aiomailru

Release 0.1.1.post1

May 07, 2020

Contents:

aiomailru is a python Mail.Ru API wrapper. The main features are:

- authorization (Authorization Code, Implicit Flow, Password Grant, Refresh Token)
- REST API methods
- web scrapers

CHAPTER 1

Usage

To use Mail.Ru API you need a registered app and Mail.Ru account. For more details, see aiomailru Documentation.

1.1 Client application

Use `ClientSession` when REST API is needed in:

- a client component of the client-server application
- a standalone mobile/desktop application

i.e. when you embed your app's info (private key) in publicly available code.

```
from aiomailru import ClientSession, API

session = ClientSession(app_id, private_key, access_token, uid)
api = API(session)

events = await api.stream.get()
friends = await api.friends.getOnline()
```

Use `access_token` and `uid` that were received after authorization. For more details, see authorization instruction.

1.2 Server application

Use `ServerSession` when REST API is needed in:

- a server component of the client-server application
- requests from your servers

```
from aiomailru import ServerSession, API

session = ServerSession(app_id, secret_key, access_token)
api = API(session)

events = await api.stream.get()
friends = await api.friends.getOnline()
```

Use `access_token` that was received after authorization. For more details, see authorization instruction.

CHAPTER 2

Installation

```
$ pip install aiomailru
```

or

```
$ python setup.py install
```

CHAPTER 3

Supported Python Versions

Python 3.5, 3.6, 3.7 and 3.8 are supported.

3.1 Getting Started

3.1.1 Installation

If you use pip, just type

```
$ pip install aiomailru
```

You can install from the source code like

```
$ git clone https://github.com/KonstantinTogoi/aiomailru.git
$ cd aiomailru
$ python setup.py install
```

3.1.2 Account

Sign up in Mail.Ru.

3.1.3 Application

After signing up visit Mail.Ru API documentation page and create a new application: https://api.mail.ru/apps/my/add.

Save **client_id** (aka **app_id**), **private_key** and **secret_key** for user authorization and executing API requests.

```
app_id = 'your_client_id'
private_key = 'your_private_key'
secret_key = 'your_secret_key'
```

3.2 Authorization

The preferred way to authorize is an `async with` statement. After authorization the session will have the following attributes:

- `session_key` aka `access_token`
- `refresh_token`
- `expires_in`
- `token_type` if Implicit Grant used
- `uid`

3.2.1 Authorization Code Grant

```
from aiomailru import CodeSession, API

app_id = 123456
private_key = ''
secret_key = 'xyz'

async with CodeSession(app_id, private_key, secret_key, code, redirect_uri) as␣
↪session:
    api = API(session)
    ...
```

About OAuth 2.0 Authorization Code Grant: https://oauth.net/2/grant-types/authorization-code/

For more details, see https://api.mail.ru/docs/guides/oauth/sites/ and https://api.mail.ru/docs/guides/oauth/mobile-web/

3.2.2 Implicit Grant

```
from aiomailru import ImplicitSession, API

app_id = 123456
private_key = 'abcde'
secret_key = ''

async with ImplicitSession(app_id, private_key, secret_key, email, passwd, scope) as␣
↪session:
    api = API(session)
    ...
```

About OAuth 2.0 Implicit Grant: https://oauth.net/2/grant-types/implicit/

For more details, see https://api.mail.ru/docs/guides/oauth/standalone/

3.2.3 Password Grant

```
from aiomailru import PasswordSession, API

app_id = 123456
private_key = 'abcde'
secret_key = ''

async with PasswordSession(app_id, private_key, secret_key, email, passwd, scope) as␣
↪session:
    api = API(session)
    ...
```

About OAuth 2.0 Password Grant: https://oauth.net/2/grant-types/password/

For more details, see https://api.mail.ru/docs/guides/oauth/client-credentials/

3.2.4 Refresh Token

```
from aiomailru import RefreshSession, API

app_id = 123456
private_key = ''
secret_key = 'xyz'

async with RefreshSession(app_id, private_key, secret_key, refresh_token) as session:
    api = API(session)
    ...
```

About OAuth 2.0 Refresh Token: https://oauth.net/2/grant-types/refresh-token/

For more details, see https://api.mail.ru/docs/guides/oauth/client-credentials/#refresh_token

3.3 Session

The session makes **GET** requests when you call instance of `APIMethod` class that are returned as attributes of an `API` class instance.

3.3.1 Request

By default, the session (`CodeSession`, `ImplicitSession`, `PasswordSession`, `RefreshSession`) tries to infer which signature generation circuit to use:

- if `uid` and `private_key` are not empty strings - **client-server** signature generation circuit is used
- else if `secret_key` is not an empty string - **server-server** signature generation circuit is used
- else exception is raised

You can explicitly set a signature generation circuit for signing requests by passing to `API` one of the sessions below.

Client-Server signature generation circuit

Let's consider the following example of API request with client-server signature:

```
from aiomailru import TokenSession, API

session = TokenSession(
    app_id=423004,
    private_key='7815696ecbf1c96e6894b779456d330e',
    secret_key='',
    access_token='be6ef89965d58e56dec21acb9b62bdaa',
    uid='1324730981306483817',
)
api = API(session)

friends = await api.friends.get()
```

It is equivalent to **GET** request:

```
https://appsmail.ru/platform/api
    ?method=friends.get
    &app_id=423004
    &session_key=be6ef89965d58e56dec21acb9b62bdaa
    &sig=5073f15c6d5b6ab2fde23ac43332b002
```

The following steps were taken:

1. request parameters were sorted and concatenated - `app_id=423004method=friends.getsession_key=be6ef89965d58e56dec21acb9b62bdaa`
2. `uid`, sorted request parameters, `private_key` were concatenated - `1324730981306483817app_id=423004method=friends.getsession_key=be6ef89965d58e56dec21acb`
3. signature `5073f15c6d5b6ab2fde23ac43332b002` calculated as MD5 of the previous string
4. signature appended to **GET** request parameters

For more details, see https://api.mail.ru/docs/guides/restapi/#client.

ClientSession

`ClientSession` is a subclass of `TokenSession`.

```
from aiomailru import ClientSession, API

session = ClientSession(app_id, 'private key', 'access token', uid)
api = API(session)
...
```

CodeClientSession

`CodeClientSession` is a subclass of `CodeSession`.

```
from aiomailru import CodeClientSession, API

async with CodeClientSession(app_id, 'private key', code, redirect_uri) as session:
    api = API(session)
    ...
```

ImplicitClientSession

`ImplicitClientSession` is a subclass of `ImplicitSession`.

```
from aiomailru import ImplicitClientSession, API

async with ImplicitClientSession(app_id, 'private key', email, passwd, scope) as␣
↪session:
    api = API(session)
    ...
```

PasswordClientSession

`PasswordClientSession` is a subclass of `PasswordSession`.

```
from aiomailru import PasswordClientSession, API

async with PasswordClientSession(app_id, 'private key', email, passwd, scope) as␣
↪session:
    api = API(session)
    ...
```

RefreshClientSession

`RefreshClientSession` is a subclass of `RefreshSession`.

```
from aiomailru import RefreshClientSession, API

async with RefreshClientSession(app_id, 'private key', refresh_token) as session:
    api = API(session)
    ...
```

Server-Server signature generation circuit

Let's consider the following example of API request with server-server signature:

```
from aiomailru import TokenSession, API

session = TokenSession(
    app_id=423004,
    private_key='',
    secret_key='3dad9cbf9baaa0360c0f2ba372d25716',
    access_token='be6ef89965d58e56dec21acb9b62bdaa',
    uid='',
)
api = API(session)

friends = await api.friends.get()
```

It is equivalent to **GET** request:

```
https://appsmail.ru/platform/api
    ?method=friends.get
    &app_id=423004
    &session_key=be6ef89965d58e56dec21acb9b62bdaa
    &sig=4a05af66f80da18b308fa7e536912bae
```

The following steps were taken:

1. parameter `secure = 1` appended to parameters
2. request parameters were sorted and concatenated - `app_id=423004method=friends.getsecure=1session_key=be6ef89965d58e56dec21acb9b62bdaa`
3. sorted request parameters and `secret_key` were concatenated - `1324730981306483817app_id=423004method=friegetsession_key=be6ef89965d58e56dec21acb9b62bdaa3dad9cbf9baaa0360c0f2ba372d25716`
4. signature `4a05af66f80da18b308fa7e536912bae` calculated as MD5 of the previous string
5. signature appended to **GET** request parameters

For more details, see https://api.mail.ru/docs/guides/restapi/#server.

ServerSession

`ServerSession` is a subclass of `TokenSession`.

```
from aiomailru import ServerSession, API

session = ServerSession(app_id, 'secret key', 'access token')
api = API(session)
...
```

CodeServerSession

`CodeServerSession` is a subclass of `CodeSession`.

```
from aiomailru import CodeServerSession, API

async with CodeServerSession(app_id, 'secret key', code, redirect_uri) as session:
    api = API(session)
    ...
```

ImplicitServerSession

`ImplicitServerSession` is a subclass of `ImplicitSession`.

```
from aiomailru import ImplicitServerSession, API

async with ImplicitServerSession(app_id, 'secret key', email, passwd, scope) as␣
↪session:
    api = API(session)
    ...
```

PasswordServerSession

`PasswordServerSession` is a subclass of `PasswordSession`.

```
from aiomailru import PasswordServerSession, API

async with PasswordServerSession(app_id, 'secret key', email, passwd, scope) as␣
↪session:
    api = API(session)
    ...
```

RefreshServerSession

`RefreshServerSession` is a subclass of `RefreshSession`.

```
from aiomailru import RefreshServerSession, API

async with RefreshServerSession(app_id, 'secret key', refresh_token) as session:
    api = API(session)
    ...
```

3.3.2 Response

By default, a session after executing request returns response's body as `dict` if executing was successful, otherwise it raises exception.

You can pass `pass_error` parameter to `TokenSession` for returning original response (including errors).

3.3.3 Error

In case of an error, by default, exception is raised. You can pass `pass_error` parameter to `TokenSession` for returning original error's body as `dict`:

```
{
    "error": {
        "error_code": 202,
        "error_msg": "Access to this object is denied"
    }
}
```

3.4 REST API

List of all methods is available here: https://api.mail.ru/docs/reference/rest/.

3.4.1 Executing requests

For executing API requests call an instance of `APIMethod` class. You can get it as an attribute of `API` class instance or as an attribute of other `APIMethod` class instance.

```
from aiomailru import API

api = API(session)

events = await api.stream.get()  # events for current user
friends = await api.friends.get()  # current user's friends
```

Under the hood each API request is enriched with parameters to generate signature:

- `method`
- `app_id`
- `session_key`
- `secure`

and with the following parameter after generating signature:

- `sig`, see https://api.mail.ru/docs/guides/restapi/#sig

3.4.2 Objects

Some objects are returned in several methods.

User

field	description
uid `string`	User ID.
first_name `string`	First name.
last_name `string`	Last name.
nick `string`	Nickname.
status_text `string`	User status.
email `string`	E-mail address.
sex `integer, [0,1]`	User sex. Possible values: - *0* - male - *1* - female
show_age `integer, [0,1]`	Information whether the user allows to show the age.
birthday `string`	User's date of birth. Returned as DD.MM.YYYY.
has_my `integer, [0,1]`	Information whether the user has profile.
has_pic `integer, [0,1]`	Information whether the user has profile photo.
pic `string`	URL of user's photo.

Continued on next page

Table 1 – continued from previous page

field	description
pic_small `string`	URL of user's photo with at most 45 pixels on the longest side.
pic_big `string`	URL of user's photo with at most 600 pixels on the longest side.
pic_22 `string`	URL of square photo of the user photo with 22 pixels in width.
pic_32 `string`	URL of square photo of the user photo with 32 pixels in width.
pic_40 `string`	URL of square photo of the user photo with 40 pixels in width.
pic_50 `string`	URL of square photo of the user photo with 50 pixels in width.
pic_128 `string`	URL of square photo of the user photo with 128 pixels in width.
pic_180 `string`	URL of square photo of the user photo with 180 pixels in width.
pic_190 `string`	URL of square photo of the user photo with 190 pixels in width.
link `string`	Returns a website address of a user profile.
referer_type `string`	Referer type. Possible values: *- stream.install - stream.publish - invitation - catalog - suggests - left menu suggest - new apps - guestbook - agent*
referer_id `string`	Identifies where a user came from; see https://api.mail.ru/docs/guides/ref/.
is_online `integer, [0,1]`	Information whether the user is online.
is_friend `integer, [0,1]`	Information whether the user is a friend of current user.
friends_count `integer`	Number of friends.
follower `integer, [0,1]`	Information whether the user is a follower of current user.
following `integer, [0,1]`	Information whether current user is a follower of the user.
subscribe `integer, [0,1]`	Information whether current user is a subscriber of the user.
subscribers_count `integer`	Number of subscribers.
video_count `integer`	Number of videos.
is_verified `integer, [0,1]`	Information whether the user is verified.
vip `integer, [0,1]`	Information whether the user is vip.

Continued on next page

Table 1 – continued from previous page

field	description
app_installed `integer, [0,1]`	Information whether the user has installed the current app.
last_visit `integer`	Date (in Unixtime) of the last user's visit.
cover `object`	Information about profile's cover; see *Cover*.
group_info `object`	Object with following fields: - **category_id** `integer` - **short_description** `string` - **full_description** `string` - **interests** `string` - **posts_cnt** `integer` - **category_name** `string` - **rules** `string`
location `object`	Object with following fields: - **country** `object`: {**id** `integer`, **name** `string`} - **city** `object`: {**id** `integer`, **name** `string`} - **region** `object`: {**id** `integer`, **name** `string`}

Event

Object describes an event and contains following fields:

field	description
thread_id `string`	Comment thread ID in the following format: `<User's checksum><ID>`.
authors `array`	Information about authors; see *User*.
type_name `string`	Event type name.
click_url `string` Returns only if current event is likeable.	Event URL.
likes_count `integer` Returns only if current event is likeable.	Number of "likes".
attachments `array`	Information about attachments to the event (link, image, video, audio, user, ...) if any; see *Attachments*.
time `integer`	Date (in Unixtime) of the event.
huid `string`	Event ID in the following format: `<User's checksum><Event ID>`.
generator `object`	Object with the following fields: - **icon** `string` - URL of app icon. - **url** `string` - App url. - **app_id** `integer` - App ID. - **type** `string` - App type. - **title** `string` - App title.
user_text `string`	User text.
is_liked_by_me `integer, [0,1]`	Shows if current user has liked the event.
subtype `string`	"event"
is_commentable `integer, [0,1]`	Shows if the event is commentable.
type `string`	Event type; see *Event types*.
is_likeable `integer, [0,1]`	Shows if the event is likeable.
id `string`	Event ID.
text_media `array` Returns only if event's type name is *micropost*.	Information about text; see *Attachments*.
comments_count `integer` Returns only if current event is commentable.	Number of comments.
action_links `array`	Each object contains following fields: - **text** `string` - **href** `string`

Event types

- 1-1 Photo
- 1-2 Video
- 1-3 Photo mark
- 1-4 Video mark
- 1-6 TYPE_PHOTO_WAS_SELECTED
- 1-7 Music
- 1-8 Photo comment
- 1-9 TYPE_PHOTO_SUBSCRIPTION
- 1-10 Video comment
- 1-11 TYPE_PHOTO_WAS_MODERATED
- 1-12 TYPE_VIDEO_WAS_MODERATED
- 1-13 TYPE_VIDEO_TRANSLATION
- 1-14 Private photo comment
- 1-15 Private video comment
- 1-16 Music comment
- 1-17 TYPE_PHOTO_NEW_COMMENT
- 1-18 TYPE_VIDEO_NEW_COMMENT
- 3-1 Blog post
- 3-2 Blog post comment
- 3-3 Join community
- 3-4 Community
- 3-5 TYPE_USER_COMMUNITY_LEAVE
- 3-6 TYPE_BLOG_COMMUNITY_POST
- 3-7 TYPE_USER_GUESTBOOK
- 3-8 TYPE_BLOG_CHALLENGE_ACCEPT
- 3-9 TYPE_BLOG_CHALLENGE_THROW
  - 3-10 TYPE_BLOG_SUBSCRIPTION
- 3-12 Blog post mark
- 3-13 Community post mark
- 3-23 Post in micro blog
- 3-25 Private post in micro blog
- 4-1 TYPE_QUESTION
- 4-2 TYPE_QUESTION_ANSWER
- 4-6 TYPE_QUESTION_ANSWER_PRIVATE
- 5-1 TYPE_USER_FRIEND

- 5-2 TYPE_USER_ANKETA
- 5-4 TYPE_USER_CLASSMATES
- 5-5 TYPE_USER_CAREER
- 5-7 TYPE_USER_AVATAR
- 5-9 TYPE_USER_PARTNER
- 5-10 TYPE_GIFT_SENT
- 5-11 TYPE_GIFT_RECEIVED
- 5-12 TYPE_USER_MILITARY
- 5-13 TYPE_USER_PARTNER_APPROVED
- 5-15 TYPE_USER_ITEM
- 5-16 App install
- 5-17 App event
- 5-18 Community post
- 5-19 Post in community guestbook
- 5-20 Join community
- 5-21 Community video
- 5-22 Community photo
- 5-24 App event
- 5-24 TYPE_APP_INFO
- 5-26 Link share
- 5-27 Event like
- 5-29 Video share
- 5-30 Comment to link share
- 5-31 Comment to video share
- 5-32 Micropost comment

Like

Object wraps an event that a user liked and contains following fields:

field	description
time `integer`	Date (in Unixtime) of the "like".
author `object`	Information about the user; see *User*.
huid `string`	Like ID in the following format: `<User's checksum><Like ID>`.
subevent `object`	Information about the event; see *Event*.
subtype `string`	"like".
is_commentable `integer, [0,1]`	0.
id `string`	Like ID.
is_likeable `integer, [0,1]`	0.

Comment

Object wraps an event that a user commented and contains following fields:

field	description
time `integer`	Date (in Unixtime) of the comment.
huid `string`	Comment ID in the following format: `<User's checksum><Comment ID>`.
subevent `object`	Information about the event; see *Event*.
subtype `string`	"comment".
comment `object`	Object with following fields: - **text** `string` - Text. - **time** `integer` - Date (in Unixtime) of the comment. - **is_deleted** `integer [0,1]` - Shows if the comment deleted. - **id** `string` - Comment ID. - **author** `object` - Information about the user; see *User*. - **text_media** `object` - Object: {**object** `string` and **content** `string`}.
is_commentable `integer, [0,1]`	0.
id `string`	Comment ID.
is_likeable `integer, [0,1]`	0.

Attachments

Information about event's media attachments is returned in field **attachments** and contains an `array` of objects. Each object contains field **object** with type name that defines all other fields.

text

contains following fields:

field
object `string, ["text"]`
content `string`

tag

contains one additional field **content** with an object with following fields:

field
is_blacklist `integer, [0,1]`
tag `string`

link

contains one additional field content with an object with following fields:

field
type-id `string, ["text"]`
contents `string`

or contains following fields:

field
object `string, ["link"]`
text `string`
url `string`

avatar

contains one additional field **new** with an object with following fields:

field
thread_id `string`
width `integer`
click_url `string`
album_id `string`
src `string`
height `integer`
desc `string`
src_hires `string`
id `string`
owner_id `string`

image

contains following fields:

field
likes_count `integer`
thread_id `string`
width `string`
object `string, ["image"]`
click_url `string`
album_id `string`
src `string`
resized_src `string`
height `string`
src_filed `string`
src_hires `string`
id `string`
owner_id `string`
comments_count `integer`

All fields but **object** and **src** may not be returned.

music

contains following fields:

field
is_add `integer`
click_url `string`
object `string, ["music"]`
name `string`
author `string`
duration `integer`
file_url `string`
uploader `string`
mid `string`

video

contains following fields:

field
width `integer`
object `string, ["video"]`
album_id `string`
view_count `integer`
desc `string`
comments_count `integer`
likes_count `integer`
thread_id `string`
image_filed `string`
click_url `string`
src `string`
duration `integer`
height `integer`
is_liked_by_me `integer`
external_id `string`
owner_id `string`
title `string`

app

contains one additional field **content** with an object with following fields:

field
PublishStatus `object` Object with following fields: - **My** `string` - **Mobile** `string`
ID `string`
InstallationsSpaced `string`
ShortName `string`
Genre `array` Each object contains following fields: - **name** `string` - **id** `string` - **admin_genre** `integer, [0,1]`
Votes `object` Object with following fields: - **VoteSum** `string` - **VotesCount** `string` - **VotesStarsWidth** `string` - **Votes2IntRounded** `string` - **Votes2DigitRounded** `string`
Installations `integer`
ShortDescription `string`
Name `string`
Description `string`
Pictures `object`

group

contains one additional field **content** with an object; see *User*.

gift

contains one additional field **content** with an object with following fields:

field
is_private `integer, [0,1]`
click_url `string`
is_anonymous `integer, [0,1]`
time `integer`
is_read `integer, [0,1]`
to `object` see *User*.
gift `object`
from `object` see *User*.
text `string`
rus_time `string`
long_id `string`

Other

Objects that are not classified.

Cover

Object contains information about profile's cover.

field
cover_position `string`
width `string`
size `string`
aid `string`
pid `string`
thread_id `string`
owner `string`
target_album `object` Information about target album; see *Target Album*.
click_url `string`
src `string`
height `string`
cover_width `string`
created `string`
comment `string`
src_small `string`
cover_height `string`
title `string`

Target Album

Object contains information about cover's target album.

field
link `string`
owner `string`
sort_order `string`
sort_by `string`
description `string`
privacy_desc `string`
size `integer`
aid `string`
created `integer`
cover_pid `string`
cover_url `string`
is_commentable `integer, [0,1]`
title `string`
updated `integer`
privacy `integer`
can_read_comment `integer, [0,1]`

3.5 Scrapers

The following scrapers are available:

- `groups.get`
- `groups.getInfo`
- `groups.join`
- `stream.getByAuthor`, works only with a group's id

```
from aiomailru.scrapers import APIScraper

api = APIScraper(session)
groups = await api.groups.get(scrape=True)  # current user's groups
```

Scrapers have the following requirements:

- Cookies
- Pyppeteer
- Browserless

3.5.1 Cookies

If `session` is instance of `TokenSession` you must set cookies that were given by `ImplicitSession`:

```
session = ServerSession(app_id, secret_key, access_token, cookies=cookies)
```

3.5.2 Pyppeteer

Scrapers require an instance of Chrome.

You can start a new Chrome process:

```
from aiomailru.scrapers import APIScraper
from pyppeteer import launch

browser = await launch()
api = APIScraper(session, browser=browser)

print(browser.wsEndpoint)  # your browser's endpoint
```

or connect to the existing Chrome:

```
from aiomailru.scrapers import APIScraper
from pyppeteer import connect

browser_conn = {'browserWSEndpoint': 'your_endpoint'}
browser = await connect(browser_conn)
api = APIScraper(session, browser=browser)
```

Export environment variable

```
$ export PYPPETEER_BROWSER_ENDPOINT='your_endpoint'
```

to automatically connect to Chrome:

```
from aiomailru.scrapers import APIScraper
api = APIScraper(session)  # connects to PYPPETEER_BROWSER_ENDPOINT
```

3.5.3 Browserless

You can replace `pyppeteer.launch` with `pyppeteer.connect`. See https://www.browserless.io

Start headless chrome using

```
$ docker-compose up -d chrome
```

Export environment variable

```
$ export PYPPETEER_BROWSER_ENDPOINT=ws://localhost:3000
```

to automatically connect to Browserless container:

```
from aiomailru.scrapers import APIScraper
api = APIScraper(session)  # connects to ws://localhost:3000
```

CHAPTER 4

Indices and tables

- genindex
- modindex