.

Parameters

- **$tag** (*string*) – Name of the tag

Returns Response

getRecentMedia (*$tag*, *$count = null*, *$minTagId = null*, *$maxTagId = null*)
Get a list of recently tagged media.

Parameters

- **$tag** –
- **$count** –

- **$minTagId** (*string|null*) – Return media before this min_tag_id
- **$maxTagId** (*string|null*) – Return media after this max_tag_id

Returns Response

search (*$tag*)
Search for tags by name.

Parameters

- **$tag** (*string*) – Name of the tag

Returns Response

__construct (*AdapterInterface $client*)
Creates a new instance of *AbstractRepository*.

Parameters

- **$client** (*AdapterInterface*) –

Larabros\Elogram\Repositories\UsersRepository

class UsersRepository
UsersRepository

property client
protected AdapterInterface

get (*$id = 'self'*)
Get information about a user.

Parameters

- **$id** (*string*) – The ID of the user. Default is `self`

Returns Response

getMedia (*$id = 'self'*, *$count = null*, *$minId = null*, *$maxId = null*)
Get the most recent media published by a user.

Parameters

- **$id** –
- **$count** (*int|null*) – Count of media to return
- **$minId** (*int|null*) – Return media later than this min_id
- **$maxId** (*int|null*) – Return media earlier than this max_id

Returns Response

getLikedMedia (*$count = null*, *$maxLikeId = null*)
Get the list of recent media liked by the owner of the access token.

Parameters

- **$count** (*int|null*) – Count of media to return
- **$maxLikeId** (*int|null*) – Return media liked before this id

Returns Response

search (*$query*, *$count = null*)
Get a list of users matching the query.

Parameters

- **$query** –
- **$count** –

Returns Response

find(*$username*)
Searches for and returns a single user's information. If no results are found, `null` is returned.

Parameters

- **$username** (*string*) – A username to search for

Returns Response|null

follows()
Get the list of users this user follows.

Returns Response

followedBy()
Get the list of users this user is followed by.

Returns Response

requestedBy()
List the users who have requested this user's permission to follow.

Returns Response

getRelationship(*$targetUserId*)
Get information about the relationship of the owner of the access token to another user.

Parameters

- **$targetUserId** (*string*) – The ID of the target user

Returns Response

setRelationship(*$targetUserId*, *$action*)
Modify the relationship between the owner of the access token and the target user.

Parameters

- **$targetUserId** (*string*) – The ID of the target user
- **$action** (*string*) – Can be one of: `follow | unfollow | approve | ignore`

Returns Response

__construct(*AdapterInterface $client*)
Creates a new instance of *AbstractRepository*.

Parameters

- **$client** (*AdapterInterface*) –