
mod.io Documentation

Release 0.6.0

Clement Julia

Jun 28, 2023

Contents:

1	Basic Usage	3
2	Getting an OAuth 2 Access Token	5
3	Example	7
4	Installation	9
5	Uninstalling	11
6	Indices and tables	59
	Python Module Index	61
	Index	63

mod.io is a python object-oriented wrapper library for the mod.io API that supports both sync and async applications. Most blocking methods have both a synchronous version and async method for use within async applications.

CHAPTER 1

Basic Usage

```
import modio

client = modio.Client(
    api_key="your api key here",
    access_token="your o auth 2 token here"
)

game = client.get_game(345)
#gets the game with id 345

print(game.name)
#prints the name of the game

mod = game.get_mod(231)
#gets the mod for that game with id 231
```


CHAPTER 2

Getting an OAuth 2 Access Token

To perform writes, you will need to authenticate your users via OAuth 2. To make this easy this library provides you with two functions to use in order to obtain your Access Token. You will need an API Key and an email address to which you have access in order for this to work. Once you have both, follow the example below, you can either run this in a REPL or as a Python script. Don't forget to edit the script to add your own api key and email address.

CHAPTER 3

Example

```
import modio

client = modio.Client(api_key="your api key here")

#request a security code be sent at this email adress
client.email_request("necro@mordor.com")

#check your email for the security code
code = input("Code: ")

oauth2 = client.email_exchange(code)
#your oauth2 token is now stored in the variable

#to save into a file simply
with open("oauth2.txt", "w") as file:
    file.write(oauth2)

#and now the token is stored in oauth2.txt
```

See more examples [here](#) .

CHAPTER 4

Installation

```
pip install mod.io
```



```
pip uninstall mod.io
```

5.1 Ratelimits and Retries

By default, when the library gets ratelimited, it will sleep for the duration required and then retry sending the request. This behavior covers most cases as the library will only sleep for about 60 seconds or less. However this is not always desirable and as such the library also provides you with the ability to decide when you want to sleep and when you want the library to raise the error through the *Client.ratelimit_max_sleep* parameter. By default this parameter is set to infinity which means that the library will always sleep the full duration. Be warned, if you are doing some heavy work using POST requests this could make you sleep for large durations like one hour. If you want finer control, you can pass an int representing the maximum number of seconds to sleep. Passing 0 will mean the library will never sleep and always raise the error.

When letting the library raise the error, you can handle the ratelimiting yourself using the *Client.retry_after* attribute to know how long you should wait before trying the request again. Some quick examples to make everything clear:

- *ratelimit_max_sleep* is 60 and you're ratelimited with *retry_after* being 60 -> library sleeps for 60 seconds
- *ratelimit_max_sleep* is 60 and you're ratelimited with *retry_after* being 3600 -> library raises the error
- *ratelimit_max_sleep* is infinity and you're ratelimited with *retry_after* being 60 -> library sleeps for 60 seconds
- *ratelimit_max_sleep* is 0 and you're ratelimited with *retry_after* being 3600 -> library raises the error
- *ratelimit_max_sleep* is 3600 and you're ratelimited with *retry_after* being 3600 -> library sleeps for 3600 seconds

5.2 Client

The Client object is the base class from which all the requests are made, this is where you can get your games, authenticate and get the models for your authenticated user.

```
class modio.client.Client(*, api_key=None, access_token=None, lang='en', version='v1',
                           test=False, platform=None, portal=None, ratelimit_max_sleep=inf)
```

Represents an authenticated client to make requests to the mod.io API with. If you desire to make aysnc requests you must call `Client.start` before making any async request.

Parameters

- **api_key** (*Optional[str]*) – The api key that will be used to authenticate the bot while it makes most of its GET requests. This can be generated on the mod.io website. Optional if an access token is supplied.
- **access_token** (*Optional[str]*) – The OAuth 2 token that will be used to make more complex GET requests and to make POST requests. This can either be generated using the library's `oauth2` functions or through the mod.io website. This is referred as an access token in the rest of the documentation. If an access token is supplied it will be used for all requests.
- **lang** (*Optional[str]*) – The mod.io API provides localization for a collection of languages. To specify responses from the API to be in a particular language, simply provide the `lang` parameter with an ISO 639 compliant language code. Default is US English.
- **test** (*Optional[bool]*) – Whether or not to use the mod.io test environment. If not included will default to `False`.
- **version** (*Optional[str]*) – An optional keyword argument to allow you to pick a specific version of the API to query, usually you shouldn't need to change this. Default is the latest supported version.
- **platform** (*Optional[TargetPlatform]*) – The platform to target with requests.
- **portal** (*Optional[TargetPortal]*) – The portal to target with requests.
- **ratelimit_max_sleep** (*Optional[int]*) – The maximum amount of time the library will sleep in the case of a `ratelimit`. If the `ratelimit` header returned dictates a longer sleep than that value then the library will instead raise the `ratelimit`. If it is less then the library will sleep for the duration required before retrying the request once.

retry_after

Number of seconds until the rate limits are reset for this API Key/access token. Is 0 until the API returns a 429.

Type `int`

rate_limit

rate_remain

retry_after

set_platform (*platform: Optional[modio.enums.TargetPlatform] = None*) → None

Change the platform targetted by the client. Call without an argument to not target any specific platform.

Parameters **platform** (*Optional[TargetPlatform]*) – The platform to set

set_portal (*portal: Optional[modio.enums.TargetPortal] = None*) → None

Change the portal targetted by the client. Call without an argument to not target any specific portal.

Parameters **portal** (*Optional[TargetPortal]*) – The portal to set

get_game (*game_id: int*) → `modio.game.Game`

Queries the mod.io API for the given game ID and if found returns it as a `Game` instance. If not found raises `NotFound`.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters `game_id` (*int*) – The ID of the game to query the API for

Raises `NotFound` – A game with the supplied id was not found.

Returns The game with the given ID

Return type *Game*

get_games (*, *filters: modio.objects.Filter = None*) → *modio.objects.Returned[modio.game.Game][modio.game.Game]*
 Gets all the games available on mod.io. Returns a named tuple with parameters results and pagination.
 This method takes *filtering arguments*

This method has an async equivalent prefixed with ‘*async_*’. You must use `Client.start` before using the async equivalent.

Parameters `filters` (*Optional[Filter]*) – A instance of `Filter` to be used for filtering, paginating and sorting results

Returns The results and pagination tuple from this request

Return type *Returned[List[Game], Pagination]*

get_my_user () → *modio.entities.User*

Gets the authenticated user’s details (aka the user who created the API key/access token)

This method has an async equivalent prefixed with ‘*async_*’. You must use `Client.start` before using the async equivalent.

Raises `Forbidden` – The access token is invalid/missing

Returns The authenticated user

Return type *User*

get_my_subs (*, *filters: modio.objects.Filter = None*) → *modio.objects.Returned[modio.mod.Mod][modio.mod.Mod]*
 Gets all the mods the authenticated user is subscribed to. This method takes *filtering arguments*

This method has an async equivalent prefixed with ‘*async_*’. You must use `Client.start` before using the async equivalent.

Parameters `filter` (*Optional[Filter]*) – A instance of `Filter` to be used for filtering, paginating and sorting results

Raises `Forbidden` – The access token is invalid/missing

Returns The results and pagination tuple from this request

Return type *Returned[List[Mod], Pagination]*

get_my_events (*, *filters: modio.objects.Filter = None*) → *modio.objects.Returned[modio.entities.Event][modio.entities.Event]*

Get events that have been fired specifically for the authenticated user. This method takes *filtering arguments*

This method has an async equivalent prefixed with ‘*async_*’. You must use `Client.start` before using the async equivalent.

Parameters `filter` (*Optional[Filter]*) – A instance of `Filter` to be used for filtering, paginating and sorting results

Returns The results and pagination tuple from this request

Return type *Returned[List[Event], Pagination]*

get_my_games (*filters: modio.objects.Filter = None*) → *modio.objects.Returned[modio.game.Game][modio.game.Game]*
 Get all the games the authenticated user added or is a team member of. This method takes *filtering arguments*

This method has an async equivalent prefixed with ‘`async_`’. You must use `Client.start` before using the async equivalent.

Parameters `filter` (*Optional*[`Filter`]) – A instance of `Filter` to be used for filtering, paginating and sorting results

Raises `Forbidden` – The access token is invalid/missing

Returns The results and pagination tuple from this request

Return type *Returned*[*List*[`Game`], *Pagination*]

`get_my_mods` (*, *filters*: *modio.objects.Filter* = *None*) → *modio.objects.Returned*[*modio.mod.Mod*][*modio.mod.Mod*]

Get all the mods the authenticated user added or is a team member of. This method takes *filtering arguments*

This method has an async equivalent prefixed with ‘`async_`’. You must use `Client.start` before using the async equivalent.

Parameters `filter` (*Optional*[`Filter`]) – A instance of `Filter` to be used for filtering, paginating and sorting results

Raises `Forbidden` – The access token is invalid/missing

Returns The results and pagination tuple from this request

Return type *Returned*[*List*[`Mod`], *Pagination*]

`get_my_modfiles` (*, *filters*: *modio.objects.Filter* = *None*) → *modio.objects.Returned*[*modio.entities.ModFile*][*modio.entities.ModFile*]

Get all the mods the authenticated user uploaded. The returned modfile objects cannot be edited or deleted and do not have a `game_id` attribute. Returns a named tuple with parameters results and pagination. This method takes *filtering arguments*

This method has an async equivalent prefixed with ‘`async_`’. You must use `Client.start` before using the async equivalent.

Parameters `filter` (*Optional*[`Filter`]) – A instance of `Filter` to be used for filtering, paginating and sorting results

Raises `Forbidden` – The access token is invalid/missing

Returns The results and pagination tuple from this request

Return type *Returned*[*List*[`ModFile`], *Pagination*]

`get_my_ratings` (*, *filters*: *modio.objects.Filter* = *None*) → *modio.objects.Returned*[*modio.entities.Rating*][*modio.entities.Rating*]

Get all the ratings the authenticated user has submitted. Returns a named with parameter results and pagination. This method takes *filtering arguments*

This method has an async equivalent prefixed with ‘`async_`’. You must use `Client.start` before using the async equivalent.

Parameters `filter` (*Optional*[`Filter`]) – A instance of `Filter` to be used for filtering, paginating and sorting results

Raises `Forbidden` – The access token is invalid/missing

Returns The results and pagination tuple from this request

Return type *Returned*[*List*[`Rating`], *Pagination*]

`async_email_exchange` (*code*: *int*, *, *date_expires*: *datetime.datetime* = *None*) → *str*

`async_email_request` (*email*: *str*)

```

async_get_game (game_id: int) → modio.game.Game
async_get_games (*, filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.game.Game][modio.game.Game]
async_get_my_events (*, filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.entities.Event][modio.entities.Event]
async_get_my_games (filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.game.Game][modio.game.Game]
async_get_my_modfiles (*, filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.entities.ModFile][modio.entities.ModFile]
async_get_my_mods (*, filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.mod.Mod][modio.mod.Mod]
async_get_my_mutes (*, filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.entities.User][modio.entities.User]
async_get_my_ratings (*, filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.entities.Rating][modio.entities.Rating]
async_get_my_subs (*, filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.mod.Mod][modio.mod.Mod]
async_get_my_user () → modio.entities.User

```

close ()

This method has no sync equivalent. You must use `Client.start` before using this method This function is used to clean up the client in order to close the application that it uses gracefully. At the moment it is only used to close the client's Session.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

start ()

This method has no sync equivalent. You must use `Client.start` before using this method This function is used to start up the async part of the client. This is required to avoid sync users from having to clean up stuff.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

```
get_my_mutes (*, filters: modio.objects.Filter = None) → modio.objects.Returned[modio.entities.User][modio.entities.User]
```

Get all users muted by this user

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters **filter** (*Optional*[Filter]) – A instance of Filter to be used for filtering, paginating and sorting results

Raises Forbidden – The access token is invalid/missing

Returns The results and pagination tuple from this request

Return type *Returned*[List[User], *Pagination*]

email_request (email: str)

Posts an email request for an OAuth2 token. A code will be sent to the given email address which can then be entered into `email_exchange` ().

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters `email` (*str*) – A valid email to which the 5-digit code will be sent

email_exchange (*code: int, *, date_expires: datetime.datetime = None*) → *str*
Exchanges the given 5-digit code for an OAuth2 token.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters

- **code** (*int*) – A 5-digit code received by email less than 15 minutes ago
- **date_expires** (*Optional[datetime.datetime]*) – Datetime of when the token will expire. By default this is a year, value cannot be greater than a year.

Raises

- `Unauthorized` – Invalid security code
- `ValueError` – Security code was not 5 digits long

Returns The access code.

Return type *str*

5.3 Games

Documentation on the object representing a mod.io Game

Games are the umbrella entities under which all mods are stored.

class `modio.game.Game` (***attrs*)

Represents an instance of a Game. Do not create manually.

id

ID of the game. Filter attribute.

Type *int*

status

Status of the game. (see [status](#) and [visibility](#) for details) Filter attribute.

Type *Status*

submitter

Instance of the modio user who submitted the game. Filter attribute.

Type *Optional[User]*

date

UNIX timestamp of the date the game was registered. Filter attribute.

Type *datetime.datetime*

updated

UNIX timestamp of the date the game was last updated. Filter attribute.

Type *datetime.datetime*

live

UNIX timestamp of the date the game went live. Filter attribute.

Type *datetime.datetime*

presentation

Presentation style used on the mod.io website. Filter attribute.

Type *Presentation*

submission

Submission process modders must follow. Filter attribute.

Type *Submission*

curation

Curation process used to approve mods. Filter attribute.

Type *Curation*

community

Community features enabled on the mod.io website. Filter attribute.

Type *Community*

revenue

Revenue capabilities mods can enable. Filter attribute.

Type *Revenue*

api

Level of API access allowed by this game. Filter attribute.

Type *APIAccess*

maturity_options

Switch to allow developers to select if they flag their mods as containing mature content. Filter attribute.

Type *MaturityOptions*

ugc

Word used to describe user-generated content (mods, items, addons etc). Filter attribute.

Type *str*

icon

The game icon

Type *Image*

logo

The game logo

Type *Image*

header

The game header

Type *Image*

name

Name of the game. Filter attribute.

Type *str*

name_id

sub_domain name for the game (https://name_id.mod.io). Filter attribute.

Type *str*

summary

Summary of the game. Filter attribute.

Type `str`

instructions

Instructions on uploading mods for this game, only applicable if `submission` equals 0

Type `str`

instructions_url

Link to a mod.io guide, your modding wiki or a page where modders can learn how to make and submit mods to your games profile. Filter attribute.

Type `str`

profile

URL to the game's mod.io page.

Type `str`

tag_options

List of tags from which mods can pick

Type `List[TagOption]`

stats

The game stats

Type `Optional[GameStats]`

other_urls

A dictionary of labels and urls for the game

Type `Dict[str, str]`

platforms

Platforms this games supports

Type `List[GamePlatform]`

get_mod (*mod_id: int*) → `modio.mod.Mod`

Queries the mod.io API for the given mod ID and if found returns it as a `Mod` instance. If not found raises `NotFound`.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters `mod_id (int)` – The ID of the mod to query the API for

Raises `NotFound` – A mod with the supplied id was not found.

Returns The mod with the given ID

Return type class: `Mod`

get_mods (*, *filters: modio.objects.Filter = None*) → `modio.objects.Returned[modio.mod.Mod][modio.mod.Mod]`

Gets all the mods available for the game. Returns a named tuple with parameters results and pagination.

This method takes *filtering arguments*

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters `filters (Optional[Filter])` – A instance of `Filter` to be used for filtering, paginating and sorting results

Returns The results and pagination tuple from this request

Return type `Returned[List[Mod], Pagination]`

get_mod_events (*, filters: modio.objects.Filter = None) → modio.objects.Returned[modio.entities.Event][modio.entities.Event]

Gets all the mod events available for this game sorted by latest event first. This method takes *filtering arguments*

This method has an async equivalent prefixed with ‘*async_*’. You must use Client.start before using the async equivalent.

Parameters **filters** (*Optional*[Filter]) – A instance of Filter to be used for filtering, paginating and sorting results

Returns The results and pagination tuple from this request

Return type *Returned*[List[Event], *Pagination*]

get_tag_options (*, filters: modio.objects.Filter = None)

Gets all the game tags available for this game. Updates the tag_option attribute. This method takes *filtering arguments*

This method has an async equivalent prefixed with ‘*async_*’. You must use Client.start before using the async equivalent.

Parameters **filters** (*Optional*[Filter]) – A instance of Filter to be used for filtering, paginating and sorting results

Returns The results and pagination tuple from this request

Return type *Returned*[List[TagOption], *Pagination*]

get_stats (*, filters: modio.objects.Filter = None)

Get the stats for the game. This method takes *filtering arguments*

This method has an async equivalent prefixed with ‘*async_*’. You must use Client.start before using the async equivalent.

Parameters **filter** (*Optional*[Filter]) – A instance of Filter to be used for filtering, paginating and sorting results

Returns The stats for the game.

Return type *GameStats*

get_mods_stats (*, filters: modio.objects.Filter = None)

Gets the stat for all the mods of this game. This method takes *filtering arguments*

This method has an async equivalent prefixed with ‘*async_*’. You must use Client.start before using the async equivalent.

Parameters **filter** (*Optional*[Filter]) – A instance of Filter to be used for filtering, paginating and sorting results

Returns The results and pagination tuple from this request

Return type *Returned*[List[ModStats], *Pagination*]

add_mod (mod: modio.objects.NewMod) → modio.mod.Mod

Add a mod to this game.

This method has an async equivalent prefixed with ‘*async_*’. You must use Client.start before using the async equivalent.

Parameters **mod** (NewMod) – The mod to be submitted

Raises *ValueError* – One of the requirements for a parameter has not been met.

Returns The newly created mod

Return type *Mod***add_media** (*, logo: str = None, icon: str = None, header: str = None)

Upload new media to to the game. This function can take between 1 to 3 arguments depending on what media you desire to upload/update.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Parameters

- **logo** (*Optional[str]*) – Path to the file that you desire to be the game's logo. Dimensions must be at least 640x360 and we recommended you supply a high resolution image with a 16 / 9 ratio. mod.io will use this logo to create three thumbnails with the dimensions of 320x180, 640x360 and 1280x720.
- **icon** (*Optional[str]*) – Path to the file that you desire to be the game's icon. Must be gif, jpg or png format and cannot exceed 1MB in filesize. Dimensions must be at least 64x64 and a transparent png that works on a colorful background is recommended. mod.io will use this icon to create three thumbnails with the dimensions of 64x64, 128x128 and 256x256.
- **header** (*Optional[str]*) – Path to the file that you desire to be the game's header. Must be gif, jpg or png format and cannot exceed 256KB in filesize. Dimensions of 400x100 and a light transparent png that works on a dark background is recommended.

Returns A message containing the result of the query if successful.

Return type *Message***add_tag_options** (name: str, *, tags: *Optional[List[str]]* = None, hidden: *Optional[bool]* = False, locked: *Optional[bool]* = False, tag_type: *Optional[Literal[dropdown, checkboxes]]* = 'dropdown')

Add tags which mods can apply to their profiles. If the tag names already exists, settings such as hidden or type will be overwritten to the values provided and all the tags will be added to the group.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Parameters

- **name** (*str*) – Name of the tag group
- **type** (*Optional[Literal['dropdown', 'checkboxes']]*) – Defaults to dropdown dropdown : Mods can select only one tag from this group, dropdown menu shown on site profile. checkboxes : Mods can select multiple tags from this group, checkboxes shown on site profile.
- **hidden** (*Optional[bool]*) – Whether or not this group of tags should be hidden from users and mod devs. Defaults to False
- **locked** (*Optional[bool]*) – Whether or not mods can assign from this group of tag to themselves. If locked only game admins will be able to assign the tag. Defaults to False.
- **tags** (*Optional[List[str]]*) – Array of tags that mod creators can apply to their mod

async_add_media (*, logo: str = None, icon: str = None, header: str = None)**async_add_mod** (mod: modio.objects.NewMod) → modio.mod.Mod**async_add_tag_options** (name: str, *, tags: *Optional[List[str]]* = None, hidden: *Optional[bool]* = False, locked: *Optional[bool]* = False, tag_type: *Optional[Literal[dropdown, checkboxes]]* = 'dropdown')


```

async_delete_tag_options (name: str, *, tags: Optional[List[str]] = None) → bool
async_get_mod (mod_id: int) → modio.mod.Mod
async_get_mod_events (*, filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.entities.Event][modio.entities.Event]
async_get_mods (*, filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.mod.Mod][modio.mod.Mod]
async_get_mods_stats (*, filters: modio.objects.Filter = None)
async_get_owner () → modio.entities.User
async_get_stats (*, filters: modio.objects.Filter = None)
async_get_tag_options (*, filters: modio.objects.Filter = None)
async_report (name: str, summary: str, report_type: modio.enums.Report = <Report.generic: 0>)
get_owner () → modio.entities.User

```

Get the original submitter of the resource.

This method has an async equivalent prefixed with ‘*async_*’. You must use `Client.start` before using the async equivalent.

Returns The original submitter

Return type *User*

```

report (name: str, summary: str, report_type: modio.enums.Report = <Report.generic: 0>)

```

Report a this game, make sure to read mod.io’s ToU to understand what is and isnt allowed.

This method has an async equivalent prefixed with ‘*async_*’. You must use `Client.start` before using the async equivalent.

Parameters

- **name** (*str*) – Name of the report
- **summary** (*str*) – Detailed description of your report. Make sure you include all relevant information and links to help moderators investigate and respond appropriately.
- **report_type** (*Report*) – Report type

Returns The returned message on the success of the query.

Return type *Message*

```

delete_tag_options (name: str, *, tags: Optional[List[str]] = None) → bool

```

Delete one or more tags from a tag option.

This method has an async equivalent prefixed with ‘*async_*’. You must use `Client.start` before using the async equivalent.

Parameters

- **name** (*str*) – Name of the group from which you wish to delete from
- **tags** (*Optional[List[str]]*) – Optional. Tags to delete from group. If left blank the entire group will be deleted

Returns Returns True if the tags were sucessfully removed, False if the requests was sucessful but the tags was not removed (if the tag wasn’t part of the option.)

Return type *bool*

5.4 Mod

Documentation on the object representing a mod.io Mod

Module storing representation of the mod objects

```
class modio.mod.Mod (**attrs)  
    Represent a modio mod object.
```

Filter-Only Attributes

These attributes can only be used at endpoints which return instances of this class and takes filter arguments. They are not attached to the object itself and trying to access them will cause an `AttributeError`

sort_downloads [str] Sort argument, provide to sort function to sort by most/least downloaded

sort_popular [str] Sort argument, provide to sort function to sort by most/least popular

sort_rating [str] Sort argument, provide to sort function to sort by weighed rating

sort_subscribers [str] Sort argument, provide to sort function to sort by most/least subscribers

id

ID of the mod. Filter attribute.

Type `int`

status

Status of the mod. Filter attribute.

Type `Status`

visible

Visibility of the mod. Filter attribute.

Type `Visibility`

game_id

ID of the game the mod is for. Filter attribute.

Type `int`

submitter

Instance of the modio User that submitted the mod. Filter attribute.

Type `User`

date

UNIX timestamp of the date the mod was registered. Filter attribute.

Type `datetime.datetime`

updated

UNIX timestamp of the date the mod was last updated. Filter attribute.

Type `datetime.datetime`

live

UNIX timestamp of the date the mod went live. Filter attribute.

Type `datetime.datetime`

logo

The mod logo

Type *Image*

homepage

Link to the homepage of the mod, can be None. Filter attribute.

Type *str*

name

Name of the mod. Filter attribute.

Type *str*

name_id

sub_domain mod for the game (https://game_name.mod.io/name_id). Filter attribute.

Type *str*

summary

Summary of the mod. Filter attribute.

Type *str*

description

Detailed description of the mod, supports HTML. Filter attribute.

Type *str*

metadata

Metadata stored by developers which may include properties on how information required. Can be None. Filter attribute.

Type *str*

maturity

Maturity option of the mod. Filter attribute.

Type *Maturity*

profile

URL of the mod's modio profile

Type *str*

file

Latest released instance. Can be None. Filter attribute.

Type *ModFile*

media

Contains mod media data (links and images)

Type *ModMedia*

stats

Summary of all stats for this mod

Type *ModStats*

tags

Tags for this mod. Filter attribute.

Type *dict*

kvp

Contains key-value metadata. Filter attribute.

Type *dict*

plaintext

description field converted into plaintext.

Type `str`

mod_key = `'id'`

kvp

get_file (*file_id: int*) → `modio.entities.ModFile`

Get the Mod File with the following ID.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters **file_id** (*int*) – ID of the mod file you wish to retrieve

Raises `NotFound` – A mod with that ID has not been found

Returns The found modfile

Return type `ModFile`

get_files (*, *filters: modio.objects.Filter = None*) → `modio.objects.Returned[modio.entities.ModFile][modio.entities.ModFi`

Get all mod files for this mod. Returns a named tuple with parameters results and pagination. This method takes *filtering arguments*

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters **filter** (*Optional[Filter]*) – A instance of `Filter` to be used for filtering, paginating and sorting results

Returns The results and pagination tuple from this request

Return type `Returned[List[ModFile], Pagination]`

get_events (*, *filters: modio.objects.Filter = None*) → `modio.objects.Returned[modio.entities.Event][modio.entities.Event]`

Get all events for that mod sorted by latest. Returns, a named tuple with parameters results and pagination. This method takes *filtering arguments*

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters **filter** (*Optional[Filter]*) – A instance of `Filter` to be used for filtering, paginating and sorting results

Returns The results and pagination tuple from this request

Return type `Returned[List[Event], Pagination]`

get_tags (*, *filters: modio.objects.Filter = None*) → `modio.objects.Returned[dict][dict]`

Gets all the tags for this mod. Updates the instance's tag attribute. Returns a named tuple with parameters results and pagination. This method takes *filtering arguments*

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters **filter** (*Optional[Filter]*) – A instance of `Filter` to be used for filtering, paginating and sorting results

Returns The results and pagination tuple from this request

Return type `Returned[List[Tag], Pagination]`

get_metadata () → modio.objects.Returned[dict][dict]

Returns a dict of metakey-metavalue pairs. This will also update the mod's kvp attribute.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Returns The results and pagination tuple from this request

Return type *Returned*[List[*MetaData*], *Pagination*]

get_dependencies (*, filters: modio.objects.Filter = None) → modio.objects.Returned[dict][dict]

Returns a dict of dependency_id-date_added pairs. Returns a named tuple with parameters results and pagination. This method takes *filtering arguments*

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Parameters filter (*Optional*[*Filter*]) – A instance of Filter to be used for filtering, paginating and sorting results

Returns The results and pagination tuple from this request

Return type *Returned*[List[*Dependency*], *Pagination*]

get_team (*, filters: modio.objects.Filter = None) → modio.objects.Returned[modio.entities.TeamMember][modio.entities.TeamMember]

Returns a list of TeamMember object representing the Team in charge of the mod. This method takes *filtering arguments*

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Parameters filter (*Optional*[*Filter*]) – A instance of Filter to be used for filtering, paginating and sorting results

Returns The results and pagination tuple from this request

Return type *Returned*[List[*TeamMember*], *Pagination*]

get_comments (*, filters: modio.objects.Filter = None) → modio.objects.Returned[modio.entities.Comment][modio.entities.Comment]

Returns a list of all the top level comments for this mod with comments replying to top level comments stored in the children attribute. This can be flattened using the utils.flatten function. This method takes *filtering arguments*

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Parameters filter (*Optional*[*Filter*]) – A instance of Filter to be used for filtering, paginating and sorting results

Returns The results and pagination tuple from this request

Return type *Returned*[List[*Comment*], *Pagination*]

add_comment (content: str, *, reply: int = None) → modio.entities.Comment

Add a comment to the mod page. You can specify a comment to reply too.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Parameters

- **content** (*str*) – The content of the comment
- **reply** (*Optional*[*Comment*]) – The comment to reply to

Returns The comment created

Return type *Comment*

get_stats () → modio.entities.ModStats

Returns a ModStats object, representing a series of stats for the mod.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Returns The stats summary object for the mod.

Return type Stats

edit (***fields*) → modio.mod.Mod

Used to edit the mod details. Successful editing will return the updated mod.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Parameters

- **status** (*Status*) – For game admins only.
- **visible** (*Visibility*) – Modify the game visibility
- **name** (*str*) – Name of the mod, cannot exceed 80 characters
- **name_id** (*str*) – Subdomain for the mod, cannot exceed 80 characters
- **summary** (*str*) – Summary of the mod, cannot exceed 250 characters
- **description** (*str*) – Detailed description for your mod, which can include details such as 'About', 'Features', 'Install Instructions', 'FAQ', etc. HTML supported and encouraged.
- **homepage** (*str*) – URL to the official homepage for this mod.
- **stock** (*str*) – Maximum number of subscribers for this mod. A value of 0 disables this limit.
- **maturity** (*Maturity*) – Maturity option of the mod.
- **metadata** (*str*) – Metadata stored by the mod developer which may include properties as to how the item works, or other information you need to display.

Returns The updated version of the mod

Return type *Mod*

delete ()

Delete a mod and set its status to deleted.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

add_file (*file: modio.objects.NewModFile*) → modio.entities.ModFile

Adds a new file to the mod, to do so first construct an instance of NewModFile and then pass it to the function.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Parameters **file** (*NewModFile*) – The mod file to upload

Raises *modioException* – file argument must be type NewModFile

Returns The modfile after being processed by the mod.io API

Return type *ModFile*

add_media (*, *logo*: *Optional[str]* = None, *images*: *Union[str, List[str], None]* = (), *youtube*: *List[str]* = (), *sketchfab*: *List[str]* = ())

Upload new media to the mod.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters

- **logo** (*Optional[str]*) – Path to the logo file. If on windows, must be escaped. Image file which will represent your mods logo. Must be gif, jpg or png format and cannot exceed 8MB in filesize. Dimensions must be at least 640x360 and we recommended you supply a high resolution image with a 16 / 9 ratio. mod.io will use this logo to create three thumbnails with the dimensions of 320x180, 640x360 and 1280x720.
- **images** (*Optional[Union[str, list]]*) – Can be either the path to a file called .zip file containing all the images or a list of paths to multiple image files. If on windows, must be escaped. Only valid gif, jpg and png images in the zip file will be processed.
- **youtube** (*Optional[List[str]]*) – List of youtube links to be added to the gallery
- **sketchfab** (*Optional[List[str]]*) – List of sketchfab links to be added to the gallery.

Returns A message confirming the submission of the media

Return type *Message*

delete_media (*, *images*: *Optional[List[str]]* = (), *youtube*: *Optional[List[str]]* = (), *sketchfab*: *Optional[List[str]]* = ())

Delete media from the mod page.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters

- **images** (*Optional[List[str]]*) – Optional. List of image filenames that you want to delete
- **youtube** (*Optional[List[str]]*) – Optional. List of youtube links that you want to delete
- **sketchfab** (*Optional[List[str]]*) – Optional. List sketchfab links that you want to delete

subscribe () → modio.mod.Mod

Subscribe to the mod. Returns None if user is already subscribed.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Returns The mod that was just subscribed to, if the user was already subscribed it will return None

Return type *Mod*

unsubscribe ()

Unsubscribe from a mod. Returns None if the user is not subscribed.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

add_tags (*tags)

Add tags to a mod, tags are case insensitive and duplicates will be removed. Tags which are not in the game's tag_options will not be added.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Parameters **tags** (*List[str]*) – list of tags to be added.

delete_tags (*tags)

Delete tags from the mod, tags are case insensitive and duplicates will be removed. Providing no arguments will remove every tag from the mod.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Parameters **tags** (*List[str]*) – List of tags to remove, if no list is provided, will remove every tag from the mod.

add_metadata (**metadata)

Add metadata key-value pairs to the mod. To submit new meta data, pass meta data keys as keyword arguments and meta data value as a list of values. E.g pistol_dmg = [800, 400]. Keys support alphanumeric, '-' and '_'. Total length of key and values cannot exceed 255 characters. To add meta-keys which contain a dash in their name they must be passed as an unpacked dictionary.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Example

mod.add_metadata(difficulty=["hard", "medium", "easy"]) This will add the values "hard", "medium" and "easy" to the meta key "difficulty"

mod.add_metadata({"test-var": ["test1", "test2", "test3"]})** This will add the values "test1", "test2" and "test3" to meta key "test-var"

Returns message on the status of the successful added meta data

Return type *Message*

add_negative_rating ()

Changes the mod rating to negative, the author of the rating will be the authenticated user. If the mod has already been negatively rated by the user it will return False. If the negative rating is successful it will return True.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

add_positive_rating ()

Changes the mod rating to positive, the author of the rating will be the authenticated user. If the mod has already been positively rated by the user it will return False. If the positive rating is successful it will return True.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

async_add_comment (content: str, *, reply: int = None) → modio.entities.Comment

async_add_dependencies (dependencies: List[Union[int, Mod]])

async_add_file (file: modio.objects.NewModFile) → modio.entities.ModFile


```

async_add_media (*, logo: Optional[str] = None, images: Union[str, List[str], None] = (), youtube:
    List[str] = (), sketchfab: List[str] = ())
async_add_metadata (**metadata)
async_add_negative_rating()
async_add_positive_rating()
async_add_tags(*tags)
async_add_team_member(email: str, level: modio.enums.Level, *, position: Optional[str] = None)
async_delete()
async_delete_dependencies(dependencies: List[Union[int, Mod]])
async_delete_media(* , images: Optional[List[str]] = (), youtube: Optional[List[str]] = (), sketch-
    fab: Optional[List[str]] = ())
async_delete_metadata(**metadata)
async_delete_tags(*tags)
async_edit(**fields) → modio.mod.Mod
async_get_comments(* , filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.entities.Comment][modio.entities.Comment]
async_get_dependencies(* , filters: modio.objects.Filter = None) →
    modio.objects.Returned[dict][dict]
async_get_events(* , filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.entities.Event][modio.entities.Event]
async_get_file(file_id: int) → modio.entities.ModFile
async_get_files(* , filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.entities.ModFile][modio.entities.ModFile]
async_get_metadata() → modio.objects.Returned[dict][dict]
async_get_owner() → modio.entities.User
async_get_stats() → modio.entities.ModStats
async_get_tags(* , filters: modio.objects.Filter = None) → modio.objects.Returned[dict][dict]
async_get_team(* , filters: modio.objects.Filter = None) →
    modio.objects.Returned[modio.entities.TeamMember][modio.entities.TeamMember]
async_report(name: str, summary: str, report_type: modio.enums.Report = <Report.generic: 0>)
async_subscribe() → modio.mod.Mod
async_unsubscribe()
delete_metadata(**metadata)

```

Deletes metadata from a mod. To do so pass the meta-key as a keyword argument and the meta-values you wish to delete as a list. You can pass an empty list in which case all meta-values for the meta-key will be deleted. To delete meta-keys which contain a dash in their name they must be passed as an unpacked dictionary.

This method has an async equivalent prefixed with ‘*async_*’. You must use `Client.start` before using the async equivalent.

Example

`mod.delete_metadata(difficulty=["easy"])` This will remove the value “easy” from the meta key “difficulty”

`mod.delete_metadata(difficulty=[])` This will remove the meta key “difficulty”

`mod.delete_metadata(**{"test-var": ["test1"]})` This will remove the value “test1” from the meta key “test-var”

`mod.delete_metadata(**{"test-var": []})` This will remove the meta key “test-var”

`get_owner()` → `modio.entities.User`

Get the original submitter of the resource.

This method has an async equivalent prefixed with ‘`async_`’. You must use `Client.start` before using the async equivalent.

Returns The original submitter

Return type *User*

`report(name: str, summary: str, report_type: modio.enums.Report = <Report.generic: 0>)`

Report a this game, make sure to read mod.io’s ToU to understand what is and isn’t allowed.

This method has an async equivalent prefixed with ‘`async_`’. You must use `Client.start` before using the async equivalent.

Parameters

- **`name`** (*str*) – Name of the report
- **`summary`** (*str*) – Detailed description of your report. Make sure you include all relevant information and links to help moderators investigate and respond appropriately.
- **`report_type`** (*Report*) – Report type

Returns The returned message on the success of the query.

Return type *Message*

`add_dependencies(dependencies: List[Union[int, Mod]])`

Add mod dependencies required by the corresponding mod. A dependency is a mod that should be installed for this mod to run.

This method has an async equivalent prefixed with ‘`async_`’. You must use `Client.start` before using the async equivalent.

Parameters **`dependencies`** (*List[Union[int, Mod]]*) – List of mod ids to submit as dependencies.

`delete_dependencies(dependencies: List[Union[int, Mod]])`

Delete mod dependencies required by this mod. You must supply at least one dependency.

This method has an async equivalent prefixed with ‘`async_`’. You must use `Client.start` before using the async equivalent.

Parameters **`dependencies`** (*List[Union[int, Mod]]*) – List of dependencies to remove

`add_team_member(email: str, level: modio.enums.Level, *, position: Optional[str] = None)`

Add a user to the mod team. Will fire a `MOD_TEAM_CHANGED` event.

This method has an async equivalent prefixed with ‘`async_`’. You must use `Client.start` before using the async equivalent.

Parameters

- **email** (*str*) – mod.io email of the user you wish to add
- **level** (*Level*) – Level of permissions you grant the user
- **position** (*Optional[str]*) – Title of the user position

5.5 Misc Models

Documentation on all the other objects returned by mod.io requests which are not large enough to warrant their own page.

Module for miscs objects.

class modio.entities.**Message** (***attrs*)

A simple representation of a modio Message, used when modio returns a status message for the query that was accomplished.

code

An http response code

Type *int*

message

The server response to the request

Type *str*

class modio.entities.**Image** (***attrs*)

A representation of a modio image, which stand for the Logo, Icon and Header of a game/mod or the Avatar of a user.Can also be a regular image.

filename

Name of the file

Type *str*

original

Link to the original file

Type *str*

small

A link to a smaller version of the image, processed by Size varies based on the object being processed. Can be None.

Type *str*

medium

A link to a medium version of the image, processed by Size varies based on the object being processed. Can be None.

Type *str*

large

A link to a large version of the image, processed by Size varies based on the object being processed. Can be None.

Type *str*

class modio.entities.**Event** (***attrs*)

Represents a mod event.

Filter-Only Attributes

These attributes can only be used at endpoints which return instances of this class and takes filter arguments. They are not attached to the object itself and trying to access them will cause an `AttributeError`

latest [bool] Returns only the latest unique events, which is useful for checking if the primary modfile has changed.

subscribed [bool] Returns only events connected to mods the authenticated user is subscribed to, which is useful for keeping the users mods up-to-date.

id
Unique ID of the event. Filter attribute.

Type `int`

mod
ID of the mod this event is from. Filter attribute.

Type `int`

user
ID of the user that made the change. Filter attribute.

Type `int`

date
UNIX timestamp of the event occurrence. Filter attribute.

Type `datetime.datetime`

type
Type of the event. Filter attribute.

Type `EventType`

game_id
ID of the game that the mod the user change came from. Can be None if it is a mod event. Filter attribute.

Type `int`

type

class `modio.entities.Comment` (***attrs*)
Represents a comment on a mod page.

id
ID of the comment. Filter attribute.

Type `int`

resource_id
The parent resource. Filter attribute.

Type `int`

user
Instance of the user that submitted the comment. Filter attribute.

Type `User`

date
Unix timestamp of date the comment was posted. Filter attribute.

Type `datetime.datetime`

parent_id

ID of the parent this comment is replying to. 0 if comment is not a reply. Filter attribute.

Type `int`

position

The position of the comment. Filter attribute. How it works: - The first comment will have the position '01'. - The second comment will have the position '02'. - If someone responds to the second comment the position will be '02.01'. - A maximum of 3 levels is supported.

Type `int`

karma

Total karma received for the comment. Filter attribute.

Type `int`

karma_guest

Total karma received from guests for this comment

Type `int`

content

Content of the comment. Filter attribute.

Type `str`

children

List of comment replying to this one

Type `List[Comment]`

level

The level of nesting from 1 to 3 where one is top level and three is the deepest level

Type `int`

edit (*content*)

Update the contents of a comment.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Parameters **content** (*str*) – The new content of the comment

Returns The comment with the new content

Return type *Comment*

delete ()

Remove the comment.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

add_positive_karma ()

Add positive karma to the comment

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Returns The updated comment

Return type *Comment*

add_negative_karma()

Add negative karma to the comment

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Returns The updated comment

Return type *Comment*

async_add_negative_karma()

async_add_positive_karma()

async_delete()

Remove the comment.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

async_edit() (*content*)

class modio.entities.**ModFile** (***attrs*)

A object to represents modfiles. If the modfile has been returned for the `me/modfile` endpoint then `edit()` and `delete()` cannot be called as a game is lacking.

id

ID of the modfile. Filter attribute.

Type *int*

mod

ID of the mod it was added for. Filter attribute.

Type *int*

date

UNIX timestamp of the date the modfile was submitted. Filter attribute.

Type *datetime.datetime*

scanned

UNIX timestamp of the date the file was virus scanned. Filter attribute.

Type *datetime.datetime*

virus_status

Current status of the virus scan for the file. Filter attribute.

Type *VirusStatus*

virus

True if a virus was detected, False if it wasn't. Filter attribute.

Type *bool*

virus_hash

VirusTotal proprietary hash to view the scan results.

Type *str*

size

Size of the file in bytes. Filter attribute.

Type *int*

hash
MD5 hash of the file. Filter attribute.
Type `str`

filename
Name of the file. Filter attribute.
Type `str`

version
Version of the file. Filter attribute.
Type `str`

changelog
Changelog for the file. Filter attribute.
Type `str`

metadata
Metadata stored by the game developer for this file. Filter attribute.
Type `str`

url
url to download file
Type `str`

date_expires
UNIX timestamp of when the url expires
Type `datetime.datetime`

game_id
ID of the game of the mod this file belongs to. Can be None if this file was returned from the me/modfiles endpoint.
Type `int`

platforms
List of platforms this file is available on.
Type `List[ModFilePlatform]`

edit (***fields*)
Edit the file's details. Returns an updated instances of the file.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters

- **version** (`str`) – Change the release version of the file
- **changelog** (`str`) – Change the changelog of this release
- **active** (`bool`) – Change whether or not this is the active version.
- **metadata_blob** (`str`) – Metadata stored by the game developer which may include properties such as what version of the game this file is compatible with.

Returns The updated file

Return type `ModFile`

delete()

Deletes the modfile, this will raise an error if the file is the active release for the mod.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Raises `Forbidden` – You cannot delete the active release of a mod

url_is_expired()

Check if the url is still valid for this modfile.

Returns True if it's still valid, else False

Return type `bool`

async_delete()

async_edit(fields)**

async_get_owner() → `modio.entities.User`

get_owner() → `modio.entities.User`

Get the original submitter of the resource.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Returns The original submitter

Return type `User`

class `modio.entities.ModMedia(**attrs)`

Represents all the media for a mod.

youtube

A list of youtube links

Type `List[str]`

sketchfab

A list of SketchFab links

Type `List[str]`

images

A list of image objects (gallery)

Type `List[Image]`

class `modio.entities.BasePlatform(**attrs)`

Base class for a platform.

class `modio.entities.GamePlatform(**attrs)`

The platform for a game.

platform

The platform

Type `TargetPlatform`

label

The human readable platform label

Type `str`

moderated

Whether the platform is moderated by game admins

Type `bool`

class `modio.entities.ModPlatform(**attrs)`

The platform for a mod

platform

The platform

Type `TargetPlatform`

modfile_live

The ID of the modfile currently live for that platform.

Type `int`

class `modio.entities.ModFilePlatform(**attrs)`

The platform for a mod file

platform

The platform

Type `TargetPlatform`

status

The status of the modfile for the corresponding platform.

Type `ModFilePlatformStatus`

class `modio.entities.TagOption(**attrs)`

Represents a game tag group, a category of tags from which a mod may pick one or more.

name

Name of the tag group

Type `str`

type

Can be either “checkbox” where users can chose multiple tags from the list or “dropdown” in which case only one tag can be chosen from the group

Type `str`

hidden

Whether or not the tag is only accessible to game admins, used for internal mod filtering.

Type `bool`

locked

Whether or not mods can self assign from this tag option.

Type `bool`

tags

Array of tags for this group

Type `List[str]`

class `modio.entities.Rating(**attrs)`

Represents a rating, objects obtained from the `get_my_ratings` endpoint

game_id

The ID of the game the rated mod is for.

Type `int`

mod_id

The ID of the mod that was rated

Type `int`

rating

The rating type

Type `RatingType`

date

UNIX timestamp of whe the rating was added

Type `datetime.datetime`

mod_key = 'mod_id'

add_negative_rating()

Changes the mod rating to negative, the author of the rating will be the authenticated user. If the mod has already been negatively rated by the user it will return False. If the negative rating is successful it will return True.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

add_positive_rating()

Changes the mod rating to positive, the author of the rating will be the authenticated user. If the mod has already been positevely rated by the user it will return False. If the positive rating is successful it will return True.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

async_add_negative_rating()

async_add_positive_rating()

async_delete()

delete()

Removes a rating. Returns true if the rating was succcessfully removed.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

class modio.entities.**ModStats** (**attrs)

Represents a summary of stats for a mod

id

Mod ID of the stats. Filter attribute.

Type `int`

rank

Current rank of the mod. Filter attribute.

Type `int`

rank_total

Number of ranking spots the current rank is measured against. Filter attribute

Type `int`

downloads

Amount of times the mod was downloaded. Filter attribute

Type `int`

subscribers
Amount of subscribers. Filter attribute

Type `int`

total
Number of times this item has been rated.

Type `int`

positive
Number of positive ratings. Filter attribute

Type `int`

negative
Number of negative ratings. Filter attribute

Type `int`

percentage
Percentage of positive rating (positive/total)

Type `int`

weighted
Overall rating of this item calculated using the Wilson score confidence interval. This column is good to sort on, as it will order items based on number of ratings and will place items with many positive ratings above those with a higher score but fewer ratings.

Type `int`

text
Textual representation of the rating in format. This is currently not updated by the lib so you'll have to poll the resource's endpoint again.

Type `str`

date_expires
Unix timestamp until this mods's statistics are considered stale. Endpoint should be polled again when this expires.

Type `datetime.datetime`

is_stale `() → bool`
Returns a bool depending on whether or not the stats are considered stale.

Returns True if stats are expired, False else.

Return type `bool`

class `modio.entities.GameStats` `(**attrs)`
A stat object containing the stats specific to games

id
The id of the game

Type `int`

mods_count_total
The total count of mods for this game

Type `int`

mods_download_today

The amount of mod downloaded today

Type `int`

mods_download_total

The amount of mods downloaded all times

Type `int`

mods_download_daily_avg

Average daily mod downlaods

Type `int`

mods_subscribers_total

Total amount of subscribers to all mods

Type `int`

date_expires

The date at which the stats are considered “stale” and no longer accurate.

Type `datetime.datetime`

is_stale() → `bool`

Returns a bool depending on whether or not the stats are considered stale.

Returns True if stats are expired, False else.

Return type `bool`

class `modio.entities.Theme` (***attrs*)

Object representing a game’s theme. This is mostly useful if you desire to create a visual interface for a game or one of its mods. All attributes are hex color codes.

primary

Primary color of the game

Type `string`

dark

The “dark” color of the game

Type `string`

light

The “light” color of the game

Type `string`

success

The color of a successful action with the game interface

Type `string`

warning

The color of a warning with the game interface

Type `string`

danger

The color of a danger warning with the game interface

Type `string`

class modio.entities.Tag

mod.io Tag objects are represented as dictionaries and are returned as such by the function of this library, each entry of in the dictionary is composed of the tag name as the key and the date_added as the value. Use dict.keys() to access tags as a list.

Filter-Only Attributes

These attributes can only be used at endpoints which return instances of this class and takes filter arguments. They are not attached to the object itself and trying to access them will cause an AttributeError

date [datetime.datetime] Unix timestamp of date tag was added.

tag [str] String representation of the tag.

class modio.entities.MetaData

mod.io MetaData objects are represented as dictionaries and are returned as such by the function of this library, each entry of in the dictionary is composed of the metakey as the key and the metavalue as the value.

class modio.entities.Dependencies

mod.io Dependencies objects are represented as dictionaries and are returned as such by the function of this library, each entry of in the dictionary is composed of the dependency (mod) id as the key and the date_added as the value. Use dict.keys() to access dependencies as a list.

class modio.entities.User (**attrs)

Represents a modio user.

id

ID of the user. Filter attribute.

Type int

name_id

Subdomain name of the user. For example: <https://mod.io/members/username-id-here>. Filter attribute.

Type str

username

Name of the user. Filter attribute.

Type str

last_online

Unix timestamp of date the user was last online.

Type datetime.datetime

avatar

Contains avatar data

Type Image

tz

Timezone of the user, format is country/city. Filter attribute.

Type str

lang

Users language preference. See localization for the supported languages. Filter attribute.

Type str

profile

URL to the user's mod.io profile.

Type `str`

mute ()

Mute a user, this will hide all mods authored by them from the authenticated user.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

unmute ()

Unmute a user, this will show all mods authored by them from the authenticated user.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

async_mute ()

async_report (*name: str, summary: str, report_type: modio.enums.Report = <Report.generic: 0>*)

async_unmute ()

report (*name: str, summary: str, report_type: modio.enums.Report = <Report.generic: 0>*)

Report a this game, make sure to read mod.io's ToU to understand what is and isn't allowed.

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Parameters

- **name** (`str`) – Name of the report
- **summary** (`str`) – Detailed description of your report. Make sure you include all relevant information and links to help moderators investigate and respond appropriately.
- **report_type** (`Report`) – Report type

Returns The returned message on the success of the query.

Return type `Message`

class `modio.entities.TeamMember` (***attrs*)

Inherits from `User`. Represents a user as part of a team. .. rubric:: Filter-Only Attributes

These attributes can only be used at endpoints which return instances of this class and takes filter arguments. They are not attached to the object itself and trying to access them will cause an `AttributeError`

user_id [int] Unique id of the user.

username [str] Username of the user.

id

ID of the user. Filter attribute.

Type `int`

name_id

Subdomain name of the user. For example: <https://mod.io/members/username-id-here>. Filter attribute.

Type `str`

username

Name of the user. Filter attribute.

Type `str`

last_online

Unix timestamp of date the user was last online.

Type `datetime.datetime`

avatar

Contains avatar data

Type `Image`

tz

Timezone of the user, format is country/city. Filter attribute.

Type `str`

lang

Users language preference. See localization for the supported languages. Filter attribute.

Type `str`

profile

URL to the user's mod.io profile.

Type `str`

team_id

The id of the user in the context of their team, not the same as user id. Filter attribute.

Type `int`

level

Permission level of the user

Type `Level`

date

Unix timestamp of the date the user was added to the team. Filter attribute.

Type `datetime.datetime`

position

Custom title given to the user in this team. Filter attribute.

Type `str`

mod

The mod object the team is attached to.

Type `Mod`

async_mute()

async_report (*name: str, summary: str, report_type: modio.enums.Report = <Report.generic: 0>*)

async_unmute()

mute()

Mute a user, this will hide all mods authored by them from the authenticated user.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

report (*name: str, summary: str, report_type: modio.enums.Report = <Report.generic: 0>*)

Report a this game, make sure to read mod.io's ToU to understand what is and isnt allowed.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

Parameters

- **name** (*str*) – Name of the report
- **summary** (*str*) – Detailed description of your report. Make sure you include all relevant information and links to help moderators investigate and respond appropriately.
- **report_type** (*Report*) – Report type

Returns The returned message on the success of the query.

Return type *Message*

unmute ()

Unmute a user, this will show all mods authored by them from the authenticated user.

This method has an async equivalent prefixed with 'async_'. You must use Client.start before using the async equivalent.

5.6 modio Objects

Documentation on objects unique to this library which the user manipulates and sometimes creates.

Module for user instanced classes.

class modio.objects.**NewMod** (***attrs*)

This class is unique to the library, it represents a mod to be submitted. The class must be instantiated with the appropriate parameters and then passed to game.add_mod().

Parameters

- **name** (*str*) – Name of the mod.
- **name_id** (*Optional[str]*) – Subdomain name for the mod. Optional, if not specified the name will be use. Cannot exceed 80 characters
- **summary** (*str*) – Brief overview of the mod, cannot exceed 250 characters.
- **description** (*Optional[str]*) – Detailed description of the mod, supports HTML.
- **homepage** (*Optional[str]*) – Official homepage for your mod. Must be a valid URL. Optional
- **stock** (*Optional[int]*) – Maximum number of subscribers for this mod. Optional, if not included disables
- **metadata** (*Optional[str]*) – Metadata stored by developers which may include properties on how information required. Optional. E.g. "rogue,hd,high-res,4k,hd textures"
- **maturity** (*Optional[Maturity]*) – Choose if the mod contains mature content.
- **visible** (*Optional[Visibility]*) – Visibility status of the mod
- **logo** (*str*) – Path to the file. If on windows, must have escaped.

add_tags (**tags*)

Used to add tags to the mod, returns self for fluid chaining.

Parameters **tags** (*List[str]*) – List of tags, duplicate tags will be ignored.

class modio.objects.**NewModFile** (***attrs*)

This class is unique to the library and represents a file to be submitted. The class must be instantiated and then passed to mod.add_file().

Parameters

- **version** (*str*) – Version of the mod that this file represents
- **changelog** (*str*) – Changelog for the release
- **active** (*Optional[bool]*) – Label this upload as the current release. Optional, if not included defaults to True.
- **metadata** (*str*) – Metadata stored by the game developer which may include properties such as what version of the game this file is compatible with.

add_file (*path*)

Used to add a file.

The binary file for the release. For compatibility you should ZIP the base folder of your mod, or if it is a collection of files which live in a pre-existing game folder, you should ZIP those files. Your file must meet the following conditions:

- File must be zipped and cannot exceed 10GB in filesize
- **Mods which span multiple game directories are not supported** unless the game manages this
- Mods which overwrite files are not supported unless the game manages this

Parameters **path** (*str*) – Path to file, if on windows must be escaped.

class modio.objects.**Filter** (*filters=None*)

This class is unique to the library and is an attempt to make filtering modio data easier. Instead of passing filter keywords directly you can pass an instance of this class which you have previously fine tuned through the various methods. For advanced users it is also possible to pass filtering arguments directly to the class given that they are already in modio format. If you don't know the modio format simply use the methods, all method return self for fluid chaining. This is also used for sorting and pagination. These instances can be save and reused at will. Attributes which can be used as filters will be marked as "Filter attributes" in the docs for the class the endpoint returns an array of. E.g. ID is marked as a filter argument for in the class Game and therefore in `get_games()` it can be used a filter.

Parameters **filters** (*Optional[dict]*) – A dict which contains modio filter keyword and the appropriate value.

text (*query*)

Full-text search is a lenient search filter that is only available if the endpoint you are querying contains a name column.

Parameters **query** (*str*) – The words to identify. `filter.text("The Lord of the Rings")` - This will return every result where the name column contains any of the following words: 'The', 'Lord', 'of', 'the', 'Rings'.

equals (***kwargs*)

The simplest filter you can apply is columnname equals. This will return all rows which contain a column matching the value provided. There are not set parameters, this methods takes any named keywords and transforms them into arguments that will be passed to the request. E.g. 'id=10' or 'name="Best Mod"'

not_equals (***kwargs*)

Where the preceding column value does not equal the value specified. There are not set parameters, this methods takes any named keywords and transforms them into arguments that will be passed to the request. E.g. 'id=10' or 'name="Best Mod"'

like (***kwargs*)

Where the string supplied matches the preceding column value. This is equivalent to SQL's LIKE. Consider using wildcard's * for the best chance of results as described below. There are not set parameters, this methods takes any named keywords and transforms them into arguments that will be passed to the request. E.g. 'id=10' or 'name="Best Mod"'

not_like (***kwargs*)

Where the string supplied does not match the preceding column value. This is equivalent to SQL's NOT LIKE. This is equivalent to SQL's LIKE. Consider using wildcard's * for the best chance of results as described below. There are not set parameters, this methods takes any named keywords and transforms them into arguments that will be passed to the request. E.g. 'id=10' or 'name="Best Mod"'

values_in (***kwargs*)

Where the supplied list of values appears in the preceding column value. This is equivalent to SQL's IN. There are not set parameters, this methods takes any named keywords and values as lists and transforms them into arguments that will be passed to the request. E.g. 'id=[10, 3, 4]' or 'name=["Best","Mod"]'

values_not_in (***kwargs*)

Where the supplied list of values does NOT appears in the preceding column value. This is equivalent to SQL's NOT IN. There are not set parameters, this methods takes any named keywords and values as lists and transforms them into arguments that will be passed to the request. E.g. 'id=[10, 3, 4]' or 'name=["Best","Mod"]'

max (***kwargs*)

Where the preceding column value is smaller than or equal to the value specified. There are not set parameters, this methods takes any named keywords and transforms them into arguments that will be passed to the request. E.g. 'game_id=40'

min (***kwargs*)

Where the preceding column value is greater than or equal to the value specified. There are not set parameters, this methods takes any named keywords and transforms them into arguments that will be passed to the request. E.g. 'game_id=40'

smaller_than (***kwargs*)

Where the preceding column value is smaller than the value specified. There are not set parameters, this methods takes any named keywords and transforms them into arguments that will be passed to the request. E.g. 'game_id=40'

greater_than (***kwargs*)

Where the preceding column value is greater than the value specified. There are not set parameters, this methods takes any named keywords and transforms them into arguments that will be passed to the request. E.g. 'game_id=40'

bitwise (***kwargs*)

Some columns are stored as bits within an integer. You can combine any number of options for the column of the object you are querying. This is dependent on which item is being queried. These can be added together to check for multiple options at once. E.g if Option A: 1 and Option B: 2 then submitting 3 will return items that have both option A and B enabled.

sort (*key*, *, *reverse=False*)

Allows you to sort the results by the value of a top level column with a single value.

Parameters

- **key** (*str*) – The column by which to sort the results
- **reverse** (*Optional[bool]*) – Optional, defaults to False. Whether to sort by ascending (False) or descending (True) order.

limit (*limit*)

Allows to limit the amount of results returned per query.

Parameters **limit** (*int*) – Limit of returned results for the query

offset (*offset*)

Allows to offset the first result by a certain amount.

Parameters `offset` (*int*) – The number of results to skip.

get_dict ()

Utility methods to get all filters while omitting None values

Returns The dict of filters

Return type Dict[str, Union[str, int]]

class modio.objects.Pagination (***attrs*)

This class is unique to the library and represents the pagination data that some of the endpoints return.

count

Number of results returned by the request.

Type int

limit

Maximum number of results returned.

Type int

offset

Number of results skipped over

Type int

total

Total number of results available for that endpoint with those filters.

Type int

max ()

Returns True if there are no additional results after this set.

min ()

Returns True if there are no additional results before this set.

next ()

Returns the offset required for the next set of results. If the max results have been reached this returns the current offset.

previous ()

Returns the offset required for the previous set of results. If the min results have been reached this returns the current offset.

page ()

Returns the current page number. Page numbers start at 0

class modio.objects.Returned

A named tuple returned by certain methods which return multiple results and need to return pagination data along with it.

results

The list of results returned. This is typed accordingly to the method that returns it.

Type List[Result]

pagination

Pagination metadata attached to the results

Type *Pagination*

results

Alias for field number 0

pagination

Alias for field number 1

count()

Return number of occurrences of value.

index()

Return first index of value.

Raises ValueError if the value is not present.

class modio.objects.Object (**attrs)

A dud class that can be used to replace other classes, keyword arguments passed will become attributes.

5.7 Filtering, Sorting and Pagination

This library supports the filtering and sorting under the form of the Filter object which can be instantiated and edited in order to fine tune the returned results. First instantiate the filter with or without parameters and then call any of the various help methods attached to the object to add additional paramaters. In addition to filtering, the object supports three sorting methods: sort, offset and limit. All three are explained in depth in the Filter section of the docs.

```
import modio

client = modio.Client(api_key="api key goes here")
filters = modio.Filter()

filters.text("The Lord of the Rings")
#This will return every result where the name column contains any of
#the following words: 'The', 'Lord', 'of', 'the', 'Rings'

filters.equal(id=10)
# Get all results where the id column value is 10.

filters.like(name="The Witcher*")
#Get all results where 'The Witcher' is succeeded by any value

filters.not_like(name="*Asset Pack")
#Get all results where Asset Pack NOT is proceeded by any value.

filters.values_in(id=[3,11,16,29])
#Get all results where the id column value is 3, 11, 16 and 29.

filters.sort("name")
#Sort name in ascending order

filters.sort("id", reverse=True)
#Sort id in descending order

filters.limit(20)
#limit to 20 results

filters.offset(5)
#skip the first five results

games, pagination_metadata = client.get_games(filters=filters)
#returns all the result that meet the above criteria
```

In addition, this library also supports and extends the pagination metadata provided by modio in the form of the `Pagination` object. The pagination object can be used both to gather additional data on the pagination, such as if you've reached the last page, or what page you are on. In addition, it can be passed to the `Filter.offset()` of the `Filter` instance you used to obtain the results to get the next page of results easily by simply passing the edited filter instance. For example if we want to get the next page of results we can simply do:

```
import modio

client = modio.Client(api_key="api key goes here")
filters = modio.Filter()
filters.text("The Lord of the Rings")
games, pagination = client.get_games(filters=filters)

filters.offset(pagination.next_page())
games, pagination = client.get_games(filters=filters)
```

5.8 Asynchronous mod.io

Most blocking requests in this library have an async equivalent which can be accessed by simply prefixing a method with `async_`. Methods with an async equivalent will be labelled as such with:

This method has an async equivalent prefixed with `'async_'`. You must use `Client.start` before using the async equivalent.

Certain methods are also exclusively async, these methods will be labelled with:

This method has no sync equivalent. You must use `Client.start` before using this method

5.8.1 Basic Usage

```
import modio
import asyncio

async def example():
    client = modio.Client(api_key="your api key here", access_token="your o auth 2_
↪token here")
    await client.start() # this is essential to instance the async sessions

    game = await client.get_game(345)
    #gets the game with id 345

    print(game.name)
    #prints the name of the game

    mod = await game.get_mod(231)
    #gets the mod for that game with id 231

    await client.close()
    #cleans up the client to gracefully shut down, client will have to be
    #re started if other queries are to be made

def main():
    loop = asyncio.get_event_loop()
    loop.run_until_complete(example())
```

(continues on next page)

(continued from previous page)

```
loop.close()

if __name__ == '__main__':
    main()
```

5.8.2 Getting an OAuth 2 Access Token

To perform writes, you will need to authenticate your users via OAuth 2. To make this easy this library provides you with two functions to use in order to obtain your Access Token. You will need an API Key and an email adress to which you have access in order for this to work. Once you have both, follow the example below, you can either run this in a REPL or as a Python script. Don't forget to edit the script to add your own api key and email adress.

5.8.3 Example

```
import modio
import asyncio

async def auth():
    client = modio.Client(api_key="your api key here")
    client.start()

    #request a security code be sent at this email address
    await client.email_request("necro@mordor.com")

    #check your email for the security code
    code = input("Code: ")

    oauth2 = await client.email_exchange(code)

    #your oauth2 token is now stored in the variable

    #to save simply
    with open("oauth2.txt", "w") as f:
        f.write(oauth2)

    #and now the token is stored in oauth2.txt

def main():
    loop = asyncio.get_event_loop()
    loop.run_until_complete(auth())
    loop.close()

if __name__ == '__main__':
    main()
```

5.9 Utility Functions

Utility functions for the library

```
modio.utils.concat_docs(cls)
    Does it look like I'm enjoying this?
```

`modio.utils.find(iterable, **fields)`

Finds the first item in the :attrs: iterable that has the :attrs: attr equal to :attrs: value. For example:

```
game = find(client.get_all_games(), id=2)
```

would find the first :class: Game whose id is 2 and return it. If no entry is found then None is returned.

```
game = find(client.get_all_games(), name="John")
```

would find the first :class: Game whose name is 'John'. If not entry is found then None is returned

`modio.utils.get(iterable, **fields)`

Returns a list of items in the :attrs: iterable that have the :attrs: attr equal to :attrs: value. For example:

```
game = get(client.get_all_games(), id=2)
```

would find the all :class: Game whose id is 2 and return them as a list. If no entry is found then the empty list is returned.

```
game = find(client.get_all_games(), name="John")
```

would find all :class: Game whose name is 'John'. If not entry is found then an empty list is returned

`modio.utils.ratelimit_retry(max_retries)`

`modio.utils.async_ratelimit_retry(max_retries)`

5.10 Enumerators

Modio enums as defined by the API

class `modio.enums.IntFlagMixin`

Mixin class for IntFlags containing formatting methods.

class `modio.enums.TargetPlatform`

Enums for different type of target platforms

```
windows = 'Windows'
```

```
mac = 'Mac'
```

```
linux = 'Linux'
```

```
android = 'Android'
```

```
ios = 'iOS'
```

```
xboxone = 'XboxOne'
```

```
xboxseriesx = 'XboxSeriesX'
```

```
ps4 = 'PS4'
```

```
ps5 = 'PS5'
```

```
switch = 'Switch'
```

```
oculus = 'Oculus'
```

```
source = 'Source'
```

class `modio.enums.TargetPortal`

Enums for different type of target portals

```
apple = 'Apply'
```

```
discord = 'Discord'
epic = 'EGS'
facebook = 'Facebook'
gog = 'GOG'
google = 'Google'
itchio = 'Itchio'
nintendo = 'Nintendo'
openid = 'OpenID'
psn = 'PSN'
steam = 'Steam'
xboxlive = 'XBoxLive'

class modio.enums.Status
    Status of the game. 0 : Not accepted 1 : Accepted (default) 2 : Archived (default) 3 : Deleted

    not_accepted = 0
    accepted = 1
    archived = 2
    deleted = 3

class modio.enums.ModFilePlatformStatus
    Status of a modfile for the specific platform.

    0 : Pending 1 : Accepted 2 : Denied

    pending = 0
    accepted = 1
    denied = 2

class modio.enums.Presentation
    0 : Display mods for that game in a grid on mod.io 1 : Display mods for that game in a table on mod.io

    grid = 0
    table = 1

class modio.enums.Submission
    0 : Mod uploads must occur via a tool created by the game developers 1 : Mod uploads can occur from anywhere,
    including the website and API

    restricted = 0
    unrestricted = 1

class modio.enums.Curation
    0 : No curation: Mods are immediately available to play 1 : Paid curation: Mods are immediately available to
    play unless they choose to receive donations. These mods must be accepted to be listed 2 : Full curation: All
    mods must be accepted by someone to be listed

    no_curation = 0
    paid_curation = 1
    full_curation = 2
```



```
class modio.enums.Community
```

0 : All of the options below are disabled 1 : Discussion board enabled 2 : Guides and news enabled ? : Above options can be added together to create custom settings (e.g 3 : discussion board, guides and news enabled)

```
disabled = 0
```

```
discussion_boards = 1
```

```
guides_news = 2
```

```
class modio.enums.Revenue
```

0 : All of the options below are disabled 1 : Allow mods to be sold 2 : Allow mods to receive donations 4 : Allow mods to be traded 8 : Allow mods to control supply and scarcity ? : Above options can be added together to create custom settings (e.g 3 : allow mods to be sold and receive donations)

```
disabled = 0
```

```
sold = 1
```

```
donations = 2
```

```
traded = 4
```

```
full_control = 8
```

```
class modio.enums.APIAccess
```

0 : All of the options below are disabled 1 : Allow 3rd parties to access this games API endpoints 2 : Allow mods to be downloaded directly (if disabled all download URLs will contain a frequently changing verification hash to stop unauthorized use) ? : Above options can be added together to create custom settings (e.g 3 : allow 3rd parties to access this games API endpoints and allow mods to be downloaded directly)

```
disabled = 0
```

```
third_party = 1
```

```
direct_downloads = 2
```

```
class modio.enums.MaturityOptions
```

0 [Don't allow mod developpers to decide whether or not to flag their mod as] containing mature content (if game devs wish to handle it)

1 [Allow mod developpers to decide whether or not to flag their mod as] containing mature content

```
forbidden = 0
```

```
allowed = 1
```

```
class modio.enums.Maturity
```

0 : None 1 : Alcohol 2 : Drugs 4 : Violence 8 : Explicit ? : Above options can be added together to create custom settings (e.g 3 : alcohol and drugs present)

```
none = 0
```

```
alcohol = 1
```

```
drugs = 2
```

```
violence = 4
```

```
explicit = 8
```

```
class modio.enums.VirusStatus
```

0 : Not scanned 1 : Scan complete 2 : In progress 3 : Too large to scan 4 : File not found 5 : Error Scanning

```
not_scanned = 0
```

```
    scan_complete = 1
    in_progress = 2
    too_large = 3
    not_found = 4
    error = 5
class modio.enums.Visibility
    0 : Hidden 1 : Public
    hidden = 0
    public = 1
class modio.enums.Level
    Level of permission the user has. 1 : Moderator (can moderate comments and content attached) 4 : Manager (moderator access, including uploading builds and editing settings except supply and team members) 8 : Administrator (full access, including editing the supply and team)
    moderator = 1
    creator = 4
    admin = 8
class modio.enums.Report
    0 : Generic Report 1 : DMCA Report
    generic = 0
    dmca = 1
class modio.enums.EventType
    An enum to render all event types easy to compare.
    file_changed = 0
    available = 1
    unavailable = 2
    edited = 3
    deleted = 4
    team_changed = 5
    comment_added = 6
    comment_deleted = 7
    team_join = 8
    team_leave = 9
    subscribe = 10
    unsubscribe = 11
class modio.enums.RatingType
    The type of rating submitted (good, bad, neutral)
    good = 1
    neutral = 0
```

```
bad = -1
```

5.11 Exceptions

Errors generate by mod.io and the library.

exception `modio.errors.modioException (text, code=None, ref=None, errors=None)`

Base exception for the lib

code

The status code if this error was raised from a request

Type `Optional[int]`

ref

The ref error code provided by mod.io

Type `Optional[int]`

text

The unformatted text of the error

Type `str`

errors

The validation errors if any exist

Type `Optional[dict]`

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

5.12 Changelog

The page attempt to keep a clear list of breaking/non-breaking changes and new features made to the library.

Table of Contents

- *v0.6.0*
 - *New Features*
- *v0.5.0*
 - *New Features*
 - *Bugs Fixed*
- *v0.4.3*
 - *New Features*
 - *Bugs Fixed*
- *v0.4.2*
 - *New Features*

- *Bugs Fixed*
- *v0.4.1*
- *v0.4.0*
 - *New Features*
 - *Removed Features*
- *v0.3.1*
 - *New Features*
 - *Removed Features*

5.12.1 v0.6.0

New Features

- The behavior of the library when being ratelimited can now be customised with the *Client.ratelimit_max_sleep* parameter. By default this is set to infinity to keep the same behavior as before. More info in this section [Ratelimits and Retries](#)
- Added new platform type *TargetPlatform.source*

5.12.2 v0.5.0

This patch adds support for targetting platforms and portals

New Features

- New *platform* and *portal* parameters for *Client*
- New *Client.set_portal* and *Client.set_platform* methods
- New *TargetPortal* to represent portals
- Library will now retry any ratelimited requests once after sleeping

Bugs Fixed

- Fixed ratelimit sleep not being enforced properly

5.12.3 v0.4.3

New Features

- *Platform* object has been split into *GamePlatform*, *ModPlatform* and *ModFilePlatform* to better reflect the API models
- New *Mod.platforms* attribute
- *Game.platforms* is now a *List[GamePlatform]*, different class but same attributes

Bugs Fixed

- *Modfile.platforms* fixed, now a *List[ModFilePlatforms]* with correct attributes

5.12.4 v0.4.2

New Features

- *ModFile* now has a *platforms* attribute

Bugs Fixed

- *Game* now properly has a *platforms* attribute
- *Filter.max* no longer overflows

5.12.5 v0.4.1

Small dependency bugfix

5.12.6 v0.4.0

This patch focuses on making sure none of the new attributes of the mod.io API models slip through the cracks and that they are all being parsed and added to the correct library models.

New Features

- *Client.email_exchange* now supports *date_expire*
- New object *Platform*
- *Stats* renamed to *ModStats*, new *GameStats* object
- New enum *TargetPlatform*
- New attributes for Game: *stats*, *other_urls*, *platforms*
- *expires* attribute renamed to *date_expires*
- New methods *Comment.add_positive_karma* and *Comment.add_negative_karma* and async equivalents
- Added comment added/deleted event support
- *Game.get_stats* renamed to *Game.get_mods_stats*
- New function *Game.get_stats* that gets stats for the game rather than for the mods of the game
- New example *examples/polling_events* showing how to use the filter class to only get the latest attributes
- *Game.add_tag_options* now supports the *locked* option
- New attribute for TagOption: *locked*
- *Rating.mod* renamed to *Rating.mod_id*
- Library is now typed, making it easier to use with IDEs

Removed Features

- *Comment.mod* is now deprecated and removed, replaced with *Comment.resource_id*
- *Comment.karma_guest* is deprecated and has been removed

5.12.7 v0.3.1

This version of the library represents a major rework. The most important is the merge of the async and sync library. They now form a single library in which blocking methods have a async equivalent with the same name but prefixed with *async_*

New Features

- Ratelimits are now enforced by the library
- *filter* parameters of functions renamed to *filters*
- *Mod.game* and *ModFile.game* renamed to *game_id*
- Muting/unmuting users and getting mutes now supported
- Editing/adding/deleting comments now supported
- *Game.submitter* is now optional
- Many methods that used to take *id* now take *{entity}_id* where {entity} is something like *mod* or *game*
- Entities no longer update themselves but rather return the updated entity where possible.

Removed Features

- Many of exceptions have been removed, the library now uses the base exception for most errors
- Removed the account links support, looking into a better implementation
- Many removed endpoints have had their method also removed

CHAPTER 6

Indices and tables

- `genindex`
- `modindex`
- `search`

m

- `modio.client`, [11](#)
- `modio.entities`, [31](#)
- `modio.enums`, [51](#)
- `modio.errors`, [55](#)
- `modio.game`, [16](#)
- `modio.mod`, [22](#)
- `modio.objects`, [44](#)
- `modio.utils`, [50](#)

A

- accepted (*modio.enums.ModFilePlatformStatus* attribute), 52
- accepted (*modio.enums.Status* attribute), 52
- add_comment() (*modio.mod.Mod* method), 25
- add_dependencies() (*modio.mod.Mod* method), 30
- add_file() (*modio.mod.Mod* method), 26
- add_file() (*modio.objects.NewModFile* method), 45
- add_media() (*modio.game.Game* method), 20
- add_media() (*modio.mod.Mod* method), 27
- add_metadata() (*modio.mod.Mod* method), 28
- add_mod() (*modio.game.Game* method), 19
- add_negative_karma() (*modio.entities.Comment* method), 33
- add_negative_rating() (*modio.entities.Rating* method), 38
- add_negative_rating() (*modio.mod.Mod* method), 28
- add_positive_karma() (*modio.entities.Comment* method), 33
- add_positive_rating() (*modio.entities.Rating* method), 38
- add_positive_rating() (*modio.mod.Mod* method), 28
- add_tag_options() (*modio.game.Game* method), 20
- add_tags() (*modio.mod.Mod* method), 27
- add_tags() (*modio.objects.NewMod* method), 44
- add_team_member() (*modio.mod.Mod* method), 30
- admin (*modio.enums.Level* attribute), 54
- alcohol (*modio.enums.Maturity* attribute), 53
- allowed (*modio.enums.MaturityOptions* attribute), 53
- android (*modio.enums.TargetPlatform* attribute), 51
- api (*modio.game.Game* attribute), 17
- APIAccess (class in *modio.enums*), 53
- apple (*modio.enums.TargetPortal* attribute), 51
- archived (*modio.enums.Status* attribute), 52
- args (*modio.errors.modioException* attribute), 55
- async_add_comment() (*modio.mod.Mod* method), 28
- async_add_dependencies() (*modio.mod.Mod* method), 28
- async_add_file() (*modio.mod.Mod* method), 28
- async_add_media() (*modio.game.Game* method), 20
- async_add_media() (*modio.mod.Mod* method), 28
- async_add_metadata() (*modio.mod.Mod* method), 29
- async_add_mod() (*modio.game.Game* method), 20
- async_add_negative_karma() (*modio.entities.Comment* method), 34
- async_add_negative_rating() (*modio.entities.Rating* method), 38
- async_add_negative_rating() (*modio.mod.Mod* method), 29
- async_add_positive_karma() (*modio.entities.Comment* method), 34
- async_add_positive_rating() (*modio.entities.Rating* method), 38
- async_add_positive_rating() (*modio.mod.Mod* method), 29
- async_add_tag_options() (*modio.game.Game* method), 20
- async_add_tags() (*modio.mod.Mod* method), 29
- async_add_team_member() (*modio.mod.Mod* method), 29
- async_delete() (*modio.entities.Comment* method), 34
- async_delete() (*modio.entities.ModFile* method), 36
- async_delete() (*modio.entities.Rating* method), 38
- async_delete() (*modio.mod.Mod* method), 29
- async_delete_dependencies() (*modio.mod.Mod* method), 29
- async_delete_media() (*modio.mod.Mod* method), 29
- async_delete_metadata() (*modio.mod.Mod* method), 29
- async_delete_tag_options()

(modio.game.Game method), 21

async_delete_tags() (*modio.mod.Mod method*), 29

async_edit() (*modio.entities.Comment method*), 34

async_edit() (*modio.entities.ModFile method*), 36

async_edit() (*modio.mod.Mod method*), 29

async_email_exchange() (*modio.client.Client method*), 14

async_email_request() (*modio.client.Client method*), 14

async_get_comments() (*modio.mod.Mod method*), 29

async_get_dependencies() (*modio.mod.Mod method*), 29

async_get_events() (*modio.mod.Mod method*), 29

async_get_file() (*modio.mod.Mod method*), 29

async_get_files() (*modio.mod.Mod method*), 29

async_get_game() (*modio.client.Client method*), 14

async_get_games() (*modio.client.Client method*), 15

async_get_metadata() (*modio.mod.Mod method*), 29

async_get_mod() (*modio.game.Game method*), 21

async_get_mod_events() (*modio.game.Game method*), 21

async_get_mods() (*modio.game.Game method*), 21

async_get_mods_stats() (*modio.game.Game method*), 21

async_get_my_events() (*modio.client.Client method*), 15

async_get_my_games() (*modio.client.Client method*), 15

async_get_my_modfiles() (*modio.client.Client method*), 15

async_get_my_mods() (*modio.client.Client method*), 15

async_get_my_mutes() (*modio.client.Client method*), 15

async_get_my_ratings() (*modio.client.Client method*), 15

async_get_my_subs() (*modio.client.Client method*), 15

async_get_my_user() (*modio.client.Client method*), 15

async_get_owner() (*modio.entities.ModFile method*), 36

async_get_owner() (*modio.game.Game method*), 21

async_get_owner() (*modio.mod.Mod method*), 29

async_get_stats() (*modio.game.Game method*), 21

async_get_stats() (*modio.mod.Mod method*), 29

async_get_tag_options() (*modio.game.Game method*), 21

async_get_tags() (*modio.mod.Mod method*), 29

async_get_team() (*modio.mod.Mod method*), 29

async_mute() (*modio.entities.TeamMember method*), 43

async_mute() (*modio.entities.User method*), 42

async_ratelimit_retry() (in module *modio.utils*), 51

async_report() (*modio.entities.TeamMember method*), 43

async_report() (*modio.entities.User method*), 42

async_report() (*modio.game.Game method*), 21

async_report() (*modio.mod.Mod method*), 29

async_subscribe() (*modio.mod.Mod method*), 29

async_unmute() (*modio.entities.TeamMember method*), 43

async_unmute() (*modio.entities.User method*), 42

async_unsubscribe() (*modio.mod.Mod method*), 29

available (*modio.enums.EventType attribute*), 54

avatar (*modio.entities.TeamMember attribute*), 43

avatar (*modio.entities.User attribute*), 41

B

bad (*modio.enums.RatingType attribute*), 54

BasePlatform (class in *modio.entities*), 36

bitwise() (*modio.objects.Filter method*), 46

C

changelog (*modio.entities.ModFile attribute*), 35

children (*modio.entities.Comment attribute*), 33

Client (class in *modio.client*), 11

close() (*modio.client.Client method*), 15

code (*modio.entities.Message attribute*), 31

code (*modio.errors.modioException attribute*), 55

Comment (class in *modio.entities*), 32

comment_added (*modio.enums.EventType attribute*), 54

comment_deleted (*modio.enums.EventType attribute*), 54

Community (class in *modio.enums*), 52

community (*modio.game.Game attribute*), 17

concat_docs() (in module *modio.utils*), 50

content (*modio.entities.Comment attribute*), 33

count (*modio.objects.Pagination attribute*), 47

count() (*modio.objects.Returned method*), 48

creator (*modio.enums.Level attribute*), 54

Curation (class in *modio.enums*), 52

curation (*modio.game.Game attribute*), 17

D

danger (*modio.entities.Theme attribute*), 40

dark (*modio.entities.Theme attribute*), 40

date (*modio.entities.Comment attribute*), 32

date (*modio.entities.Event attribute*), 32

date (*modio.entities.ModFile attribute*), 34
 date (*modio.entities.Rating attribute*), 38
 date (*modio.entities.TeamMember attribute*), 43
 date (*modio.game.Game attribute*), 16
 date (*modio.mod.Mod attribute*), 22
 date_expires (*modio.entities.GameStats attribute*), 40
 date_expires (*modio.entities.ModFile attribute*), 35
 date_expires (*modio.entities.ModStats attribute*), 39
 delete() (*modio.entities.Comment method*), 33
 delete() (*modio.entities.ModFile method*), 35
 delete() (*modio.entities.Rating method*), 38
 delete() (*modio.mod.Mod method*), 26
 delete_dependencies() (*modio.mod.Mod method*), 30
 delete_media() (*modio.mod.Mod method*), 27
 delete_metadata() (*modio.mod.Mod method*), 29
 delete_tag_options() (*modio.game.Game method*), 21
 delete_tags() (*modio.mod.Mod method*), 28
 deleted (*modio.enums.EventType attribute*), 54
 deleted (*modio.enums.Status attribute*), 52
 denied (*modio.enums.ModFilePlatformStatus attribute*), 52
 Dependencies (*class in modio.entities*), 41
 description (*modio.mod.Mod attribute*), 23
 direct_downloads (*modio.enums.APIAccess attribute*), 53
 disabled (*modio.enums.APIAccess attribute*), 53
 disabled (*modio.enums.Community attribute*), 53
 disabled (*modio.enums.Revenue attribute*), 53
 discord (*modio.enums.TargetPortal attribute*), 51
 discussion_boards (*modio.enums.Community attribute*), 53
 dmca (*modio.enums.Report attribute*), 54
 donations (*modio.enums.Revenue attribute*), 53
 downloads (*modio.entities.ModStats attribute*), 38
 drugs (*modio.enums.Maturity attribute*), 53

E

edit() (*modio.entities.Comment method*), 33
 edit() (*modio.entities.ModFile method*), 35
 edit() (*modio.mod.Mod method*), 26
 edited (*modio.enums.EventType attribute*), 54
 email_exchange() (*modio.client.Client method*), 16
 email_request() (*modio.client.Client method*), 15
 epic (*modio.enums.TargetPortal attribute*), 52
 equals() (*modio.objects.Filter method*), 45
 error (*modio.enums.VirusStatus attribute*), 54
 errors (*modio.errors.modioException attribute*), 55
 Event (*class in modio.entities*), 31
 EventType (*class in modio.enums*), 54
 explicit (*modio.enums.Maturity attribute*), 53

F

facebook (*modio.enums.TargetPortal attribute*), 52
 file (*modio.mod.Mod attribute*), 23
 file_changed (*modio.enums.EventType attribute*), 54
 filename (*modio.entities.Image attribute*), 31
 filename (*modio.entities.ModFile attribute*), 35
 Filter (*class in modio.objects*), 45
 find() (*in module modio.utils*), 50
 forbidden (*modio.enums.MaturityOptions attribute*), 53
 full_control (*modio.enums.Revenue attribute*), 53
 full_curation (*modio.enums.Curation attribute*), 52

G

Game (*class in modio.game*), 16
 game_id (*modio.entities.Event attribute*), 32
 game_id (*modio.entities.ModFile attribute*), 35
 game_id (*modio.entities.Rating attribute*), 37
 game_id (*modio.mod.Mod attribute*), 22
 GamePlatform (*class in modio.entities*), 36
 GameStats (*class in modio.entities*), 39
 generic (*modio.enums.Report attribute*), 54
 get() (*in module modio.utils*), 51
 get_comments() (*modio.mod.Mod method*), 25
 get_dependencies() (*modio.mod.Mod method*), 25
 get_dict() (*modio.objects.Filter method*), 47
 get_events() (*modio.mod.Mod method*), 24
 get_file() (*modio.mod.Mod method*), 24
 get_files() (*modio.mod.Mod method*), 24
 get_game() (*modio.client.Client method*), 12
 get_games() (*modio.client.Client method*), 13
 get_metadata() (*modio.mod.Mod method*), 24
 get_mod() (*modio.game.Game method*), 18
 get_mod_events() (*modio.game.Game method*), 18
 get_mods() (*modio.game.Game method*), 18
 get_mods_stats() (*modio.game.Game method*), 19
 get_my_events() (*modio.client.Client method*), 13
 get_my_games() (*modio.client.Client method*), 13
 get_my_modfiles() (*modio.client.Client method*), 14
 get_my_mods() (*modio.client.Client method*), 14
 get_my_mutes() (*modio.client.Client method*), 15
 get_my_ratings() (*modio.client.Client method*), 14
 get_my_subs() (*modio.client.Client method*), 13
 get_my_user() (*modio.client.Client method*), 13
 get_owner() (*modio.entities.ModFile method*), 36
 get_owner() (*modio.game.Game method*), 21
 get_owner() (*modio.mod.Mod method*), 30
 get_stats() (*modio.game.Game method*), 19
 get_stats() (*modio.mod.Mod method*), 26
 get_tag_options() (*modio.game.Game method*), 19
 get_tags() (*modio.mod.Mod method*), 24
 get_team() (*modio.mod.Mod method*), 25

gog (*modio.enums.TargetPortal* attribute), 52
good (*modio.enums.RatingType* attribute), 54
google (*modio.enums.TargetPortal* attribute), 52
greater_than() (*modio.objects.Filter* method), 46
grid (*modio.enums.Presentation* attribute), 52
guides_news (*modio.enums.Community* attribute), 53

H

hash (*modio.entities.ModFile* attribute), 34
header (*modio.game.Game* attribute), 17
hidden (*modio.entities.TagOption* attribute), 37
hidden (*modio.enums.Visibility* attribute), 54
homepage (*modio.mod.Mod* attribute), 23

I

icon (*modio.game.Game* attribute), 17
id (*modio.entities.Comment* attribute), 32
id (*modio.entities.Event* attribute), 32
id (*modio.entities.GameStats* attribute), 39
id (*modio.entities.ModFile* attribute), 34
id (*modio.entities.ModStats* attribute), 38
id (*modio.entities.TeamMember* attribute), 42
id (*modio.entities.User* attribute), 41
id (*modio.game.Game* attribute), 16
id (*modio.mod.Mod* attribute), 22
Image (*class in modio.entities*), 31
images (*modio.entities.ModMedia* attribute), 36
in_progress (*modio.enums.VirusStatus* attribute), 54
index() (*modio.objects.Returned* method), 48
instructions (*modio.game.Game* attribute), 18
instructions_url (*modio.game.Game* attribute), 18
IntFlagMixin (*class in modio.enums*), 51
ios (*modio.enums.TargetPlatform* attribute), 51
is_stale() (*modio.entities.GameStats* method), 40
is_stale() (*modio.entities.ModStats* method), 39
itchio (*modio.enums.TargetPortal* attribute), 52

K

karma (*modio.entities.Comment* attribute), 33
karma_guest (*modio.entities.Comment* attribute), 33
kvp (*modio.mod.Mod* attribute), 23, 24

L

label (*modio.entities.GamePlatform* attribute), 36
lang (*modio.entities.TeamMember* attribute), 43
lang (*modio.entities.User* attribute), 41
large (*modio.entities.Image* attribute), 31
last_online (*modio.entities.TeamMember* attribute), 42
last_online (*modio.entities.User* attribute), 41
Level (*class in modio.enums*), 54
level (*modio.entities.Comment* attribute), 33

level (*modio.entities.TeamMember* attribute), 43
light (*modio.entities.Theme* attribute), 40
like() (*modio.objects.Filter* method), 45
limit (*modio.objects.Pagination* attribute), 47
limit() (*modio.objects.Filter* method), 46
linux (*modio.enums.TargetPlatform* attribute), 51
live (*modio.game.Game* attribute), 16
live (*modio.mod.Mod* attribute), 22
locked (*modio.entities.TagOption* attribute), 37
logo (*modio.game.Game* attribute), 17
logo (*modio.mod.Mod* attribute), 22

M

mac (*modio.enums.TargetPlatform* attribute), 51
Maturity (*class in modio.enums*), 53
maturity (*modio.mod.Mod* attribute), 23
maturity_options (*modio.game.Game* attribute), 17
MaturityOptions (*class in modio.enums*), 53
max() (*modio.objects.Filter* method), 46
max() (*modio.objects.Pagination* method), 47
media (*modio.mod.Mod* attribute), 23
medium (*modio.entities.Image* attribute), 31
Message (*class in modio.entities*), 31
message (*modio.entities.Message* attribute), 31
MetaData (*class in modio.entities*), 41
metadata (*modio.entities.ModFile* attribute), 35
metadata (*modio.mod.Mod* attribute), 23
min() (*modio.objects.Filter* method), 46
min() (*modio.objects.Pagination* method), 47
Mod (*class in modio.mod*), 22
mod (*modio.entities.Event* attribute), 32
mod (*modio.entities.ModFile* attribute), 34
mod (*modio.entities.TeamMember* attribute), 43
mod_id (*modio.entities.Rating* attribute), 37
mod_key (*modio.entities.Rating* attribute), 38
mod_key (*modio.mod.Mod* attribute), 24
moderated (*modio.entities.GamePlatform* attribute), 36
moderator (*modio.enums.Level* attribute), 54
ModFile (*class in modio.entities*), 34
modfile_live (*modio.entities.ModPlatform* attribute), 37
ModFilePlatform (*class in modio.entities*), 37
ModFilePlatformStatus (*class in modio.enums*), 52
modio.client (*module*), 11
modio.entities (*module*), 31
modio.enums (*module*), 51
modio.errors (*module*), 55
modio.game (*module*), 16
modio.mod (*module*), 22
modio.objects (*module*), 44
modio.utils (*module*), 50

modioException, 55
 ModMedia (class in modio.entities), 36
 ModPlatform (class in modio.entities), 37
 mods_count_total (modio.entities.GameStats attribute), 39
 mods_download_daily_avg (modio.entities.GameStats attribute), 40
 mods_download_today (modio.entities.GameStats attribute), 39
 mods_download_total (modio.entities.GameStats attribute), 40
 mods_subscribers_total (modio.entities.GameStats attribute), 40
 ModStats (class in modio.entities), 38
 mute() (modio.entities.TeamMember method), 43
 mute() (modio.entities.User method), 42

N

name (modio.entities.TagOption attribute), 37
 name (modio.game.Game attribute), 17
 name (modio.mod.Mod attribute), 23
 name_id (modio.entities.TeamMember attribute), 42
 name_id (modio.entities.User attribute), 41
 name_id (modio.game.Game attribute), 17
 name_id (modio.mod.Mod attribute), 23
 negative (modio.entities.ModStats attribute), 39
 neutral (modio.enums.RatingType attribute), 54
 NewMod (class in modio.objects), 44
 NewModFile (class in modio.objects), 44
 next() (modio.objects.Pagination method), 47
 nintendo (modio.enums.TargetPortal attribute), 52
 no_curation (modio.enums.Curation attribute), 52
 none (modio.enums.Maturity attribute), 53
 not_accepted (modio.enums.Status attribute), 52
 not_equals() (modio.objects.Filter method), 45
 not_found (modio.enums.VirusStatus attribute), 54
 not_like() (modio.objects.Filter method), 45
 not_scanned (modio.enums.VirusStatus attribute), 53

O

Object (class in modio.objects), 48
 oculus (modio.enums.TargetPlatform attribute), 51
 offset (modio.objects.Pagination attribute), 47
 offset() (modio.objects.Filter method), 46
 openid (modio.enums.TargetPortal attribute), 52
 original (modio.entities.Image attribute), 31
 other_urls (modio.game.Game attribute), 18

P

page() (modio.objects.Pagination method), 47
 Pagination (class in modio.objects), 47
 pagination (modio.objects.Returned attribute), 47
 paid_curation (modio.enums.Curation attribute), 52
 parent_id (modio.entities.Comment attribute), 32

pending (modio.enums.ModFilePlatformStatus attribute), 52
 percentage (modio.entities.ModStats attribute), 39
 plaintext (modio.mod.Mod attribute), 23
 platform (modio.entities.GamePlatform attribute), 36
 platform (modio.entities.ModFilePlatform attribute), 37
 platform (modio.entities.ModPlatform attribute), 37
 platforms (modio.entities.ModFile attribute), 35
 platforms (modio.game.Game attribute), 18
 position (modio.entities.Comment attribute), 33
 position (modio.entities.TeamMember attribute), 43
 positive (modio.entities.ModStats attribute), 39
 Presentation (class in modio.enums), 52
 presentation (modio.game.Game attribute), 16
 previous() (modio.objects.Pagination method), 47
 primary (modio.entities.Theme attribute), 40
 profile (modio.entities.TeamMember attribute), 43
 profile (modio.entities.User attribute), 41
 profile (modio.game.Game attribute), 18
 profile (modio.mod.Mod attribute), 23
 ps4 (modio.enums.TargetPlatform attribute), 51
 ps5 (modio.enums.TargetPlatform attribute), 51
 psn (modio.enums.TargetPortal attribute), 52
 public (modio.enums.Visibility attribute), 54

R

rank (modio.entities.ModStats attribute), 38
 rank_total (modio.entities.ModStats attribute), 38
 rate_limit (modio.client.Client attribute), 12
 rate_remain (modio.client.Client attribute), 12
 ratelimit_retry() (in module modio.utils), 51
 Rating (class in modio.entities), 37
 rating (modio.entities.Rating attribute), 38
 RatingType (class in modio.enums), 54
 ref (modio.errors.modioException attribute), 55
 Report (class in modio.enums), 54
 report() (modio.entities.TeamMember method), 43
 report() (modio.entities.User method), 42
 report() (modio.game.Game method), 21
 report() (modio.mod.Mod method), 30
 resource_id (modio.entities.Comment attribute), 32
 restricted (modio.enums.Submission attribute), 52
 results (modio.objects.Returned attribute), 47
 retry_after (modio.client.Client attribute), 12
 Returned (class in modio.objects), 47
 Revenue (class in modio.enums), 53
 revenue (modio.game.Game attribute), 17

S

scan_complete (modio.enums.VirusStatus attribute), 53
 scanned (modio.entities.ModFile attribute), 34
 set_platform() (modio.client.Client method), 12

set_portal() (*modio.client.Client* method), 12
 size (*modio.entities.ModFile* attribute), 34
 sketchfab (*modio.entities.ModMedia* attribute), 36
 small (*modio.entities.Image* attribute), 31
 smaller_than() (*modio.objects.Filter* method), 46
 sold (*modio.enums.Revenue* attribute), 53
 sort() (*modio.objects.Filter* method), 46
 source (*modio.enums.TargetPlatform* attribute), 51
 start() (*modio.client.Client* method), 15
 stats (*modio.game.Game* attribute), 18
 stats (*modio.mod.Mod* attribute), 23
 Status (class in *modio.enums*), 52
 status (*modio.entities.ModFilePlatform* attribute), 37
 status (*modio.game.Game* attribute), 16
 status (*modio.mod.Mod* attribute), 22
 steam (*modio.enums.TargetPortal* attribute), 52
 Submission (class in *modio.enums*), 52
 submission (*modio.game.Game* attribute), 17
 submitter (*modio.game.Game* attribute), 16
 submitter (*modio.mod.Mod* attribute), 22
 subscribe (*modio.enums.EventType* attribute), 54
 subscribe() (*modio.mod.Mod* method), 27
 subscribers (*modio.entities.ModStats* attribute), 39
 success (*modio.entities.Theme* attribute), 40
 summary (*modio.game.Game* attribute), 17
 summary (*modio.mod.Mod* attribute), 23
 switch (*modio.enums.TargetPlatform* attribute), 51

T

table (*modio.enums.Presentation* attribute), 52
 Tag (class in *modio.entities*), 40
 tag_options (*modio.game.Game* attribute), 18
 TagOption (class in *modio.entities*), 37
 tags (*modio.entities.TagOption* attribute), 37
 tags (*modio.mod.Mod* attribute), 23
 TargetPlatform (class in *modio.enums*), 51
 TargetPortal (class in *modio.enums*), 51
 team_changed (*modio.enums.EventType* attribute), 54
 team_id (*modio.entities.TeamMember* attribute), 43
 team_join (*modio.enums.EventType* attribute), 54
 team_leave (*modio.enums.EventType* attribute), 54
 TeamMember (class in *modio.entities*), 42
 text (*modio.entities.ModStats* attribute), 39
 text (*modio.errors.modioException* attribute), 55
 text() (*modio.objects.Filter* method), 45
 Theme (class in *modio.entities*), 40
 third_party (*modio.enums.APIAccess* attribute), 53
 too_large (*modio.enums.VirusStatus* attribute), 54
 total (*modio.entities.ModStats* attribute), 39
 total (*modio.objects.Pagination* attribute), 47
 traded (*modio.enums.Revenue* attribute), 53
 type (*modio.entities.Event* attribute), 32
 type (*modio.entities.TagOption* attribute), 37
 tz (*modio.entities.TeamMember* attribute), 43

tz (*modio.entities.User* attribute), 41

U

ugc (*modio.game.Game* attribute), 17
 unavailable (*modio.enums.EventType* attribute), 54
 unmute() (*modio.entities.TeamMember* method), 44
 unmute() (*modio.entities.User* method), 42
 unrestricted (*modio.enums.Submission* attribute), 52
 unsubscribe (*modio.enums.EventType* attribute), 54
 unsubscribe() (*modio.mod.Mod* method), 27
 updated (*modio.game.Game* attribute), 16
 updated (*modio.mod.Mod* attribute), 22
 url (*modio.entities.ModFile* attribute), 35
 url_is_expired() (*modio.entities.ModFile* method), 36
 User (class in *modio.entities*), 41
 user (*modio.entities.Comment* attribute), 32
 user (*modio.entities.Event* attribute), 32
 username (*modio.entities.TeamMember* attribute), 42
 username (*modio.entities.User* attribute), 41

V

values_in() (*modio.objects.Filter* method), 46
 values_not_in() (*modio.objects.Filter* method), 46
 version (*modio.entities.ModFile* attribute), 35
 violence (*modio.enums.Maturity* attribute), 53
 virus (*modio.entities.ModFile* attribute), 34
 virus_hash (*modio.entities.ModFile* attribute), 34
 virus_status (*modio.entities.ModFile* attribute), 34
 VirusStatus (class in *modio.enums*), 53
 Visibility (class in *modio.enums*), 54
 visible (*modio.mod.Mod* attribute), 22

W

warning (*modio.entities.Theme* attribute), 40
 weighted (*modio.entities.ModStats* attribute), 39
 windows (*modio.enums.TargetPlatform* attribute), 51
 with_traceback() (*modio.errors.modioException* method), 55

X

xboxlive (*modio.enums.TargetPortal* attribute), 52
 xboxone (*modio.enums.TargetPlatform* attribute), 51
 xboxseriesx (*modio.enums.TargetPlatform* attribute), 51

Y

youtube (*modio.entities.ModMedia* attribute), 36