AsyncDex

Release 0.1

PythonCoderAS

May 08, 2021

GETTING STARTED:

AsyncDex is an aiohttp-based async client for the MangaDex API. It respects all ratelimit rules set by MangaDex. Support for authentication is included as well as for running in anonymous mode.

CHAPTER

ONE

QUICKSTART

Warning: All AsyncDex operations have to be done inside of an async context.

Create an instance of MangadexClient:

```
from asyncdex import MangadexClient

async def main():
    client = MangadexClient()
```

Make a request for a random manga:

```
response = await client.request("GET", "/manga/random")
json = await response.json()
print(json["data"]["id"])
```

Log in to your MangaDex account:

```
await client.login(username="yourusername", password="yourpassword")
# alternate method
client = MangadexClient(username="yourusername", password="yourpassword")
await client.login()
```

View your manga follows list:

```
response = await client.request("GET", "/user/follows/manga")
json = await response.json()
[print(item["data"]["id"]) for item in json["results"]]
```

CHAPTER

TWO

CHANGELOG

2.1 v0.1

The initial release of AsyncDex.

CHAPTER

THREE

ASYNCDEX API

This page contains all of the classes, attributes, and methods of the various parts of AsyncDex.

3.1 Client

class asyncdex.**MangadexClient**(*, *username: Optional[str] = None*, *password: Optional[str] = None*, *refresh_token: Optional[str] = None*, *sleep_on_ratelimit: bool = True*, *session: Optional[aiohttp.client.ClientSession] = None*, *api_url: str = 'https://api.mangadex.org'*, *anonymous: bool = False*, ***session_kwargs*)

The main client that runs preforms all of the method requests.

> **Warning:** The client object should only be created under an async context. While it should be safe to initialize normally, the aiohttp ClientSession does not like this.

Parameters

- **username** (*str*) – The username of the user to authenticate as. Leave blank to not allow login to fetch a new refresh token. Specifying the username without specifying the password is an error.
- **password** (*str*) – The password of the user to authenticate as. Leave blank to not allow login to fetch a new refresh token. Specifying the password without specifying the username is an error.
- **refresh_token** (*str*) – The refresh token to use. Leaving the `username` and `password` parameters blank but specifying this parameter allows the client to make requests using the refresh token for as long as it is valid. Once the refresh token is invalid, if the username and password are not specified, the client will throw *Unauthorized*, unless *logout()* is used to set the client to anonymous mode.
- **sleep_on_ratelimit** (*bool*) – Whether or not to sleep when a ratelimit occurs or raise a *Ratelimit*. Defaults to True.
- **session** (*aiohttp.ClientSession*) – The session object for the client to use. If one is not provided, the client will create a new session instead. This is useful for providing a custom session.
- **anonymous** (*bool*) – Whether or not to force anonymous mode. This will clear the username and/or password.
- **session_kwargs** – Optional keyword arguments to pass on to the `aiohttp.ClientSession`.

anonymous_mode: *bool*
Whether or not the client is operating in **Anonymous Mode**, where it only accesses public endpoints.

api_base: *str*
The base URL for the MangaDex API, without a slash at the end.

async get_session_token()
Get the session token and store it inside the client.

async login(*username: Optional[str] = None*, *password: Optional[str] = None*)
Logs in to the MangaDex API.

Parameters

- **username** (*str*) – Provide a username in order to make the client stop running in anonymous mode. Specifying the username without specifying the password is an error.
- **password** (*str*) – Provide a password in order to make the client stop running in anonymous mode. Specifying the password without specifying the username is an error.

async logout()
Log out from the API. If a refresh token exists, calls the logout route on the API. The username and password are cleared, and the client is put into anonymous mode.

password: *Optional[str]*
The password of the user that the client is logged in as. This will be None when the client is operating in anonymous mode.

ratelimits: *asyncdex.ratelimit.Ratelimits*
The *Ratelimits* object that the client is using.

refresh_token: *Optional[str]*
The refresh token that the client has obtained. This will be None when the client is operating in anonymous mode, as well as if the client has not obtained a refresh token from the API.

async request(*method: str*, *url: str*, ***, *params: Optional[Mapping[str, Union[str, Sequence[str]]]] = None*, *json: Optional[Any] = None*, *with_auth: bool = True*, *retries: int = 3*, ***session_request_kwargs*) → aiohttp.client_reqrep.ClientResponse
Perform a request.

> **Warning:** All requests have to be released, otherwise connections will not be reused. Make sure to call `aiohttp.ClientResponse.release()` on the object returned by the method if you do not read data from the response.

Parameters

- **method** (*str*) – The HTTP method to use for the request.
- **url** (*str*) – The URL to use for the request. May be either an absolute URL or a URL relative to the base MangaDex API URL.
- **params** (*Mapping[str, Union[str, Sequence[str]]]*) – Optional query parameters to append to the URL. If one of the values of the parameters is an array, the elements will be automatically added to the URL in the order that the array elements appear in.
- **json** (*Any*) – JSON data to pass in a POST request.
- **with_auth** (*bool*) – Whether or not to append the session token to the request headers. Requests made without the header will behave as if the client is in anonymous mode. Defaults to `True`.

- **retries** (*int*) – The amount of times to retry. The function will recursively call itself, subtracting `1` from the original count until retries run out.
- **session_request_kwargs** – Optional keyword arguments to pass to `aiohttp.ClientSession.request()`.

Raises *Unauthorized* if the endpoint requires authentication and sufficient parameters for authentication were not provided to the client.

Returns The response.

Return type aiohttp.ClientResponse

session: aiohttp.client.ClientSession
The `aiohttp.ClientSession` that the client will use to make requests.

property session_token
The session token tht the client has obtained. This will be None when the client is operating in anonymous mode, as well as if the client has not obtained a refresh token from the API or if it has been roughly 15 minutes since the token was retrieved from the server.

sleep_on_ratelimit: bool
Whether or not to sleep when a ratelimit occurs.

username: Optional[str]
The username of the user that the client is logged in as. This will be None when the client is operating in anonymous mode.

3.2 Models

3.3 Exceptions

exception asyncdex.exceptions.**AsyncDexException**
Base exception class for all exceptions by the package.

exception asyncdex.exceptions.**Ratelimit**(*path: str*, *ratelimit_amount: int*, *ratelimit_expires: datetime.datetime*)
An exception raised if `sleep_on_ratelimit` is set to False.

path: str
The route that was taken that hit the ratelimit. This will match the path in the MangaDex API Documentation.

ratelimit_amount: int
How many calls to this path can be made once the ratelimit expires without being ratelimited again.

ratelimit_expires: datetime.datetime
A `datetime.datetime` object in UTC time representing when the ratelimit will expire.

exception asyncdex.exceptions.**HTTPException**(*path: str*, *response: aiohttp.client_reqrep.ClientResponse*)
Exceptions for HTTP status codes.

path: str
The URL taken that hit the error.

response: aiohttp.client_reqrep.ClientResponse
The `aiohttp.ClientResponse` object from the request.

exception asyncdex.exceptions.**Unauthorized**(*path: str*, *response: Optional[aiohttp.client_reqrep.ClientResponse]*)

An exception raised if a request to an endpoint requiring authorization is made where the client cannot authorize using provided information.

response: Optional[aiohttp.client_reqrep.ClientResponse]

The `aiohttp.ClientResponse` object from the request. May be `None` if a user tries to login without stored credentials.

3.4 Constants

asyncdex.constants.**ratelimit_data: List[*asyncdex.ratelimit.PathRatelimit*]**

These are the ratelimit rules taken from the API Docs.

Note: The API rules given here do not reflect all possible API ratelimit rules. The client will automatically ratelimit when appropriate headers are sent by the API. Check the latest API rules at the official API documentation.

New in version 0.1.

asyncdex.constants.**routes: Dict[str, str]**

The various predefined routes for the client. If the API changes for a given destination, the route can easily be modified without copy-pasting the route to the functions using it.

3.5 Ratelimit

class asyncdex.ratelimit.**Path**(*name: str*, *path_regex: re.Pattern*, *method: Optional[str] = None*)

A Path object representing a various path.

method: Optional[str] = None

The HTTP method for the path. Leave None if ratelimit applies to all methods.

name: str

The name of the path. This will be the value provided by `Ratelimit.path`.

path_regex: re.Pattern

A compiled regex pattern matching the path, used when the path has a variable, such as `/action/{id}`.

class asyncdex.ratelimit.**PathRatelimit**(*path:* asyncdex.ratelimit.Path, *ratelimit_amount: int*, *ratelimit_time: int*)

An object that allows the request method to check the ratelimit before making a response.

can_call(*method: str*) → bool

Returns whether or not this route can be used right now.

Parameters **method** (*str*) – The HTTP method being used.

Returns Whether or not this route can be used without ratelimit.

Return type bool

expire()

Expire the ratelimit.

path: *asyncdex.ratelimit.Path*

A *Path* object.

ratelimit_amount: int
Analogous to Ratelimit.ratelimit_amount

ratelimit_expires: datetime.datetime = datetime.datetime(1, 1, 1, 0, 0)
Analogous to Ratelimit.ratelimit_expires

ratelimit_time: int
The amount of time needed for the ratelimit to expire after the first use.

ratelimit_used: int = 0
How many times the path has been called since the last ratelimit expire.

time_until_expire() → datetime.timedelta
Returns a datetime.timedelta representing the amount of seconds for the ratelimit to expire.

update(*response: aiohttp.client_reqrep.ClientResponse*)
Update the path's ratelimit based on the headers.

Parameters **response** (*aiohttp.ClientResponse*) – The response object.

class asyncdex.ratelimit.**Ratelimits**(**ratelimits:* asyncdex.ratelimit.PathRatelimit)
An object holding all of the various ratelimits.

Parameters **ratelimits** (PathRatelimit) – The *PathRatelimit* object.

add(*obj:* asyncdex.ratelimit.PathRatelimit)
Add a new ratelimit. If the path is the same as an existing path, it will be overwritten.

Parameters **obj** (PathRatelimit) – The new ratelimit object to add.

async check(*url: str*, *method: str*) → Tuple[float, Optional[*asyncdex.ratelimit.PathRatelimit*]]
Check if a path is ratelimited.

Parameters

- **url** (*str*) – The path, starting with /
- **method** (*str*) – The HTTP method being used.

Returns A number representing the amount of seconds before ratelimit expire or -1 if there is no need to ratelimit as well as the *PathRatelimit* object if found.

Return type float

ratelimit_dictionary: Dict[re.Pattern, ***asyncdex.ratelimit.PathRatelimit***]
A dictionary where the keys are regex patterns representing the paths and the values are *PathRatelimit* objects.

remove(*obj:* asyncdex.ratelimit.PathRatelimit)
Remove a ratelimit.

Parameters **obj** (PathRatelimit) – The new ratelimit object to remove.

async sleep(*url: str*, *method: str*) → Optional[*asyncdex.ratelimit.PathRatelimit*]
Helper function that sleeps the amount of time returned by *check()*.

Parameters

- **url** (*str*) – The path, starting with /
- **method** (*str*) – The HTTP method being used.

Returns The *PathRatelimit* object if found

Return type Optional[*PathRatelimit*]

3.6 Misc

asyncdex.utils.**remove_prefix**(*prefix: str*, *string: str*) → str

Remove a prefix from a string. This is a polyfill for Python versions <3.9.

Parameters

- **prefix** (*str*) – The prefix to remove
- **string** (*str*) – The string to remove the prefix from

Returns The string without the prefix

Return type str

INDEX

T

U